aws

Entwicklerhandbuch

AWS Serverless Application Model



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

AWS Serverless Application Model: Entwicklerhandbuch

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

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

Table of Contents

Was ist AWS SAM? ... 1
Wann sollte dies verwendet werden? AWS SAM ... 1
Schlüssel-Features ... 2
Ähnliche Informationen ... 3
Funktionsweise ... 3
Was ist der AWS SAM Vorlagenspezifikation? ... 4
Was ist der AWS SAM projekt und AWS SAM Vorlage? ... 4
Was ist der AWS SAM CLI? ... 11
Weitere Informationen ... 19
Nächste Schritte ... 19
Serverlose Konzepte ... 19
Serverlose Konzepte ... 20
Infrastruktur als Code ... 21
IaC-Tools für serverlose Anwendungen AWS ... 21
Vergleich der IaC-Tools für serverlose Anwendungen ... 22
Weitere Informationen ... 23
Erste Schritte ... 25
Voraussetzungen ... 25
Melde dich an für AWS-Konto ... 26
Schritt 1: Installieren Sie AWS CLI ... 26
Schritt 2: Verwenden Sie den AWS CLI zur Konfiguration AWS Anmeldedaten ... 26
Schritt 3: (Optional) Installieren AWS Toolkit für VS Code ... 27
Nächste Schritte ... 28
Installieren Sie das AWS SAMCLI ... 28
Installation des AWS SAMCLI ... 29
Behebung von Installationsfehlern ... 41
Nächste Schritte ... 42
Optional: Überprüfen Sie das AWS SAMCLI Installationsprogramm ... 43
Hello World-Tutorial ... 56
Voraussetzungen ... 57
Schritt 1: Initialisieren Sie die Hello World-Beispielanwendung ... 58
Schritt 2: Erstellen Sie Ihre Anwendung ... 62
Schritt 3: Stellen Sie Ihre Anwendung auf dem bereit AWS Cloud ... 64
Schritt 4: Führen Sie Ihre Anwendung aus ... 69

Schritt 5: Interagieren Sie mit Ihrer Funktion im AWS Cloud ... 70
Schritt 6: Ändern und synchronisieren Sie Ihre Anwendung mit AWS Cloud ... 71
Schritt 7: (Optional) Testen Sie Ihre Anwendung lokal ... 74
Schritt 8: Löschen Sie Ihre Anwendung aus dem AWS Cloud ... 77
Fehlerbehebung ... 77
Weitere Informationen ... 77
Konvertierung von Lambda ... 78
Weitere Informationen ... 79
Wie benutzt man AWS SAM ... 80
AWS SAM CLI ... 81
Wie AWS SAMCLI werden Befehle dokumentiert ... 81
Konfiguration der AWS SAMCLI ... 83
So erstellen Sie Ihre Konfigurationsdatei (die `samconfig` Datei) ... 84
Konfigurieren Sie die Projekteinstellungen ... 84
Konfigurieren Sie Anmeldeinformationen und Grundeinstellungen ... 89
Kernbefehle ... 89
Lokales Testen mit AWS SAMCLI ... 90
Testen Sie AWS SAM Anwendungen lokal ... 91
Testen Sie Vorlagen lokal CloudFormation ... 91
Testen Sie AWS CDK-Anwendungen lokal ... 91
Testen Sie Terraform-Anwendungen lokal ... 92
AWS SAM Schablone ... 93
Aufbau einer Vorlage ... 94
YAML ... 95
Abschnitte einer Vorlage ... 95
Nächste Schritte ... 98
Globale ... 98
Ressourcen und Immobilien ... 104
AWS::Serverless::Api ... 105
AWS::Serverless::Application ... 166
AWS::Serverless::CapacityProvider ... 171
AWS::Serverless::Connector ... 183
AWS::Serverless: :Funktion ... 198
AWS::Serverless::GraphQLApi ... 363
AWS::Serverless::HttpApi ... 400
AWS::Serverless::WebSocketApi ... 433

AWS::Serverless::LayerVersion ... 453
AWS::Serverless::MicrovmImage ... 459
AWS::Serverless::NetworkConnector ... 467
AWS::Serverless::SimpleTable ... 473
AWS::Serverless::StateMachine ... 479
Generierte Ressourcen ... 521
Die Referenzierung wurde generiert CloudFormation Ressourcen ... 522
Generiert CloudFormation Ressourcenszenarien ... 523
AWS::Serverless::Api ... 526
AWS::Serverless::Application ... 528
AWS::Serverless::CapacityProvider ... 528
AWS::Serverless::Connector ... 529
AWS::Serverless::Function ... 530
AWS::Serverless::GraphQLApi ... 536
AWS::Serverless::HttpApi ... 538
AWS::Serverless::MicrovmImage ... 540
AWS::Serverless::NetworkConnector ... 541
AWS::Serverless::WebSocketApi ... 542
AWS::Serverless::LayerVersion ... 545
AWS::Serverless::SimpleTable ... 545
AWS::Serverless::StateMachine ... 546
Unterstützte Ressourcenattribute ... 547
Ausnahmen ... 548
API-Gateway-Erweiterungen ... 549
Intrinsische Funktionen ... 551
Unterstützung für Spracherweiterungen ... 551
Spracherweiterungen aktivieren ... 552
Fn:: ForEach ... 554
Unterstützte intrinsische Funktionen ... 561
Validierungsfehler ... 562
Einschränkungen ... 563
Telemetrie ... 564
Beispielanwendungen ... 565
DynamoDB-Ereignisse verarbeiten ... 565
Bevor Sie beginnen ... 565
Schritt 1: Initialisieren Sie die Anwendung ... 565

Schritt 2: Testen Sie die Anwendung lokal ... 566
Schritt 3: Verpacken Sie die Anwendung ... 566
Schritt 4: Stellen Sie die Anwendung bereit ... 567
Nächste Schritte ... 568
Amazon S3 S3-Ereignisse verarbeiten ... 568
Bevor Sie beginnen ... 569
Schritt 1: Initialisieren Sie die Anwendung ... 569
Schritt 2: Verpacken Sie die Anwendung ... 569
Schritt 3: Stellen Sie die Anwendung bereit ... 570
Schritt 4: Testen Sie die Anwendung lokal ... 571
Nächste Schritte ... 572
Entwickeln Sie Ihre Anwendung ... 573
Erstellen Sie Ihre Bewerbung ... 573
Initialisieren Sie eine neue serverlose Anwendung ... 574
Optionen für Sam Init ... 580
Fehlerbehebung ... 581
Beispiele ... 581
Weitere Informationen ... 582
Nächste Schritte ... 582
Definieren Sie Ihre Infrastruktur ... 582
Definieren Sie Anwendungsressourcen ... 583
Zugriff einrichten ... 584
Kontrollieren Sie den API-Zugriff ... 665
Steigern Sie die Effizienz mit Schichten ... 679
Code wiederverwenden ... 682
Verwalten Sie zeitbasierte Ereignisse ... 686
Orchestrierung von Anwendungen ... 689
Richten Sie die Codesignatur ein ... 691
AWS SAM Vorlagendateien validieren ... 694
Erstellen Sie Ihre Anwendung ... 694
Einführung in sam build ... 695
Standard-Build ... 711
Passen Sie Ihren Build an ... 719
Testen Sie Ihre Anwendung ... 753
Einführung in sam local ... 753
Verwenden des sam local Befehls ... 754

Einführung in sam local generate-event ... 755
Einführung in sam local invoke ... 761
Einführung in sam local start-api ... 768
Einführung in sam local start-lambda ... 774
Lokal Funktionen aufrufen ... 777
Umgebungsvariablendatei ... 779
Layer ... 781
Weitere Informationen ... 781
Lokal ausgeführtes API Gateway ... 781
Umgebungsvariablendatei ... 783
Layer ... 784
Testen Sie mit sam remote test-event ... 784
Richten Sie das ein AWS SAMCLI, um es zu verwenden sam remote test-event ... 785
Verwenden Sie den Befehl sam remote test-event ... 786
Testereignisse verwenden, die gemeinsam genutzt werden können ... 789
Verwaltung gemeinsam nutzbarer Testereignisse ... 789
Testen Sie mit sam remote invoke ... 791
Verwenden Sie den Befehl sam remote invoke ... 792
Verwenden von Sam Remote Invoke-Befehlsoptionen ... 796
Konfigurieren Sie Ihre Projektkonfigurationsdatei ... 802
Beispiele ... 802
Weiterführende Links ... 818
Automatisieren Sie Integrationstests ... 818
Generieren Sie Beispiel-Payloads ... 820
Testen und debuggen Sie langlebige Funktionen ... 821
Lokaler Test-Workflow ... 821
Allgemeine Testszenarien ... 822
Fehlerbehebung ... 822
Tipps zum Debuggen ... 823
Weitere Informationen ... 823
Debuggen Sie Ihre Anwendung ... 824
Lokales Debuggen von Funktionen ... 824
Verwenden von AWS Toolkits ... 825
Wird AWS SAM lokal im Debug-Modus ausgeführt ... 827
Übergeben Sie mehrere Laufzeitargumente ... 828
Validieren Sie mit cfn-lint ... 829

Beispiele ... 829
Weitere Informationen ... 829
Stellen Sie Ihre Anwendung und Ressourcen bereit ... 830
Einführung in sam deploy ... 830
Voraussetzungen ... 831
Bereitstellen von Anwendungen mit Sam Deploy ... 831
Best Practices ... 841
Optionen für Sam Deploy ... 842
Fehlerbehebung ... 842
Beispiele ... 842
Weitere Informationen ... 851
Optionen für die Bereitstellung ... 851
Wie benutzt man den AWS SAMCLI für die manuelle Bereitstellung ... 851
Bereitstellen mit CI/CD Systemen und Pipelines ... 852
Schrittweise Bereitstellungen ... 852
Problembehandlung bei Bereitstellungen mit dem AWS SAMCLI ... 852
Weitere Informationen ... 781
Bereitstellung mit CI/CD Systemen und Pipelines ... 853
Was ist eine Pipeline? ... 854
Wie werden AWS SAM lokale Dateien hochgeladen ... 855
Generieren Sie eine Starter-Pipeline ... 863
Passen Sie Starter-Pipelines an ... 869
Automatisieren Sie Ihre Bereitstellungen ... 871
Verwenden Sie die OIDC-Authentifizierung ... 876
Einführung in sam sync ... 879
Erkennt automatisch lokale Änderungen und synchronisiert sie mit AWS Cloud ... 880
Passen Sie an, welche lokalen Änderungen mit dem synchronisiert werden AWS Cloud ... 881
Bereiten Sie Ihre Anwendung in der Cloud für Tests und Validierung vor ... 881
Optionen für den Befehl sam sync ... 882
Fehlerbehebung ... 884
Beispiele ... 884
Weitere Informationen ... 891
Überwachen Sie Ihre Anwendung ... 892
Einblicke in Anwendungen ... 892
Konfiguration von CloudWatch Application Insights mit AWS SAM ... 892
Nächste Schritte ... 896

Mit Protokollen arbeiten ... 896
Protokolle stapelweise CloudFormation abrufen ... 897
Logs nach Lambda-Funktionsnamen abrufen ... 897
Protokollierung von Protokollen ... 897
Protokolle für einen bestimmten Zeitraum anzeigen ... 897
Protokolle filtern ... 897
Fehler beim Hervorheben ... 898
JSON ist hübsch beim Drucken ... 898
AWS SAM Referenz ... 899
AWS SAM Spezifikation und Vorlage AWS SAM ... 899
AWS SAMCLIBefehlsreferenz ... 899
AWS SAM Richtlinienvorlagen ... 900
Topics ... 900
AWS SAMCLIBefehle ... 901
gleicher Build ... 902
sam delete ... 908
sam deploy ... 910
sam init ... 917
sam list ... 921
sam local callback ... 930
sam local execution ... 935
sam local generate-event ... 942
sam local invoke ... 943
Sam lokale Start-API ... 951
Sam Local Start-Lambda ... 957
sam logs ... 963
sam package ... 967
sam pipeline bootstrap ... 970
sam pipeline init ... 975
sam publish ... 977
sam remote callback ... 979
sam remote execution ... 985
sam remote invoke ... 992
sam remote test-event ... 997
Sam Sync ... 1005
sam traces ... 1012

sam validate ... 1014
AWS SAMCLIVerwaltung ... 1016
AWS SAMCLIKonfigurationsdatei ... 1016
AWS SAMCLIVersionen verwalten ... 1023
AWS Zugangsdaten einrichten ... 1033
AWS SAMCLITelemetrie ... 1035
Fehlerbehebung ... 1037
Steckverbinder-Referenz ... 1044
Unterstützte Connector-Ressourcentypen ... 1044
Von Konnektoren erstellte IAM-Richtlinien ... 1054
Installieren von Docker ... 1077
Installation von Docker ... 1078
Nächste Schritte ... 1081
Installation von Finch ... 1081
Installation von Finch ... 1082
Konfiguration Finch mit AWS SAM CLI ... 1084
Die Installation wird überprüft ... 1085
Stoppen von Finch ... 1086
Fehlerbehebung ... 1087
Nächste Schritte ... 1088
Bild-Repositorys ... 1088
URIs des Bild-Repositorys ... 1089
Beispiele ... 1091
Schrittweise Bereitstellung ... 1091
Schrittweise erstmalige Bereitstellung einer Lambda-Funktion ... 1095
Weitere Informationen ... 1096
Wichtige Hinweise ... 1096
2023 ... 1096
Terraform-Support ... 1098
Erste Schritte ... 1098
Voraussetzungen ... 1098
Verwenden AWS SAM CLI Befehle mit Terraform ... 1099
Richten Sie ein für Terraform projects ... 1099
Richten Sie ein für Terraform Cloud ... 1105
Verwenden mit AWS SAMCLI Terraform ... 1107
Lokales Testen mit gleicher lokaler Aufruf ... 1107

Lokales Testen mit dieselbe lokale Start-API ... 1108
Lokales Testen mit sam lokales Start-Lambda ... 1109
Terraform Einschränkungen ... 1110
Verwenden mit AWS SAMCLI Serverless.tf ... 1110
Terraform-Referenz ... 1111
AWS SAM Referenz zu den unterstützten Funktionen ... 1111
Terraform spezifische Referenz ... 1111
sam-Metadaten ... 1111
AWS SAMCLITerraformUnterstützung ... 1114
Was ist der AWS SAM CLI? ... 1115
Wie verwende ich den AWS SAM CLI mit Terraform? ... 1116
Nächste Schritte ... 1116
Veröffentlichung zur Nutzung durch andere ... 1117
Voraussetzungen ... 1117
Eine neue Anwendung veröffentlichen ... 1119
Schritt 1: Fügen Sie der AWS SAM Vorlage einen `Metadata` Abschnitt hinzu ... 1119
Schritt 2: Verpacken Sie die Anwendung ... 1120
Schritt 3: Veröffentlichen Sie die Anwendung ... 1120
Schritt 4: Teilen Sie die Anwendung (optional) ... 1120
Veröffentlichen einer neuen Version einer vorhandenen Anwendung ... 1121
Weitere Themen ... 1121
Eigenschaften des Metadaten-Abschnitts ... 1121
Eigenschaften ... 1122
Anwendungsfälle ... 1125
Beispiel ... 1126
Dokumentverlauf ... 1127
... mcliv

Was ist der AWS Serverless Application Model (AWS SAM)?

AWS Serverless Application Model (AWS SAM) ist ein Open-Source-Framework für die Erstellung serverloser Anwendungen unter Verwendung von Infrastructure as Code (IaC). Mit AWS SAM der Kurzsyntax deklarieren Entwickler CloudFormationRessourcen und spezialisierte serverlose Ressourcen, die während der Bereitstellung in Infrastruktur umgewandelt werden. Wenn Sie mit arbeiten AWS SAM, werden Sie interagieren mit:

1. AWS SAM CLI- Ein Befehlszeilentool, mit dem Sie Ihre serverlosen Anwendungen entwickeln, lokal testen und bereitstellen können.
2. AWS SAM Vorlage — Eine Erweiterung davon bietet eine vereinfachte Syntax für CloudFormation die Definition serverloser Ressourcen.

Wenn Sie den sam init Befehl verwenden, erstellt er ein Projektverzeichnis, das wir als AWS SAM Projekt bezeichnen werden. Es enthält in der Regel Ihre AWS SAM Vorlage, Ihren Anwendungscode und andere Konfigurationsdateien.

Wann sollte dies verwendet werden? AWS SAM

AWS SAM ist eine ideale IaC-Lösung für Szenarien, in denen Sie eine vereinfachte serverlose Entwicklung mit der vollen Leistung von wünschen. CloudFormation Beispielsweise können Sie SAM verwenden für:

- Serverlose Anwendungen: Mit SAM können Sie schnell Funktionen, langlebige AWS Lambda Lambda-Funktionen, Amazon API Gateway Gateway-APIs, Amazon DynamoDB-Tabellen und andere serverlose Ressourcen mit minimalem Code definieren.
- CloudFormation Verbesserung: Sie können SAM mit vorhandenen CloudFormation Vorlagen kombinieren, um der herkömmlichen Infrastruktur serverlose Komponenten hinzuzufügen. SAM-Ressourcen funktionieren zusammen mit CloudFormation Standardressourcen in derselben Vorlage.
- Lokale Entwicklung und Tests: Mit SAM CLI können Sie Lambda-Funktionen lokal testen, API-Gateway-Endpunkte simulieren und serverlose Anwendungen auf Ihrem Entwicklungscomputer debuggen, bevor Sie sie bereitstellen. AWS
- CI/CD für serverlose Umgebungen: Sie können Bereitstellungspipelines mithilfe von SAM-Vorlagen erstellen, die automatisch die für Staging- und Produktionsumgebungen benötigte CloudFormation Infrastruktur generieren.

- Migration von in der Konsole erstellten Ressourcen: Sie können Lambda-Funktionen und API-Gateway-Ressourcen, die in der AWS Management Console erstellt wurden, mithilfe von SAM-Vorlagen in Infrastruktur als Code konvertieren.

Vergleich AWS SAM mit anderen IaC-Tools

- Verwenden Sie SAM statt dessen CloudFormation , um serverlose Ressourcendefinitionen zu vereinfachen und gleichzeitig die Vorlagenkompatibilität zu wahren.
- Verwenden Sie SAM anstelle von, AWS CDK wenn Sie bei der Beschreibung Ihrer Infrastruktur eher einen deklarativen als einen programmatischen Ansatz bevorzugen.
- Kombinieren Sie SAM mit, AWS CDK indem Sie die lokalen Testfunktionen von SAM CLI verwenden, um Ihre CDK-Anwendungen zu verbessern.

Schlüssel-Features

AWS SAM bietet eine Vielzahl von Vorteilen, die das Entwicklererlebnis verbessern, indem sie Ihnen Folgendes ermöglichen:

Definieren Sie Ihren Anwendungsinfrastrukturcode schnell und mit weniger Code

Erstellen Sie AWS SAM Vorlagen, um den Infrastrukturcode für Ihre serverlose Anwendung zu definieren. Stellen Sie Ihre Vorlagen direkt bereit, CloudFormation um Ihre Ressourcen bereitzustellen.

Verwalten Sie Ihre serverlosen Anwendungen während ihres gesamten Entwicklungszyklus

Verwenden Sie den AWS SAMCLI, um Ihre serverlose Anwendung während der Phasen des Entwicklungsprozesses, der Erstellung, der Bereitstellung, des Tests und der Überwachung zu verwalten. Weitere Informationen finden Sie unter AWS SAM CLI.

Stellen Sie mithilfe von Konnektoren schnell Berechtigungen zwischen Ressourcen bereit AWS SAM

Verwenden Sie AWS SAM Konnektoren in Ihren AWS SAM Vorlagen, um Berechtigungen zwischen Ihren AWS Ressourcen zu definieren. AWS SAM wandelt Ihren Code in die IAM-Berechtigungen um, die zur Umsetzung Ihrer Absicht erforderlich sind. Weitere Informationen finden Sie unter Verwaltung von Ressourcenberechtigungen mit AWS SAM Steckverbinder.

Synchronisieren Sie während der Entwicklung kontinuierlich lokale Änderungen mit der Cloud

Verwenden Sie den AWS SAMCLI sam sync Befehl, um lokale Änderungen automatisch mit der Cloud zu synchronisieren und so Ihre Entwicklungs- und Cloud-Test-Workflows zu beschleunigen. Weitere Informationen finden Sie unter Einführung in die Verwendung von sam sync to sync to AWS Cloud.

Verwalten Sie Ihre Terraform serverlosen Anwendungen

Verwenden Sie die AWS SAMCLI, um Ihre Lambda-Funktionen und -Layer lokal zu debuggen und zu testen. Weitere Informationen finden Sie unter AWS SAM CLI Terraform Support.

Ähnliche Informationen

- Informationen darüber, wie das AWS SAM funktioniert, finden Sie unter. Wie AWS SAM funktioniert
- Informationen zum Einstieg in die Nutzung AWS SAM finden Sie unterErste Schritte mit AWS SAM.
- Eine Übersicht darüber, wie Sie AWS SAM damit eine serverlose Anwendung erstellen können, finden Sie unterWie benutzt man AWS SAM.

Wie AWS SAM funktioniert

Bei der Erstellung AWS SAM Ihrer serverlosen Anwendung werden Sie mit den folgenden Komponenten interagieren:

1. AWS SAM Schablone— Eine wichtige Datei, die Ihre AWS Ressourcen definiert. Diese Vorlage enthält die AWS SAM Vorlagenspezifikation — das Open-Source-Framework mit einer vereinfachten Kurzsyntax, mit der Sie die Funktionen, Ereignisse, APIs, Konfigurationen und Berechtigungen Ihrer serverlosen Anwendung definieren. Diese Datei befindet sich im AWS SAM Projekt, dem Anwendungsordner, der erstellt wird, wenn Sie den Befehl ausführen. sam init
2. AWS SAM CLI— Ein Befehlszeilentool, das Sie mit Ihrem AWS SAM Projekt und unterstützten Integrationen von Drittanbietern verwenden können, um Ihre serverlosen Anwendungen zu erstellen und auszuführen. Das AWS SAMCLI ist das Tool, mit dem Sie Befehle für Ihr AWS SAM Projekt ausführen und es schließlich in Ihre serverlose Anwendung umwandeln.

Um Ressourcen, Zuordnungen von Ereignisquellen und andere Eigenschaften auszudrücken, die Ihre serverlose Anwendung definieren, definieren Sie Ressourcen und entwickeln Ihre Anwendung in der AWS SAM Vorlage und anderen Dateien in Ihrem Projekt. AWS SAM Sie verwenden die

AWS SAMCLI, um Befehle in Ihrem AWS SAM Projekt auszuführen. Auf diese Weise initialisieren, erstellen, testen und implementieren Sie Ihre serverlose Anwendung.

> Sind Sie neu im Bereich Serverless?
>
> Wir empfehlen Ihnen, es zu überprüfen[Serverlose Konzepte für AWS Serverless Application Model](#).

Was ist der AWS SAM Vorlagenspezifikation?

Die AWS SAM Vorlagenspezifikation ist ein Open-Source-Framework, mit dem Sie Ihren Infrastrukturcode für serverlose Anwendungen definieren und verwalten können. Die AWS SAM Vorlagenspezifikation lautet:

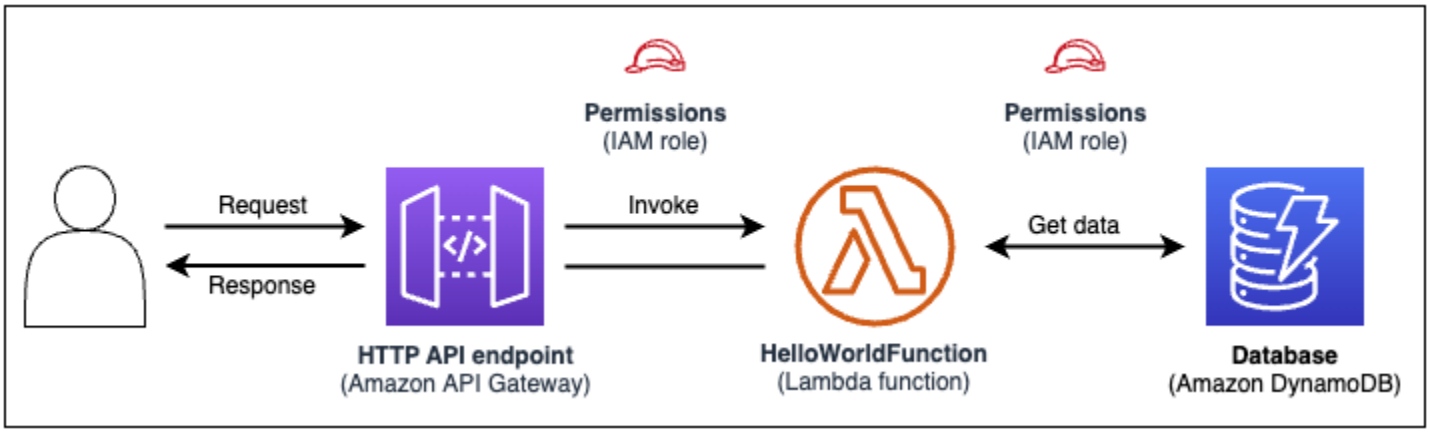
- Darauf AWS CloudFormation aufbauend — Sie verwenden die CloudFormation Syntax direkt in Ihrer AWS SAM Vorlage und profitieren dabei von der umfassenden Unterstützung von Ressourcen- und Eigenschaftenkonfigurationen. Wenn Sie bereits damit vertraut sind CloudFormation, müssen Sie sich nicht erst mit einem neuen Service vertraut machen, um Ihren Anwendungsinfrastrukturcode zu verwalten.
- Eine Erweiterung von CloudFormation — AWS SAM bietet eine eigene, einzigartige Syntax, die sich speziell auf die Beschleunigung der serverlosen Entwicklung konzentriert. Sie können sowohl die AWS SAM Syntax als CloudFormation auch innerhalb derselben Vorlage verwenden.
- Eine abstrakte, kurze Syntax — Mithilfe der AWS SAM Syntax können Sie Ihre Infrastruktur schnell, mit weniger Codezeilen und mit einer geringeren Fehlerwahrscheinlichkeit definieren. Die Syntax wurde speziell entwickelt, um die Komplexität bei der Definition Ihrer serverlosen Anwendungsinfrastruktur zu verringern.
- Transformativ — AWS SAM übernimmt die komplexe Aufgabe, Ihre Vorlage in den Code umzuwandeln, der für die Bereitstellung Ihrer Infrastruktur erforderlich ist. CloudFormation

Was ist der AWS SAM projekt und AWS SAM Vorlage?

Das AWS SAM Projekt enthält die AWS SAM Vorlage, die die AWS SAM Vorlagenspezifikation enthält. Bei dieser Spezifikation handelt es sich um das Open-Source-Framework, mit dem Sie Ihre serverlose Anwendungsinfrastruktur definieren. Es enthält einige zusätzliche Komponenten AWS, die die Arbeit mit ihnen erleichtern. In diesem Sinne sind AWS SAM Vorlagen eine Erweiterung von CloudFormation Vorlagen.

Hier ist ein Beispiel für eine einfache serverlose Anwendung. Diese Anwendung verarbeitet Anfragen, um alle Elemente aus einer Datenbank über eine HTTP-Anfrage abzurufen. Sie besteht aus den folgenden Teilen:

1. Eine Funktion, die die Logik zur Verarbeitung der Anfrage enthält.
2. Eine HTTP-API, die als Kommunikation zwischen dem Client (Anforderer) und der Anwendung dient.
3. Eine Datenbank zum Speichern von Elementen.
4. Berechtigungen für die sichere Ausführung der Anwendung.



Der Infrastrukturcode dieser Anwendung kann in der folgenden AWS SAM Vorlage definiert werden:

```
AWSTemplateFormatVersion: 2010-09-09
Transform: AWS::Serverless-2016-10-31
Resources:
  getAllItemsFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: src/get-all-items.getAllItemsHandler
      Runtime: nodejs20.x
      Events:
        Api:
          Type: HttpApi
          Properties:
            Path: /
            Method: GET
    Connectors:
      MyConn:
        Properties:
```

```
          Destination:
            Id: SampleTable
          Permissions:
            - Read
  SampleTable:
    Type: AWS::Serverless::SimpleTable
```

In 23 Codezeilen ist die folgende Infrastruktur definiert:

- Eine Funktion, die den AWS Lambda Dienst verwendet.
- Eine HTTP-API, die den Amazon API Gateway Gateway-Service verwendet.
- Eine Datenbank, die den Amazon DynamoDB-Service verwendet.
- Die AWS Identity and Access Management (IAM-) Berechtigungen, die erforderlich sind, damit diese Dienste miteinander interagieren können.

Um diese Infrastruktur bereitzustellen, wird die Vorlage auf CloudFormation bereitgestellt. AWS SAM Transformiert während der Bereitstellung die 23 Codezeilen in die CloudFormation Syntax, in der diese Ressourcen generiert werden. AWS Die transformierte CloudFormation Vorlage enthält über 200 Codezeilen!

Transformiert CloudFormation Vorlage

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Resources": {
    "getAllItemsFunction": {
      "Type": "AWS::Lambda::Function",
      "Metadata": {
        "SamResourceId": "getAllItemsFunction"
      },
      "Properties": {
        "Code": {
          "S3Bucket": "amzn-s3-demo-source-bucket-1a4x26zbcdkqr",
          "S3Key": "what-is-app/a6f856abf1b2c4f7488c09b367540b5b"
        },
        "Handler": "src/get-all-items.getAllItemsHandler",
        "Role": {
          "Fn::GetAtt": [
            "getAllItemsFunctionRole",
            "Arn"
          ]
```

```
        },
        "Runtime": "nodejs12.x",
        "Tags": [
          {
            "Key": "lambda:createdBy",
            "Value": "SAM"
          }
        ]
      }
    },
    "getAllItemsFunctionRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Action": [
                "sts:AssumeRole"
              ],
              "Effect": "Allow",
              "Principal": {
                "Service": [
                  "lambda.amazonaws.com"
                ]
              }
            }
          ]
        },
        "ManagedPolicyArns": [
          "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
        ],
        "Tags": [
          {
            "Key": "lambda:createdBy",
            "Value": "SAM"
          }
        ]
      }
    },
    "getAllItemsFunctionApiPermission": {
      "Type": "AWS::Lambda::Permission",
      "Properties": {
        "Action": "lambda:InvokeFunction",
```

```
        "FunctionName": {
          "Ref": "getAllItemsFunction"
        },
        "Principal": "apigateway.amazonaws.com",
        "SourceArn": {
          "Fn::Sub": [
            "arn:${AWS::Partition}:execute-api:${AWS::Region}:${AWS::AccountId}:
${__ApiId__}/${__Stage__}/GET/",
            {
              "__ApiId__": {
                "Ref": "ServerlessHttpApi"
              },
              "__Stage__": "*"
            }
          ]
        }
      }
    },
    "ServerlessHttpApi": {
      "Type": "AWS::ApiGatewayV2::Api",
      "Properties": {
        "Body": {
          "info": {
            "version": "1.0",
            "title": {
              "Ref": "AWS::StackName"
            }
          },
          "paths": {
            "/": {
              "get": {
                "x-amazon-apigateway-integration": {
                  "httpMethod": "POST",
                  "type": "aws_proxy",
                  "uri": {
                    "Fn::Sub": "arn:${AWS::Partition}:apigateway:
${AWS::Region}:lambda:path/2015-03-31/functions/${getAllItemsFunction.Arn}/invocations"
                  },
                  "payloadFormatVersion": "2.0"
                },
                "responses": {}
              }
            },
```

```
          "openapi": "3.0.1",
          "tags": [
            {
              "name": "httpapi:createdBy",
              "x-amazon-apigateway-tag-value": "SAM"
            }
          ]
        }
      }
    },
    "ServerlessHttpApiApiGatewayDefaultStage": {
      "Type": "AWS::ApiGatewayV2::Stage",
      "Properties": {
        "ApiId": {
          "Ref": "ServerlessHttpApi"
        },
        "StageName": "$default",
        "Tags": {
          "httpapi:createdBy": "SAM"
        },
        "AutoDeploy": true
      }
    },
    "SampleTable": {
      "Type": "AWS::DynamoDB::Table",
      "Metadata": {
        "SamResourceId": "SampleTable"
      },
      "Properties": {
        "AttributeDefinitions": [
          {
            "AttributeName": "id",
            "AttributeType": "S"
          }
        ],
        "KeySchema": [
          {
            "AttributeName": "id",
            "KeyType": "HASH"
          }
        ],
        "BillingMode": "PAY_PER_REQUEST"
      }
    },
```

```
    "getAllItemsFunctionMyConnPolicy": {
      "Type": "AWS::IAM::ManagedPolicy",
      "Metadata": {
        "aws:sam:connectors": {
          "getAllItemsFunctionMyConn": {
            "Source": {
              "Type": "AWS::Serverless::Function"
            },
            "Destination": {
              "Type": "AWS::Serverless::SimpleTable"
            }
          }
        }
      },
      "Properties": {
        "PolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Effect": "Allow",
              "Action": [
                "dynamodb:GetItem",
                "dynamodb:Query",
                "dynamodb:Scan",
                "dynamodb:BatchGetItem",
                "dynamodb:ConditionCheckItem",
                "dynamodb:PartiQLSelect"
              ],
              "Resource": [
                {
                  "Fn::GetAtt": [
                    "SampleTable",
                    "Arn"
                  ]
                },
                {
                  "Fn::Sub": [
                    "${DestinationArn}/index/*",
                    {
                      "DestinationArn": {
                        "Fn::GetAtt": [
                          "SampleTable",
                          "Arn"
                        ]
```

```
                    }
                  }
                ]
              }
            ]
          }
        ]
      },
      "Roles": [
        {
          "Ref": "getAllItemsFunctionRole"
        }
      ]
    }
  }
}
```

Mithilfe AWS SAM von definieren Sie 23 Zeilen Infrastrukturcode. AWS SAM wandelt Ihren Code in die über 200 CloudFormation Codezeilen um, die für die Bereitstellung Ihrer Anwendung erforderlich sind.

Was ist der AWS SAM CLI?

Das AWS SAMCLI ist ein Befehlszeilentool, das Sie zusammen mit AWS SAM Vorlagen und unterstützten Integrationen von Drittanbietern verwenden können, um Ihre serverlosen Anwendungen zu erstellen und auszuführen. Verwenden Sie das, um AWS SAMCLI:

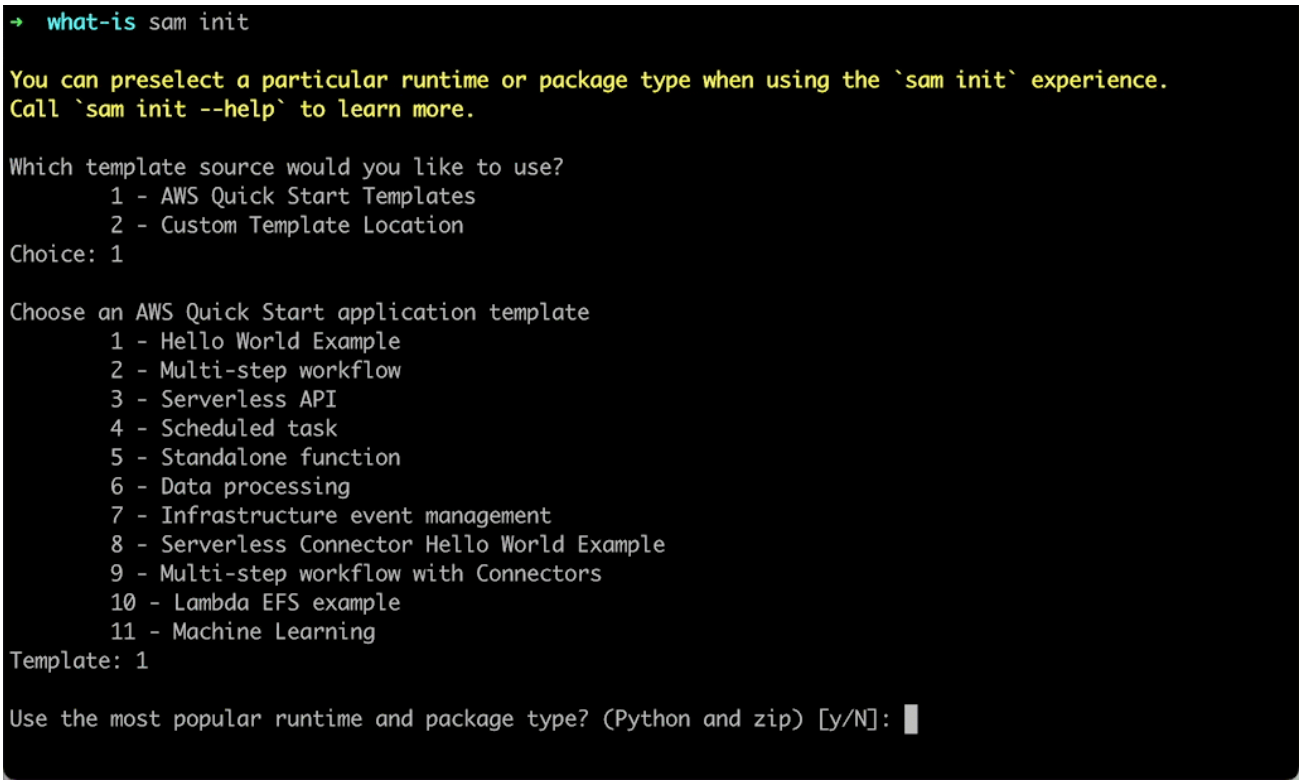
- Initialisieren Sie schnell ein neues Anwendungsprojekt.
- Erstellen Sie Ihre Anwendung für die Bereitstellung.
- Führen Sie lokales Debugging und Testen durch.
- Stellen Sie die Anwendung bereit.
- Konfigurieren Sie CI/CD Bereitstellungspipelines.
- Überwachen Sie Ihre Anwendung in der Cloud und beheben Sie Fehler.
- Synchronisieren Sie lokale Änderungen während der Entwicklung mit der Cloud.
- Und mehr!

Das AWS SAMCLI wird am besten verwendet, wenn es mit AWS SAM CloudFormation Vorlagen verwendet wird. Es funktioniert auch mit Produkten von Drittanbietern wieTerraform.

Initialisieren Sie ein neues Projekt

Wählen Sie aus den Startvorlagen oder wählen Sie einen Speicherort für benutzerdefinierte Vorlagen, um ein neues Projekt zu beginnen.

Hier verwenden wir den sam init Befehl, um ein neues Anwendungsprojekt zu initialisieren. Wir wählen zunächst das Hello World Example-Projekt aus. Das AWS SAMCLI lädt eine Startvorlage herunter und erstellt unsere Projektordner-Verzeichnisstruktur.

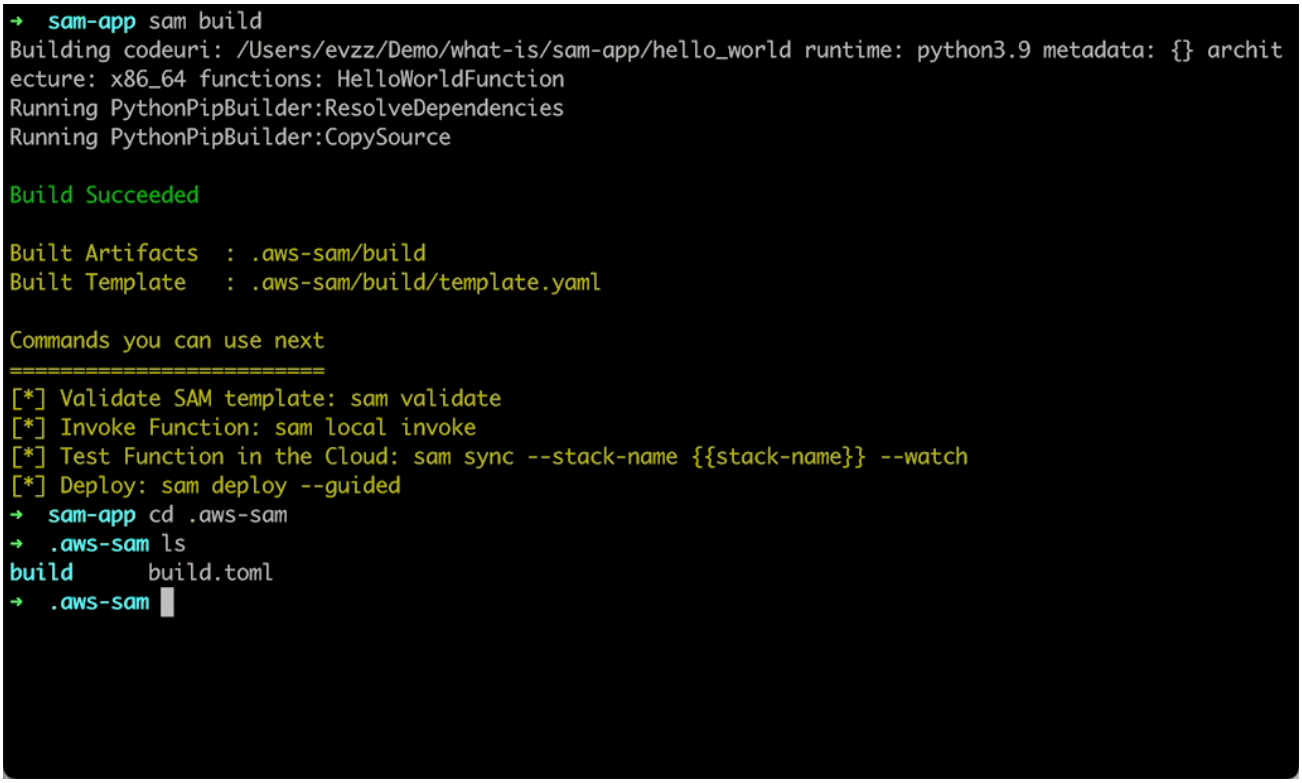


Weitere Details finden Sie unter Erstellen Sie Ihre Bewerbung in AWS SAM.

Erstellen Sie Ihre Anwendung für die Bereitstellung

Package Sie Ihre Funktionsabhängigkeiten und organisieren Sie Ihren Projektcode und Ihre Ordnerstruktur, um die Bereitstellung vorzubereiten.

Hier verwenden wir den sam build Befehl, um unsere Anwendung für die Bereitstellung vorzubereiten. Das AWS SAMCLI erstellt ein `.aws-sam` Verzeichnis und organisiert dort unsere Anwendungsabhängigkeiten und Dateien für die Bereitstellung.

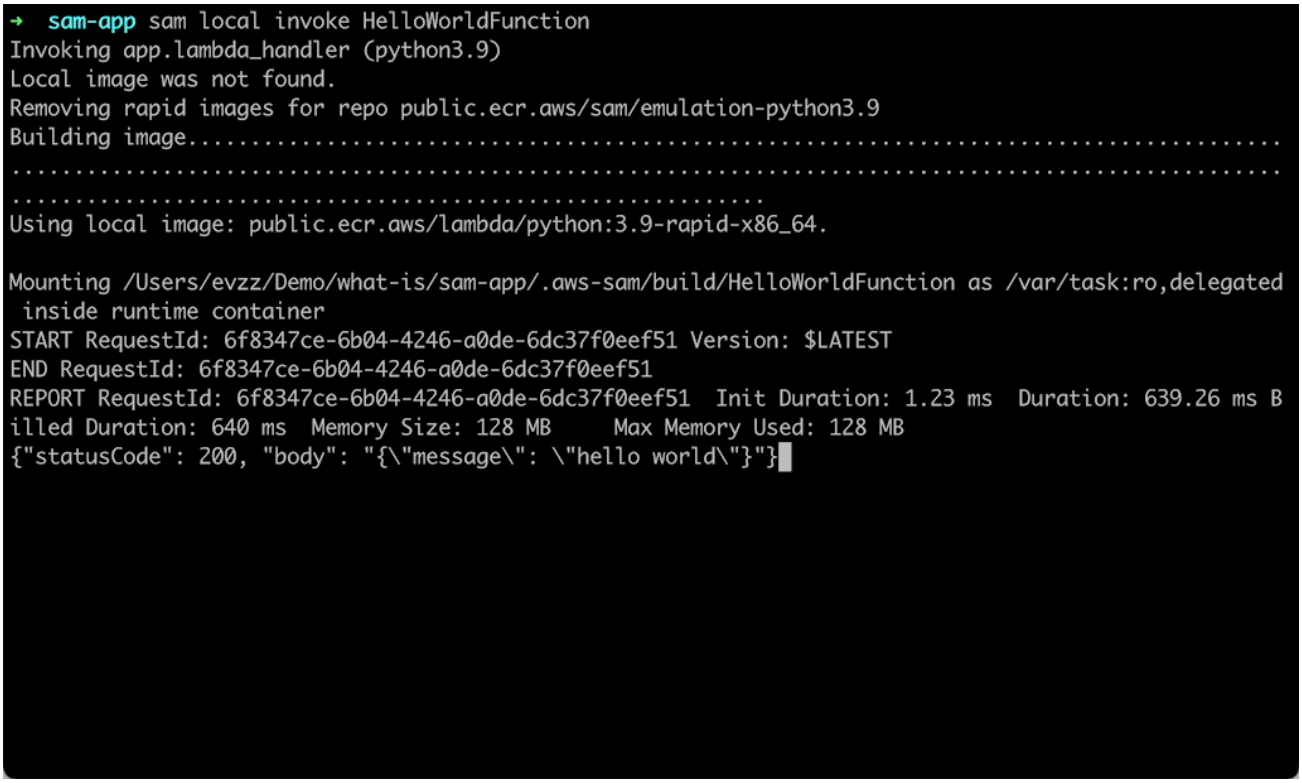


Weitere Details finden Sie unter Erstellen Sie Ihre Anwendung.

Führen Sie lokales Debugging und Testen durch

Simulieren Sie auf Ihrem lokalen Computer Ereignisse, testen Sie APIs, rufen Sie Funktionen auf und vieles mehr, um Ihre Anwendung zu debuggen und zu testen.

Hier verwenden wir den sam local invoke Befehl, um unsere lokal aufzurufen. `HelloWorldFunction` Um dies zu erreichen, AWS SAMCLI erstellt der einen lokalen Container, erstellt unsere Funktion, ruft sie auf und gibt die Ergebnisse aus. Sie können eine Anwendung wie Docker verwenden, um Container auf Ihrem Computer auszuführen.

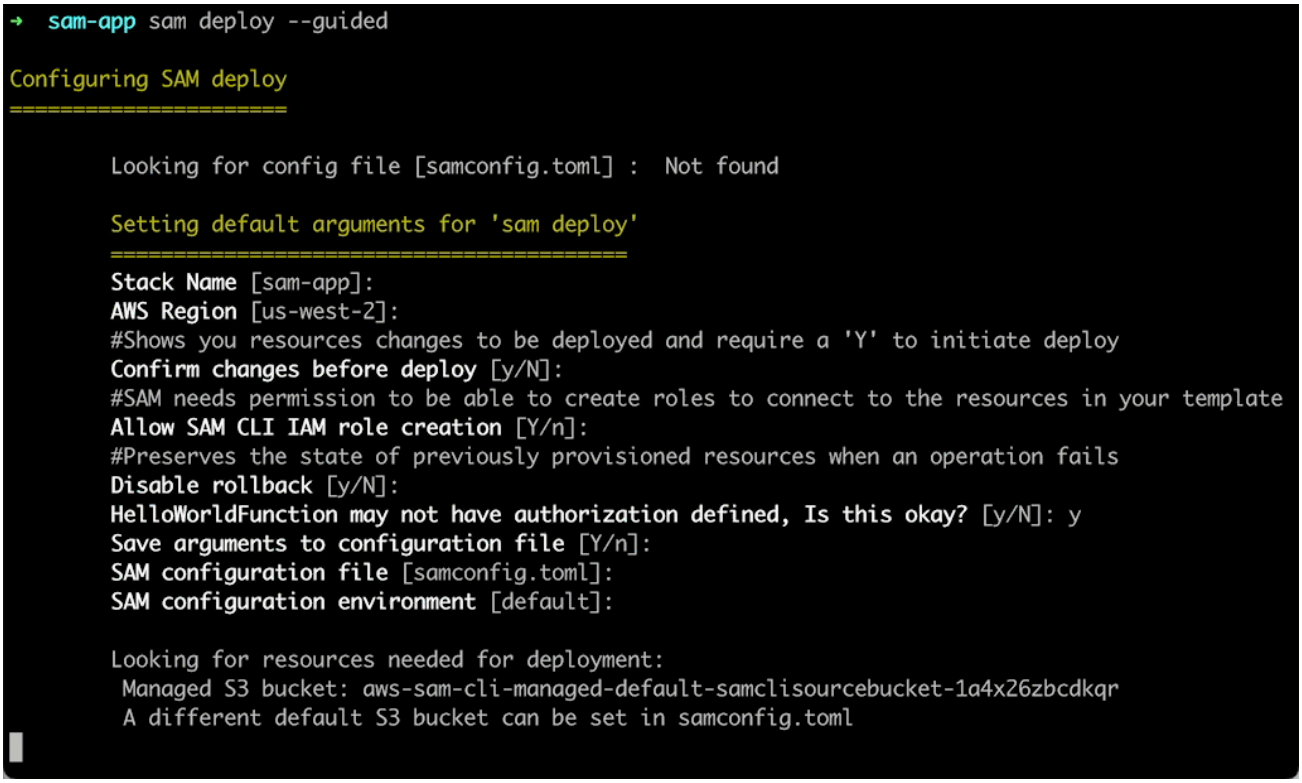


Weitere Informationen finden Sie unter Testen Sie Ihre Anwendung und Debuggen Sie Ihre Anwendung.

Bereitstellen der Anwendung

Konfigurieren Sie die Bereitstellungseinstellungen Ihrer Anwendung und stellen Sie sie in der AWS Cloud bereit, um Ihre Ressourcen bereitzustellen.

Hier verwenden wir den sam deploy --guided Befehl, um unsere Anwendung über einen interaktiven Ablauf bereitzustellen. Er AWS SAMCLI führt uns durch die Konfiguration der Bereitstellungseinstellungen unserer Anwendung, wandelt unsere Vorlage in Ressourcen um und stellt sie bereit CloudFormation, um unsere CloudFormation Ressourcen zu erstellen.



Weitere Details finden Sie unter Stellen Sie Ihre Anwendung und Ressourcen bereit.

CI/CD Bereitstellungspipelines konfigurieren

Erstellen Sie sichere Pipelines für kontinuierliche Integration und Bereitstellung (CI/CD) mithilfe eines unterstützten CI/CD Systems.

Hier verwenden wir den sam pipeline init --bootstrap Befehl, um eine CI/CD Bereitstellungspipeline für unsere Anwendung zu konfigurieren. Der AWS SAMCLI führt uns durch unsere Optionen und generiert die AWS Ressourcen und die Konfigurationsdatei, die mit unserem CI/CD System verwendet werden sollen.

```
[3] Reference application build resources
Enter the pipeline execution role ARN if you have previously created one, or we will create one for y
ou []:
Enter the CloudFormation execution role ARN if you have previously created one, or we will create one
 for you []:
Please enter the artifact bucket ARN for your Lambda function. If you do not have a bucket, we will c
reate one for you []:
Does your application contain any IMAGE type Lambda functions? [y/N]: n

[4] Summary
Below is the summary of the answers:
        1 - Account: 513423067560
        2 - Stage configuration name: dev
        3 - Region: us-west-2
        4 - Pipeline user: [to be created]
        5 - Pipeline execution role: [to be created]
        6 - CloudFormation execution role: [to be created]
        7 - Artifacts bucket: [to be created]
        8 - ECR image repository: [skipped]
Press enter to confirm the values above, or select an item to edit the value:

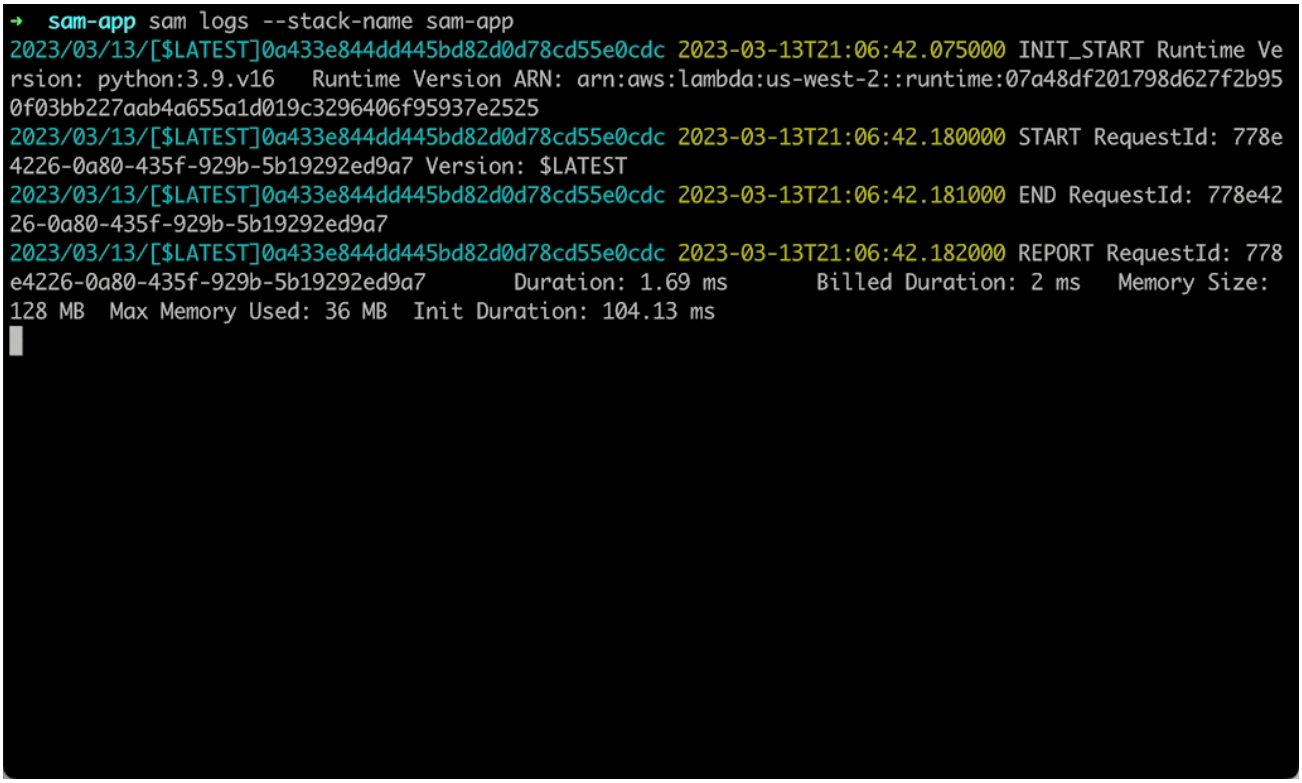
This will create the following required resources for the 'dev' configuration:
        - Pipeline IAM user
        - Pipeline execution role
        - CloudFormation execution role
        - Artifact bucket
Should we proceed with the creation? [y/N]:
```

Weitere Details finden Sie unter Bereitstellen mit CI/CD Systemen und Pipelines.

Überwachen Sie Ihre Anwendung in der Cloud und beheben Sie Fehler

Sehen Sie sich wichtige Informationen über Ihre bereitgestellten Ressourcen an, sammeln Sie Protokolle und nutzen Sie integrierte Überwachungstools wie AWS X-Ray.

Hier verwenden wir den sam list Befehl, um unsere bereitgestellten Ressourcen einzusehen. Wir rufen unseren API-Endpunkt ab und rufen ihn auf, was unsere Funktion auslöst. Dann sehen wir uns sam logs die Protokolle unserer Funktion an.

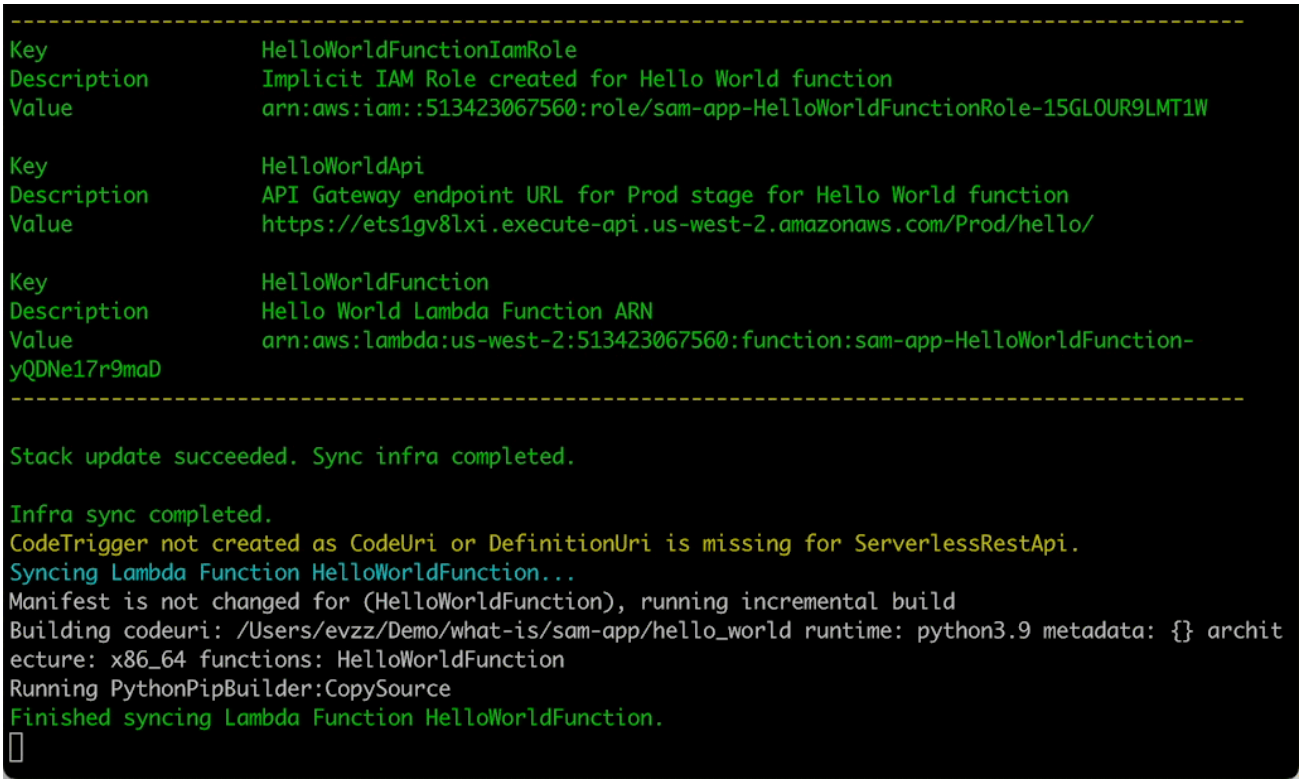


Weitere Details finden Sie unter Überwachen Sie Ihre Anwendung.

Synchronisieren Sie lokale Änderungen während der Entwicklung mit der Cloud

Synchronisieren Sie während der Entwicklung auf Ihrem lokalen Computer automatisch Änderungen mit der Cloud. Sehen Sie sich Ihre Änderungen schnell an und führen Sie Tests und Validierungen in der Cloud durch.

Hier verwenden wir den sam sync --watch Befehl, um nach lokalen Änderungen AWS SAMCLI Ausschau zu halten. Wir ändern unseren `HelloWorldFunction` Code und erkennen die Änderung AWS SAMCLI automatisch und stellen unsere Updates in der Cloud bereit.



Testen Sie unterstützte Ressourcen in der Cloud

Rufen Sie Ereignisse auf und leiten Sie sie an unterstützte Ressourcen in der Cloud weiter.

Hier verwenden wir den sam remote invoke Befehl, um eine bereitgestellte Lambda-Funktion in der Cloud zu testen. Wir rufen unsere Lambda-Funktion auf und erhalten ihre Protokolle und Antworten. Da unsere Lambda-Funktion so konfiguriert ist, dass sie Antworten streamt, AWS SAMCLI streamt sie ihre Antwort in Echtzeit zurück.



Weitere Informationen

Weitere Informationen AWS SAM finden Sie in den folgenden Ressourcen:

- Der komplette AWS SAM Workshop — Ein Workshop, in dem Sie viele der wichtigsten Funktionen des Workshops kennenlernen AWS SAM möchten.
- Sessions with SAM — Videoserie, die von unserem AWS Serverless Developer Advocate-Team zur Verwendung AWS SAM erstellt wurde.
- Serverless Land — Website, die die neuesten Informationen, Blogs, Videos, Code und Lernressourcen für AWS Serverless zusammenfasst.

Nächste Schritte

Wenn Sie es zum ersten Mal verwenden AWS SAM, finden Sie weitere Informationen unter. Erste Schritte mit AWS SAM

Serverlose Konzepte für AWS Serverless Application Model

Informieren Sie sich vor der Verwendung von () über grundlegende serverlose Konzepte. AWS Serverless Application Model AWS SAM

Serverlose Konzepte

Ereignisgesteuerte Architektur

Eine serverlose Anwendung besteht aus einzelnen AWS Diensten, z. B. AWS Lambda für Datenverarbeitung und Amazon DynamoDB für die Datenbankverwaltung, die jeweils eine spezielle Rolle erfüllen. Diese Dienste sind dann über eine ereignisgesteuerte Architektur lose miteinander integriert. Weitere Informationen zur ereignisgesteuerten Architektur finden Sie unter Was ist eine ereignisgesteuerte Architektur? .

Infrastruktur als Code (IaC)

Infrastructure as Code (IaC) ist eine Methode, Infrastruktur auf die gleiche Weise zu behandeln, wie Entwickler Code behandeln, wobei die gleiche Strenge bei der Entwicklung von Anwendungscode auf die Infrastrukturbereitstellung angewendet wird. Sie definieren Ihre Infrastruktur in einer Vorlagendatei, stellen sie bereit AWS und erstellen die Ressourcen AWS für Sie. Mit IaC definieren Sie im Code, was Sie bereitstellen AWS möchten. Einen Vergleich der für AWS verfügbaren IaC-Tools finden Sie unter. Infrastruktur als Code (IaC)

Serverlose Technologien

Mit AWS serverlosen Technologien können Sie Anwendungen erstellen und ausführen, ohne Ihre eigenen Server verwalten zu müssen. Die gesamte Serververwaltung erfolgt AWSüber und bietet viele Vorteile wie automatische Skalierung und integrierte Hochverfügbarkeit, sodass Sie Ihre Idee schnell in die Produktion umsetzen können. Durch den Einsatz serverloser Technologien können Sie sich auf den Kern Ihres Produkts konzentrieren, ohne sich um die Verwaltung und den Betrieb von Servern kümmern zu müssen. Weitere Informationen zu Serverless finden Sie im Folgenden:

- Serverlos an AWS
- Serverless Developer Guide — Bietet einen konzeptionellen Überblick über die serverlose Entwicklung in der Cloud. AWS

Eine grundlegende Einführung in die wichtigsten AWS serverlosen Dienste finden Sie unter Serverless 101: Understanding the serverless services at Serverless Land.

Serverlose Anwendung

Bei der Verwendung AWS SAM verwalten Sie zugehörige Ressourcen in einer Anwendung, die aus Ihrem AWS SAM Projekt und Ihrer Vorlage besteht. Alle Ressourcen in Ihrer Anwendung sind definiert oder in Ihrer AWS SAM Vorlage wird darauf verwiesen. Bei AWS SAM der Verarbeitung Ihrer Vorlage werden CloudFormation Ressourcen erstellt. In werden Ressourcen in einer

einzigen Einheit verwaltet CloudFormation, die als Stapel bezeichnet wird, und alle Ressourcen in einem Stapel werden durch die CloudFormation Vorlage des Stacks definiert.

Infrastruktur als Code (IaC)

Mit Infrastructure as Code (IaC) können Sie die Bereitstellung und Verwaltung Ihrer AWS Ressourcen, einschließlich serverloser Anwendungen, automatisieren. Mit IaC können Sie Ihre Infrastruktur mithilfe von Code definieren, wodurch es einfacher wird, Ihre Bereitstellungen zu versionieren, gemeinsam zu nutzen und zu replizieren. Dieser Ansatz hilft Ihnen:

- Beschleunigen Sie Ihren Entwicklungszyklus
- Vereinfachen Sie das Konfigurationsmanagement
- Verbessern Sie die Zuverlässigkeit und Konsistenz Ihrer Bereitstellungen

IaC-Tools für serverlose Anwendungen AWS

AWS bietet mehrere IaC-Tools, mit denen Sie Ihre Cloud-Ressourcen erstellen, bereitstellen und verwalten können. In diesem Abschnitt wird erklärt, wie es in dieses Ökosystem AWS SAM passt und mit anderen AWS IaC-Tools zusammenarbeitet.

AWS CloudFormation

Mithilfe von CloudFormationYAML- oder JSON-Vorlagen können Sie Ihre gesamte AWS Infrastruktur modellieren und bereitstellen. CloudFormation erledigt das Erstellen, Aktualisieren und Löschen von Ressourcen automatisch. Wenn Sie AWS SAM Anwendungen bereitstellen, CloudFormation verarbeitet es die transformierten Vorlagen, um Ihre Ressourcen zu erstellen und zu verwalten.

AWS Serverless Application Model (AWS SAM)

AWS SAM hilft Ihnen bei der Erstellung serverloser Anwendungen mit vereinfachter Syntax zur Definition serverloser Ressourcen. Sie können AWS SAM Vorlagen verwenden, um Lambda-Funktionen APIs, Datenbanken und Ereignisquellen mithilfe einer präzisen YAML-Syntax bereitzustellen. AWS SAM wandelt diese Vorlagen während der Bereitstellung in CloudFormation Vorlagen um.

> **Note**
>
> Das AWS SAM Unternehmen ist zwar auf serverlose Anwendungen spezialisiert, Sie können jedoch beliebige CloudFormation Ressourcentypen in Ihren AWS SAM Vorlagen verwenden. Dies gibt Ihnen die Flexibilität, Ressourcen, die nicht serverlos sind, bei Bedarf einzubeziehen.

AWS Cloud Development Kit (AWS CDK)

Mit AWS CDKkönnen Sie Ihre serverlose Infrastruktur mit vertrauten Programmiersprachen wie Python TypeScript, Java, C#/.Net oder Go definieren. Sie können Programmierkonstrukte wie Schleifen und Bedingungen verwenden, um Ihre Infrastruktur zu definieren und Vorlagen für die Bereitstellung zu generieren. AWS CDK CloudFormation Sie können das verwenden AWS SAMCLI, um Anwendungen, die mit erstellt wurden, lokal zu testen und zu debuggen. AWS CDK Weitere Informationen finden Sie unter Lokales Testen von CDK-Anwendungen.

Vergleich der IaC-Tools für serverlose Anwendungen

Bei der Auswahl eines IaC-Tools für Ihre serverlosen Anwendungen sollten Sie die Präferenzen Ihres Teams, die Projektanforderungen und die bestehenden Workflows berücksichtigen. In der folgenden Tabelle werden die wichtigsten Merkmale von AWS IaC-Tools für die serverlose Entwicklung verglichen:

Tool	Primäre Verwendung	Am besten geeignet für	Funktioniert mit AWS SAM	Wann soll ich wählen
CloudFormation	Verwaltung einer komplexen AWS Infrastruktur	Anwendungen, die eine detaillierte Kontrolle der AWS Ressourcen erfordern	AWS SAM Vorlagen werden während der Bereitstellung in CloudFormation Vorlagen umgewandelt	Für eine detaillierte Kontrolle über Ressourcen, die nicht serverlos sind

Tool	Primäre Verwendung	Am besten geeignet für	Funktioniert mit AWS SAM	Wann soll ich wählen
AWS SAM	Serverlose Anwendung sentwicklung	Teams, die serverlose Anwendungen mit Lambda erstellen	Native Funktiona lität	Wenn Sie sich hauptsächlich auf serverlose Architekturen mit Lambda-Funktionen, API Gateway und anderen serverlosen APIs Ressourcen konzentrieren
AWS CDK	Definition der Infrastruktur mithilfe von Programmi ersprachen	Teams, die typisierte Sprachen und einen Code-First-Ansatz bevorzuge n	Generieren Sie AWS SAM Vorlagen und verwenden Sie sie zum Testen AWS SAMCLI	Wenn Sie eine programma tische Infrastru kturdefinition oder eine komplexe Ressource nkonfigurationslog ik benötigen

> **Note**
>
> Dieser Leitfaden konzentriert sich zwar auf AWS native IaC-Tools, Terraform ist jedoch eine weitere beliebte IaC-Lösung, mit der serverlose Anwendungen definiert werden können. Das AWS SAMCLI unterstützt das lokale Testen von Lambda-Funktionen, die in Terraform definiert sind. Weitere Informationen finden Sie unter AWS SAM CLI Terraform Support.

Weitere Informationen

- Weitere Informationen zu den DevOps Methoden finden Sie AWS unter Einführung in DevOps AWS

- Informationen zur Verwendung von Lambda mit verschiedenen IaC-Tools finden Sie unter Verwenden von Lambda mit Infrastruktur als Code (IaC)

Erste Schritte mit AWS SAM

Lesen und vervollständigen Sie zunächst die Themen in diesem Abschnitt. AWS SAM AWS SAM Voraussetzungenenthält detaillierte Anweisungen zum Einrichten eines AWS Kontos, zum Erstellen von IAM-Benutzern, zum Erstellen von Schlüsselzugriffen sowie zur Installation und Konfiguration von. AWS SAMCLI Nachdem Sie die Voraussetzungen erfüllt haben, können Sie Ihren Entwicklungsansatz wählen:

- Installieren Sie das AWS SAMCLI, was Sie auf Linux-, Windows- und MacOS-Betriebssystemen tun können.
- Erste Schritte mit AWS SAM mithilfe des VS Code Toolkit für ein visuelles Entwicklungserlebnis mit integrierten Debugging- und Testfunktionen.

Nach Abschluss der Installation können Sie optional das AWS SAM Hello World-Tutorial durchgehen. Im Anschluss an dieses Tutorial werden Sie durch den Prozess der Erstellung einer grundlegenden serverlosen Anwendung mit AWS SAM geführt. Nach Abschluss des Tutorials können Sie sich mit den unter beschriebenen Konzepten vertraut machen. Wie benutzt man AWS Serverless Application Model (AWS SAM)

Themen

- AWS SAM Voraussetzungen
- Installieren Sie das AWS SAMCLI
- Tutorial: Stellen Sie eine Hello World-Anwendung bereit mit AWS SAM
- Lambda-Funktionen in Anwendungen umwandeln AWS SAM

AWS SAM Voraussetzungen

Erfüllen Sie die folgenden Voraussetzungen, bevor Sie die AWS Serverless Application Model Befehlszeilenschnittstelle (AWS SAMCLI) installieren und verwenden.

Um das verwenden zu können AWS SAMCLI, benötigen Sie Folgendes:

- Ein AWS Konto, AWS Identity and Access Management (IAM-) Anmeldeinformationen und ein IAM-Zugriffsschlüsselpaar.
- Das AWS Command Line Interface (AWS CLI) zum Konfigurieren AWS der Anmeldeinformationen.

Themen

- Melde dich an für AWS-Konto
- Schritt 1: Installieren Sie AWS CLI
- Schritt 2: Verwenden Sie den AWS CLI zur Konfiguration AWS Anmeldedaten
- Schritt 3: (Optional) Installieren AWS Toolkit für VS Code
- Nächste Schritte

Melde dich an für AWS-Konto

Um damit zu beginnen AWS, benötigen Sie eine AWS-Konto. Informationen zum Erstellen einer AWS-Konto finden Sie AWS-Konto im AWS -Kontenverwaltung Referenzhandbuch unter Erste Schritte mit einer.

Schritt 1: Installieren Sie AWS CLI

Das AWS CLI ist ein Open-Source-Tool, mit dem Sie AWS-Services mithilfe von Befehlen in Ihrer Befehlszeilen-Shell interagieren können. Das AWS SAMCLI erfordert vier Aktivitäten wie AWS CLI das Konfigurieren von Anmeldeinformationen. Weitere Informationen über das AWS CLI finden Sie unter Was ist das AWS Command Line Interface? im AWS Command Line Interface Benutzerhandbuch.

Informationen zur AWS CLI Installation von finden Sie im AWS Command Line Interface Benutzerhandbuch unter Installieren oder Aktualisieren AWS CLI der neuesten Version von.

Schritt 2: Verwenden Sie den AWS CLI zur Konfiguration AWS Anmeldedaten

So konfigurieren Sie Anmeldeinformationen mit IAM Identity Center

- Informationen zum Konfigurieren von Anmeldeinformationen mit IAM Identity Center finden Sie unter Konfigurieren Sie Ihr Profil mit dem Assistenten zum AWS Konfigurieren von SSO.

So konfigurieren Sie Anmeldeinformationen mit dem AWS CLI

1. Führen Sie den `aws configure` Befehl von der Befehlszeile aus.
2. Konfigurieren Sie Folgendes. Wählen Sie jeden Link aus, um mehr zu erfahren:

a. Zugriffsschlüssel-ID

b. Geheimer Zugangsschlüssel

c. AWS-Region

d. Ausgabeformat

Das folgende Beispiel zeigt Beispielwerte.

```
$ aws configure
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Default region name [None]: us-west-2
Default output format [None]: json
```

Das AWS CLI speichert diese Informationen in einem Profil (einer Sammlung von Einstellungen), das `default` in den `credentials` und `config` -Dateien benannt ist. Diese Dateien befinden sich in der `.aws` Datei in Ihrem Home-Verzeichnis. Standardmäßig werden die Informationen in diesem Profil verwendet, wenn Sie einen AWS CLI Befehl ausführen, der nicht explizit ein zu verwendendes Profil angibt. Weitere Informationen zu der `credentials` Datei finden Sie im AWS Command Line Interface Benutzerhandbuch unter Konfiguration und Einstellungen der Anmeldeinformationsdatei.

Weitere Informationen zur Konfiguration von Anmeldeinformationen, z. B. zur Verwendung einer vorhandenen Konfigurationsdatei und einer Anmeldeinformationsdatei, finden Sie im AWS Command Line Interface Benutzerhandbuch unter Schnelleinrichtung.

Schritt 3: (Optional) Installieren AWS Toolkit für VS Code

Für Entwickler, die eine integrierte Entwicklungsumgebung bevorzugen, bietet das AWS Toolkit für VS Code leistungsstarke Funktionen für die serverlose Entwicklung, darunter visuelles Debugging, CodeLens Integration und optimierte Bereitstellungsworkflows.

Voraussetzungen für die VS Code-Entwicklung

- Visual Studio Code (Version 1.73.0 oder höher) ist auf Ihrem System installiert
- Erweiterung zur YAML-Sprachunterstützung für VS Code

Um das zu installieren AWS Toolkit für VS Code

1. Öffnen Sie Visual Studio Code
2. Öffnen Sie die Erweiterungsansicht (Strg+Shift+X oder Cmd+Shift+X)
3. Suchen AWS Sie nach „Toolkit“
4. Installieren Sie die Erweiterung „AWS Toolkit“ von Amazon Web Services
5. Installieren Sie die „YAML“ -Erweiterung von Red Hat (für CodeLens SAM-Vorlagenfunktionen erforderlich)

Vorteile der Verwendung von VS Code mit SAM AWS

- Visuelles Debugging: Setzen Sie Breakpoints und führen Sie Ihre Lambda-Funktionen lokal Schritt für Schritt durch
- CodeLens Integration: Erstellen, Bereitstellen und Aufrufen von Funktionen direkt aus Ihrer SAM-Vorlage
- Integriertes Terminal: Greifen Sie auf AWS AWS CLI SAM-Befehle zu, ohne Ihren Editor zu verlassen
- Vorlagenvalidierung: Real-time Validierung und IntelliSense für SAM-Vorlagen

Informationen zur Konfiguration Ihrer AWS Anmeldeinformationen in VS Code finden Sie unter Einrichten von Anmeldeinformationen im AWS Toolkit für VS Code-Benutzerhandbuch.

Nächste Schritte

- Sie sind jetzt bereit, das zu installieren AWS SAMCLI und zu verwenden AWS SAM. Informationen zur Installation von AWS SAMCLI finden Sie unterInstallieren Sie das AWS SAMCLI.
- Informationen zum Einrichten von Visual Studio Code für die serverlose Entwicklung finden Sie unter AWS Toolkit für VS Code einrichten.

Installieren Sie das AWS SAMCLI

Installieren Sie die neueste Version der AWS Serverless Application Model Befehlszeilenschnittstelle (AWS SAMCLI) auf unterstützten Betriebssystemen, indem Sie die Anweisungen unter befolgenSchritt 1: Installieren Sie AWS CLI.

Informationen zur Verwaltung einer aktuell installierten Version von AWS SAMCLI, einschließlich der Aktualisierung, Deinstallation oder Verwaltung nächtlicher Builds, finden Sie unterAWS SAMCLIVersionen verwalten.

> **Installieren Sie das zum ersten Mal? AWS SAM CLI**
>
> Erfüllen Sie alle Voraussetzungen im vorherigen Abschnitt, bevor Sie fortfahren. Dies umfasst:
>
> 1. Eröffnen Sie ein AWS Konto.
> 2. Einrichtung eines sicheren Zugriffs auf AWS.
> 3. Erstellen einer Zugriffsschlüssel-ID und eines geheimen Zugriffsschlüssels.
> 4. Installation der AWS CLI.
> 5. AWS Anmeldeinformationen konfigurieren.

Themen

- Installation des AWS SAMCLI
- Behebung von Installationsfehlern
- Nächste Schritte
- Optional: Überprüfen Sie die Integrität des AWS SAM CLI installer

Installation des AWS SAMCLI

> **Note**
>
> Ab September 2023 AWS wird das AWS verwaltete Homebrew Installationsprogramm für AWS SAMCLI (`aws/tap/aws-sam-cli`) nicht mehr verwaltet. Wenn Sie Homebrew zur Installation und Verwaltung von verwenden AWS SAMCLI, finden Sie die folgenden Optionen:
>
> - Um die Nutzung fortzusetzenHomebrew, können Sie das von der Community verwaltete Installationsprogramm verwenden. Weitere Informationen finden Sie unter Verwaltung des AWS SAMCLI mit Homebrew.
> - Wir empfehlen, eine der Installationsmethoden von Erstanbietern zu verwenden, die auf dieser Seite dokumentiert sind. Bevor Sie eine dieser Methoden verwenden, finden Sie weitere Informationen unterWechseln von Homebrew.

- Weitere Informationen finden Sie in der Release-Version: 1.121.0.

Folgen Sie den Anweisungen für Ihr Betriebssystem AWS SAMCLI, um das zu installieren.

Linux

x86_64 - command line installer

1. Laden Sie die AWS SAMCLIZIP-Datei in ein Verzeichnis Ihrer Wahl herunter.
2. (Optional) Sie können die Integrität des Installationsprogramms vor der Installation überprüfen. Detaillierte Anweisungen finden Sie unter Optional: Überprüfen Sie die Integrität des AWS SAM CLI installer.
3. Entpacken Sie die Installationsdateien in ein Verzeichnis Ihrer Wahl. Das Folgende ist ein Beispiel, das das `sam-installation` Unterverzeichnis verwendet.

   > **Note**
   >
   > Wenn Ihr Betriebssystem nicht über den integrierten unzip-Befehl verfügt, verwenden Sie ein Äquivalent.

   ```
   $ unzip aws-sam-cli-linux-x86_64.zip -d sam-installation
   ```

4. Installieren Sie das, AWS SAMCLI indem Sie die `install` ausführbare Datei ausführen. Diese ausführbare Datei befindet sich in dem Verzeichnis, das im vorherigen Schritt verwendet wurde. Im Folgenden finden Sie ein Beispiel, bei dem das `sam-installation` Unterverzeichnis verwendet wird:

   ```
   $ sudo ./sam-installation/install
   ```

5. Überprüfen Sie die Installation.

   ```
   $ sam --version
   ```

   Um eine erfolgreiche Installation zu bestätigen, sollten Sie eine Ausgabe sehen, die den folgenden Text in Klammern durch die neueste verfügbare Version ersetzt:

```
SAM CLI, <latest version>
```

arm64 - command line installer

1. Laden Sie die AWS SAMCLIZIP-Datei in ein Verzeichnis Ihrer Wahl herunter.
2. (Optional) Sie können die Integrität des Installationsprogramms vor der Installation überprüfen. Detaillierte Anweisungen finden Sie unter Optional: Überprüfen Sie die Integrität des AWS SAM CLI installer.
3. Entpacken Sie die Installationsdateien in ein Verzeichnis Ihrer Wahl. Das Folgende ist ein Beispiel, das das `sam-installation` Unterverzeichnis verwendet.

   > **Note**
   >
   > Wenn Ihr Betriebssystem nicht über den integrierten unzip-Befehl verfügt, verwenden Sie ein Äquivalent.

   ```
   $ unzip aws-sam-cli-linux-arm64.zip -d sam-installation
   ```

4. Installieren Sie das, AWS SAMCLI indem Sie die `install` ausführbare Datei ausführen. Diese ausführbare Datei befindet sich in dem Verzeichnis, das im vorherigen Schritt verwendet wurde. Im Folgenden finden Sie ein Beispiel, bei dem das `sam-installation` Unterverzeichnis verwendet wird:

   ```
   $ sudo ./sam-installation/install
   ```

5. Überprüfen Sie die Installation.

   ```
   $ sam --version
   ```

   Um eine erfolgreiche Installation zu bestätigen, sollten Sie eine Ausgabe wie die folgende sehen, die jedoch den Text in Klammern durch die neueste SAM-CLI-Version ersetzt:

   ```
   SAM CLI, <latest version>
   ```

macOS

Schritte zur Installation

Verwenden Sie das Paketinstallationsprogramm, um das zu installieren AWS SAMCLI. Darüber hinaus bietet das Paketinstallationsprogramm zwei Installationsmethoden, aus denen Sie wählen können: GUI und Befehlszeile. Sie können die Installation für alle Benutzer oder nur für Ihren aktuellen Benutzer durchführen. Für die Installation für alle Benutzer ist eine Superuser-Autorisierung erforderlich.

> **Note**
>
> AWS SAM CLI unterstützt keine macOS-Versionen, die älter als macOS 13.x sind. Führen Sie ein Upgrade auf eine unterstützte Version von macOS durch oder installieren Sie AWS SAM CLI mitHomebrew.

GUI - All users

Um das Paketinstallationsprogramm herunterzuladen und das zu installieren AWS SAMCLI

> **Note**
>
> Wenn Sie das Programm zuvor AWS SAMCLI über Homebrew oder installiert habenpip, müssen Sie es zuerst deinstallieren. Detaillierte Anweisungen finden Sie unter Deinstallation des AWS SAMCLI.

1. Laden Sie das macOS pkg in ein Verzeichnis Ihrer Wahl herunter:
   - Wählen Sie für Macs mit Intel-Prozessoren x86_64 — -x86_64.pkg aws-sam-cli-macos
   - Wählen Sie für Macs, auf denen Apple Silicon ausgeführt wird, arm64 — -arm64.pkg aws-sam-cli-macos

> **Note**
>
> Sie haben die Möglichkeit, die Integrität des Installationsprogramms vor der Installation zu überprüfen. Detaillierte Anweisungen finden Sie unter Optional: Überprüfen Sie die Integrität des AWS SAM CLI installer.

2. Führen Sie die heruntergeladene Datei aus und folgen Sie den Anweisungen auf dem Bildschirm, um mit den Schritten Einführung, ReadMe und Lizenz fortzufahren.
3. Wählen Sie unter Zielauswahl die Option Für alle Benutzer dieses Computers installieren aus.
4. Wählen Sie unter Installationstyp aus, wo das installiert AWS SAMCLI werden soll, und klicken Sie auf Installieren. Der empfohlene Standardspeicherort ist`/usr/local/aws-sam-cli`.

> **Note**
>
> Um das AWS SAMCLI mit dem sam Befehl aufzurufen, erstellt das Installationsprogramm automatisch einen Symlink zwischen `/usr/local/bin/sam` und einem `/usr/local/aws-sam-cli/sam` oder dem von Ihnen ausgewählten Installationsordner.

5. Die Meldung AWS SAMCLI wird installiert und Die Installation war erfolgreich wird angezeigt. Drücken Sie auf Schließen.

Um eine erfolgreiche Installation zu überprüfen

- Stellen Sie sicher, dass der ordnungsgemäß installiert AWS SAMCLI wurde und ob Ihr Symlink konfiguriert ist, indem Sie Folgendes ausführen:

```
$ which sam
/usr/local/bin/sam
$ sam --version
SAM CLI, <latest version>
```

GUI - Current user

Um das herunterzuladen und zu installieren AWS SAMCLI

> **Note**
>
> Wenn Sie das zuvor AWS SAMCLI über Homebrew oder installiert habenpip, müssen Sie es zuerst deinstallieren. Detaillierte Anweisungen finden Sie unter Deinstallation des AWS SAMCLI.

1. Laden Sie das macOS `pkg` in ein Verzeichnis Ihrer Wahl herunter:

   - Wählen Sie für Macs mit Intel-Prozessoren x86_64 — -x86_64.pkg aws-sam-cli-macos
   - Wählen Sie für Macs, auf denen Apple Silicon ausgeführt wird, arm64 — -arm64.pkg aws-sam-cli-macos

   > **Note**
   >
   > Sie haben die Möglichkeit, die Integrität des Installationsprogramms vor der Installation zu überprüfen. Detaillierte Anweisungen finden Sie unter Optional: Überprüfen Sie die Integrität des AWS SAM CLI installer.

2. Führen Sie die heruntergeladene Datei aus und folgen Sie den Anweisungen auf dem Bildschirm, um mit den Schritten Einführung, ReadMe und Lizenz fortzufahren.
3. Wählen Sie unter Destination Select die Option Nur für mich installieren aus. Wenn Sie diese Option nicht sehen, fahren Sie mit dem nächsten Schritt fort.
4. Gehen Sie unter Installationstyp wie folgt vor:

   1. Wählen Sie aus, wo das installiert AWS SAMCLI werden soll. Der Standardspeicherort ist `/usr/local/aws-sam-cli`. Wählen Sie einen Standort aus, für den Sie Schreibberechtigungen haben. Um den Installationsort zu ändern, wählen Sie Lokal und dann Ihren Standort aus. Drücken Sie auf Weiter, wenn Sie fertig sind.
   2. Falls Sie im vorherigen Schritt nicht die Option Nur für mich installieren hatten, wählen Sie „Installationsort ändern“ > „Nur für mich installieren“ und klicken Sie auf „Weiter“.
   3. Drücken Sie auf Installieren.

5. Die Meldung AWS SAMCLI wird installiert und Die Installation war erfolgreich wird angezeigt. Drücken Sie auf Schließen.

Um einen Symlink zu erstellen

- Um den AWS SAMCLI mit dem sam Befehl aufzurufen, müssen Sie manuell einen Symlink zwischen dem AWS SAMCLI Programm und Ihrem erstellen. $PATH Erstellen Sie Ihren Symlink, indem Sie den folgenden Befehl ändern und ausführen:

```
$ sudo ln -s /path-to/aws-sam-cli/sam /path-to-symlink-directory/sam
```

  - *sudo*— Wenn Ihr Benutzer Schreibrechte hat$PATH, sudo ist dies nicht erforderlich. Andernfalls ist sudo erforderlich.
  - *path-to*— Pfad zu dem Ort, an dem Sie das AWS SAMCLI Programm installiert haben. Beispiel, `/Users/myUser/Desktop`.
  - *path-to-symlink-directory*— Ihre $PATH Umgebungsvariable. Der Standardspeicherort ist `/usr/local/bin`.

Um eine erfolgreiche Installation zu überprüfen

- Stellen Sie sicher, dass der ordnungsgemäß installiert AWS SAMCLI wurde und ob Ihr Symlink konfiguriert ist, indem Sie Folgendes ausführen:

```
$ which sam
/usr/local/bin/sam
$ sam --version
SAM CLI, <latest version>
```

Command line - All users

Um das herunterzuladen und zu installieren AWS SAMCLI

> **Note**
>
> Wenn Sie das zuvor AWS SAMCLI über Homebrew oder installiert habenpip, müssen Sie es zuerst deinstallieren. Detaillierte Anweisungen finden Sie unter Deinstallation des AWS SAMCLI.

1. Laden Sie das macOS pkg in ein Verzeichnis Ihrer Wahl herunter:

   - Wählen Sie für Macs mit Intel-Prozessoren x86_64 — -x86_64.pkg aws-sam-cli-macos
   - Wählen Sie für Macs, auf denen Apple Silicon ausgeführt wird, arm64 — -arm64.pkg aws-sam-cli-macos

   > **Note**
   >
   > Sie haben die Möglichkeit, die Integrität des Installationsprogramms vor der Installation zu überprüfen. Detaillierte Anweisungen finden Sie unter Optional: Überprüfen Sie die Integrität des AWS SAM CLI installer.

2. Ändern Sie das Installationsskript und führen Sie es aus:

```
$ sudo installer -pkg path-to-pkg-installer/name-of-pkg-installer -target /
installer: Package name is AWS SAM CLI
installer: Upgrading at base path /
installer: The upgrade was successful.
```

> **Note**
>
> Um das AWS SAMCLI mit dem sam Befehl aufzurufen, erstellt das Installationsprogramm automatisch einen Symlink zwischen und`/usr/local/bin/sam`. `/usr/local/aws-sam-cli/sam`

Um eine erfolgreiche Installation zu überprüfen

- Stellen Sie sicher, dass der ordnungsgemäß installiert AWS SAMCLI wurde und ob Ihr Symlink konfiguriert ist, indem Sie Folgendes ausführen:

```
$ which sam
/usr/local/bin/sam
$ sam --version
SAM CLI, <latest version>
```

Command line - Current user

Um das herunterzuladen und zu installieren AWS SAMCLI

> **Note**
>
> Wenn Sie das zuvor AWS SAMCLI über Homebrew oder installiert habenpip, müssen Sie es zuerst deinstallieren. Detaillierte Anweisungen finden Sie unter Deinstallation des AWS SAMCLI.

1. Laden Sie das macOS `pkg` in ein Verzeichnis Ihrer Wahl herunter:

   - Wählen Sie für Macs mit Intel-Prozessoren x86_64 — -x86_64.pkg aws-sam-cli-macos
   - Wählen Sie für Macs, auf denen Apple Silicon ausgeführt wird, arm64 — -arm64.pkg aws-sam-cli-macos

   > **Note**
   >
   > Sie haben die Möglichkeit, die Integrität des Installationsprogramms vor der Installation zu überprüfen. Detaillierte Anweisungen finden Sie unter Optional: Überprüfen Sie die Integrität des AWS SAM CLI installer.

2. Ermitteln Sie ein Installationsverzeichnis, für das Sie Schreibberechtigungen haben. Erstellen Sie dann mithilfe der Vorlage eine `xml` Datei und ändern Sie sie so, dass sie Ihrem Installationsverzeichnis entspricht. Das Verzeichnis muss bereits existieren.

Wenn Sie beispielsweise durch *path-to-my-directory* ersetzen`/Users/myUser/Desktop`, wird der `aws-sam-cli` Programmordner dort installiert.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/
PropertyList-1.0.dtd">
<plist version="1.0">
  <array>
    <dict>
      <key>choiceAttribute</key>
      <string>customLocation</string>
      <key>attributeSetting</key>
      <string>path-to-my-directory</string>
      <key>choiceIdentifier</key>
      <string>default</string>
    </dict>
  </array>
</plist>
```

3. Speichern Sie die `xml` Datei und überprüfen Sie, ob sie gültig ist, indem Sie Folgendes ausführen:

```
$ installer -pkg path-to-pkg-installer \
-target CurrentUserHomeDirectory \
-showChoicesAfterApplyingChangesXML path-to-your-xml-file
```

In der Ausgabe sollten die Einstellungen angezeigt werden, die auf das AWS SAMCLI Programm angewendet werden.

4. Führen Sie den folgenden Befehl aus, um Folgendes zu installieren AWS SAMCLI:

```
$ installer -pkg path-to-pkg-installer \
-target CurrentUserHomeDirectory \
-applyChoiceChangesXML path-to-your-xml-file

# Example output
installer: Package name is AWS SAM CLI
installer: choices changes file 'path-to-your-xml-file' applied
installer: Upgrading at base path base-path-of-xml-file
installer: The upgrade was successful.
```

Um einen Symlink zu erstellen

- Um den AWS SAMCLI mit dem sam Befehl aufzurufen, müssen Sie manuell einen Symlink zwischen dem AWS SAMCLI Programm und Ihrem erstellen. $PATH Erstellen Sie Ihren Symlink, indem Sie den folgenden Befehl ändern und ausführen:

```
$ sudo ln -s /path-to/aws-sam-cli/sam /path-to-symlink-directory/sam
```

  - *sudo*— Wenn Ihr Benutzer Schreibrechte hat$PATH, sudo ist dies nicht erforderlich. Andernfalls ist sudo erforderlich.
  - *path-to*— Pfad zu dem Ort, an dem Sie das AWS SAMCLI Programm installiert haben. Beispiel, `/Users/myUser/Desktop`.
  - *path-to-symlink-directory*— Ihre $PATH Umgebungsvariable. Der Standardspeicherort ist `/usr/local/bin`.

Um eine erfolgreiche Installation zu überprüfen

- Stellen Sie sicher, dass der ordnungsgemäß installiert AWS SAMCLI wurde und ob Ihr Symlink konfiguriert ist, indem Sie Folgendes ausführen:

```
$ which sam
/usr/local/bin/sam
$ sam --version
SAM CLI, <latest version>
```

Windows

Windows Installer (MSI) -Dateien sind die Paket-Installationsdateien für das Windows-Betriebssystem.

Gehen Sie wie folgt vor, um das AWS SAMCLI mithilfe der MSI-Datei zu installieren.

1. Laden Sie die AWS SAMCLI 64-Bit-Version herunter.
2. (Optional) Sie können die Integrität des Installationsprogramms vor der Installation überprüfen. Detaillierte Anweisungen finden Sie unter Optional: Überprüfen Sie die Integrität des AWS SAM CLI installer.
3. Überprüfen Sie die Installation.

Überprüfen Sie die Installation nach Abschluss der Installation, indem Sie eine neue Befehlszeile oder eine neue PowerShell Eingabeaufforderung öffnen. Sie sollten in der Lage sein, `sam` von der Befehlszeile aus aufzurufen.

```
sam --version
```

Nach erfolgreicher Installation von sollten Sie eine Ausgabe wie die folgende sehen: AWS SAMCLI

```
SAM CLI, <latest version>
```

4. Aktivieren Sie lange Pfade (nur Windows 10 und neuer).

> **Important**
>
> Sie interagieren AWS SAMCLI möglicherweise mit Dateipfaden, die die maximale Pfadbeschränkung von Windows überschreiten. Dies kann `sam init` aufgrund von Windows 10-Einschränkungen zu Fehlern bei der Ausführung führen. MAX_PATH Um dieses Problem zu beheben, muss das neue Verhalten für lange Pfade konfiguriert werden.

Informationen zum Aktivieren langer Pfade finden Sie unter Aktivieren langer Pfade in Windows 10, Version 1607 und höher in der Dokumentation zur Entwicklung von Microsoft Windows-Apps.

5. Installieren Sie Git.

Um Beispielanwendungen mit dem `sam init` Befehl herunterzuladen, müssen Sie auch Git installieren. Eine Anleitung dazu findest du unter Git installieren.

Behebung von Installationsfehlern

Linux

Docker-Fehler: „Es kann keine Verbindung zum Docker-Daemon hergestellt werden. Läuft der Docker-Daemon auf diesem Host?“

In einigen Fällen müssen Sie Ihre Instance möglicherweise neu starten`ec2-user`, um Berechtigungen für den Zugriff auf den Docker-Daemon zu erteilen. Wenn Sie diesen Fehler erhalten, versuchen Sie, Ihre Instance neu zu starten.

Shell-Fehler: „Befehl nicht gefunden“

Wenn Sie diesen Fehler erhalten, kann Ihre Shell die AWS SAMCLI ausführbare Datei im Pfad nicht finden. Überprüfen Sie den Speicherort des Verzeichnisses, in dem Sie die AWS SAMCLI ausführbare Datei installiert haben, und stellen Sie dann sicher, dass sich das Verzeichnis in Ihrem Pfad befindet.

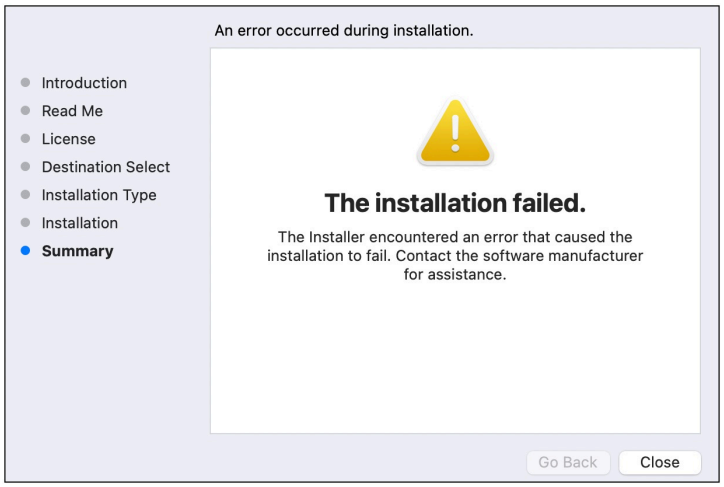
AWS SAMCLIFehler: „/lib64/libc.so.6: Version `GLIBC_2.14' nicht gefunden (erforderlich für/.so.1)“ usr/local/aws-sam-cli/dist/libz

Wenn Sie diesen Fehler erhalten, verwenden Sie eine nicht unterstützte Version von Linux und die integrierte Glibc-Version ist veraltet. Versuchen Sie es mit einer der folgenden Methoden:

- Aktualisieren Sie Ihren Linux-Host auf die 64-Bit-Version einer aktuellen Distribution von CentOS, Fedora, Ubuntu oder Amazon Linux 2.
- Folgen Sie den Anweisungen für. Installieren Sie das AWS SAMCLI

macOS

Die Installation ist fehlgeschlagen



Wenn Sie das AWS SAMCLI für Ihren Benutzer installieren und ein Installationsverzeichnis ausgewählt haben, für das Sie keine Schreibberechtigungen haben, kann dieser Fehler auftreten. Versuchen Sie es mit einer der folgenden Methoden:

1. Wählen Sie ein anderes Installationsverzeichnis aus, für das Sie Schreibberechtigungen haben.
2. Löschen Sie das Installationsprogramm. Laden Sie es dann herunter und führen Sie es erneut aus.

Nächste Schritte

Weitere Informationen über die AWS SAMCLI und wie Sie mit der Erstellung eigener serverloser Anwendungen beginnen können, finden Sie im Folgenden:

- Tutorial: Stellen Sie eine Hello World-Anwendung bereit mit AWS SAM— Step-by-step Anweisungen zum Herunterladen, Erstellen und Bereitstellen einer grundlegenden serverlosen Anwendung.
- Der komplette AWS SAM Workshop — Ein Workshop, in dem Sie viele der wichtigsten Funktionen des Workshops kennenlernen möchten AWS SAM .
- AWS SAM Beispielanwendungen und Muster — Beispielanwendungen und Muster von Community-Autoren, mit denen Sie weiter experimentieren können.

Optional: Überprüfen Sie die Integrität des AWS SAM CLI installer

Wenn Sie die AWS Serverless Application Model Befehlszeilenschnittstelle (AWS SAMCLI) mit einem Paketinstallationsprogramm installieren, können Sie deren Integrität vor der Installation überprüfen. Dies ist ein optionaler, aber dringend empfohlener Schritt.

Die beiden Bestätigungsoptionen, die Ihnen zur Verfügung stehen, sind:

- Überprüfen Sie die Signaturdatei des Paketinstallers.
- Überprüfen Sie den Hashwert des Paketinstallers.

Sofern für Ihre Plattform verfügbar, empfehlen wir, die Option für die Signaturdatei zu überprüfen. Diese Option bietet eine zusätzliche Sicherheitsebene, da die Schlüsselwerte hier veröffentlicht und getrennt von unserem GitHub Repository verwaltet werden.

Topics

- Überprüfen Sie die Signaturdatei des Installationsprogramms
- Überprüfen Sie den Hashwert

Überprüfen Sie die Signaturdatei des Installationsprogramms

Linux

arm64 — Befehlszeilen-Installationsprogramm

AWS SAM benutzt GnuPG, um das ZIP-Installationsprogramm zu signieren AWS SAMCLI. Die Überprüfung erfolgt in den folgenden Schritten:

1. Verwenden Sie den öffentlichen Primärschlüssel, um den öffentlichen Schlüssel des Unterzeichners zu überprüfen.
2. Verwenden Sie den öffentlichen Schlüssel des Unterzeichners, um das Installationsprogramm des AWS SAMCLI Pakets zu verifizieren.

Um die Integrität des öffentlichen Schlüssels des Unterzeichners zu überprüfen

1. Kopieren Sie den öffentlichen Primärschlüssel und speichern Sie ihn als `.txt` Datei auf Ihrem lokalen Computer. Beispiel, *`primary-public-key.txt`*.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2.0.22 (GNU/Linux)

mQINBGRuSzMBEADsqiwOy78w7F4+sshaMFRIwRGNRm94p5Qey2KMZBxekFtoryVD
D9jEOnvupx4tvhfBHz5EcUHCEOdl4MTqdBy6vVAshozgxVb9RE8JpECn5lw7XC69
4Y7Gy1TKKQMEWtDXElkGxIFdUWvWjSnPlzfnoXwQYGeE93CUS3h5dImP22Yk1Ct6
eGGhlcbg1X4L8EpFMj7GvcsU8f7ziVI/PyC1Xwy39Q8/I67ip5eU5ddxO/xHqrbL
YC7+8pJPbRMej2twT2LrcpWWYAbprMtRoa6WfE0/thoo3xhHpIMHdPfAA86ZNGIN
kRLjGUg7jnPTRW4Oin3pCc8nT4Tfc1QERkHm641gTC/jUvpmQsM6h/FUVP2i5iE/
JHpJcMuL2Mg6zDo3x+3gTCf+Wqz3rZzxB+wQT3yryZs6efcQy7nROiRxYBxCSXX0
2cNYzsYLb/bYaW8yqWIHD5IqKhw269gp2E5Khs60zgS3CorMb5/xHgXjUCVgcu8a
a8ncdf9fjl3WS5p0ohetPbO2ZjWv+MaqrZOmUIgKbA4RpWZ/fU97P5BW9ylwmIDB
sWy0cMxg8MlvSdLytPieogaM0qMg3u5qXRGBr6Wmevkty0qgnmpGGc5zPiUbtOE8
CnFFqyxBpj5IOnG0KZGVihvn+iRxrv6GO7WWO92+Dc6m94U0EEiBR7QiOwARAQAB
tDRBV1MgU0FNIENMSSBQcmltYXJ5IDxhd3Mtc2FtLWNsaS1wcmltYXJ5QGFtYXpv
bi5jb20+iQI/BBMBCQApBQJkbkszAhsvBQkHhM4ABwsJCAcDAgEGFQgCCQoLBBYC
AwECHgECF4AACgkQQv1fenOtiFqTuhAAzi5+ju5UVOWqHKevOJSO08T4QB8HcqAE
SVO3mY6/j29knkcL8ubZP/DbpV7QpHPI2PB5qSXsiDTP3IYPbeY78zHSDjljaIK3
njJLMScFeGPyfPpwMsuY4nzrRIgAtXShPA8N/k4ZJcafnpNqKj7QnPxiC1KaIQWm
pOtvb8msUF3/s0UTa5Ys/lNRhVC0eGg32ogXGdojZA2kHZWdm9udLo4CDrDcrQT7
NtDcJASapXSQL63XfAS3snEc4e1941YxcjfYZ33rel8K9juyDZfi1slWR/L3AviI
QFIaqSHzyOtP1oinUkoVwL8ThevKD3Ag9CZflZLzNCV7yqlF8RlhEZ4zcE/3s9El
WzCFsozb5HfE1AZonmrDh3SyOEIBMcS6vG5dWnvJrAuSYv2rX38++K5Pr/MIAfOX
DOI1rtA+XDsHNv9lSwSy0lt+iClawZANO9IXCiN1rOYcVQlwzDFwCNWDgkwdOqS0
gOA2f8NF9lE5nBbeEuYquoOl1Vy8+ICbgOFs9LoWZlnVh7/RyY6ssowiU9vGUnHI
L8f9jqRspIz/Fm3JD86ntZxLVGkeZUz62FqErdohYfkFIVcv7GONTEyrz5HLlnpv
FJ0MR0HjrMrZrnOVZnwBKhpbLocTsH+3t5It4ReYEX0f1DIOL/KRwPvjMvBVkXY5
hblRVDQoOWc=
=d9oG
-----END PGP PUBLIC KEY BLOCK-----
```

2. Importieren Sie den öffentlichen Primärschlüssel in Ihren Schlüsselbund.

```
$ gpg --import primary-public-key.txt

gpg: directory `/home/.../.gnupg' created
gpg: new configuration file `/home/.../.gnupg/gpg.conf' created
gpg: WARNING: options in `/home/.../.gnupg/gpg.conf' are not yet active during this
 run
gpg: keyring `/home/.../.gnupg/secring.gpg' created
gpg: keyring `/home/.../.gnupg/pubring.gpg' created
gpg: /home/.../.gnupg/trustdb.gpg: trustdb created
```

```
gpg: key 73AD885A: public key "AWS SAM CLI Primary <aws-sam-cli-
primary@amazon.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
```

3. Kopieren Sie den öffentlichen Schlüssel des Unterzeichners und speichern Sie ihn als Datei auf Ihrem lokalen Computer. `.txt` Beispiel, *`signer-public-key.txt`*.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2.0.22 (GNU/Linux)

mQINBGgrxIgBEADGCTudveeeVbWpZDGX9Ni57mBRMVSJwQJ6F/PC34jw0DozxTtd
H+ZPsXLvLwerN/DVXbK8E1qNZ5RGptak8j7MPz+MC3n4txibEJpB61vpjJJM+9cC
7whaMLDT/SbykHYXdrnHqa8KsUJl7rPLJcaRN722NSxvYVMIOA9ffVXV7cfEyZi5
MbYF2Gc9LNbKaknImIva7EKeeh2/wI6YCqC5yytyfWU5dL6oHXsgTnFL9mhziMxv
WhyzawyJG6EJZsJ3WLlbIKApN6XZSXyCxOvlBrebYZjD5v0nA+TJaQ7is8atjtOI
DGe0AViw7kO8ChTpjA7YG/Uu7n/Fy7qLF/3Nz0b6cBNjemjBazQ3A3KNCpi5hqFM
Uo1WpoVLr5CXQnc0B3fBUnTIoxi0Sk5MKjH9AbYxfgqEX0ZJB9hAlc6LIEy0Yru6
MMBrIHE86IMl1NfE/DeLnCdPG23+1PttwyOt3+9z5QwmPe3VPpEfCySPcdxHKZSP
rLile8qDznEvlPDvQ0qkBxdMtVa2yct5VJkdqy6UrN2xa0dpspHjRUjHh/EY/xMt
fwMUjOKohaZ/1pjotCcksAsZWUxCNcFvLYxuxeytVk4F09Es1hj4ihhLUI+43/ic
3DHSEiext7Q8/UccNArkhSCT7UOvvL7QTuP+pjYTyiC8Vx6g/Y5Ht5+qywARAQAB
tDBBV1MgU0FNIENMSSBUZWFtIDxhd3Mtc2FtLWNsaS1zaWduZXJAYW1hem9uLmNv
bT6JAj8EEwEJACkFAmgrxIgCGy8FCQPCZwAHCwkIBwMCAQYVCAIJCgsEFgIDAQIe
AQIXgAAKCRBAlKuxvt/atJo6EAC/5C8uJs76W5f5V5XNAMzwBFiZuYpop3DRReCo
P68ZZylokAC9ShRZnIOujpDJtlNS7T/G00BzmcpspkYYE531ALaXcHWmb9XV0Ajg
J8iboAVBLY0C7mhL/cbJ3v9QlpXXjyTuhexkJCV8rdHVX/0H8WqTZplEaRuZ7p8q
PMxddg4ClwstYuH3O/dmNdlGqfb4Fqy8MnV1yGSXRs5Jf+sDlN2UO4mbpyk/mr1c
f/jFxmx86IkCWJVvdXWCVTe2AFy3NHCdLtdnEvFhokCOQd9wibUWX0j9vq4cVRZT
qamnpAQaOlH3lXOwrjqo8b1AIPoRWSfMtCYvh6kA8MAJv4cAznzXILSLtOE0mzaU
qp5qoy37wNIjeztX6c/q4wss05qTlJhnNu4s3nh5VHultooaYpmDxp+ala5TWeuM
KZDI4KdAGF4z0Raif+N53ndOYIiXkY0goUbsPCnVrCwoK9PjjyoJncq7c14wNl5O
IQUZEjyYAQDGZqs5XSfY4zW2cCXatrfozKF7R1kSU14DfJwPUyksoNAQEQezfXyq
kr0gfIWK1r2nMdqS7WgSx/ypS5kdyrHuPZdaYfEVtuezpoT2lQQxOSZqqlp5hI4R
nqmPte53WXJhbC0tgTIJWn+Uy/d5Q/aSIfD6o8gNLS1BDs1j1ku0XKu1sFCHUcZG
aerdsIkCHAQQAQkABgUCaCvFeAAKCRBC/V96c62IWt3/D/9gOLzWtz62lqJRCsri
wcA/yz88ayKb/GUv3FCT5Nd9JZt8y1tW+AE3SPTdcpfZmt5UN2sRzljO61mpKJzp
eBvYQ9og/34ZrRQqeg8bz02u34LKYl1gD0xY0bWtB7TGIxIZZYqZECoPR0Dp6ZzB
abzkRSsJkEk0vbZzJhfWFYs98qfp/G0suFSBE79O8Am33DB2jQ/Sollh1VmNE6Sv
EOgR6+2yEkS2D0+msJMa/V82v9gBTPnxSlNV1d8Dduvt9rbM3LoxiNXUgx/s52yY
U6H3bwUcQ3UY6uRe1UWo5QnMFcDwfg43+q5rmjB4xQyX/BaQyF5K0hZyG+42/pH1
EMwl8qN617FTxo3hvQUi/cBahlhQ8EVYsGnHDVxLCisbq5iZvp7+XtmMy1Q417gT
EQRo8feJh31elGWlccVR2pZgIm1PQ69dzzseHnnKkGhifik0bDGo5/IH2EgI1KFn
```

```
SG399RMU/qRzOPLVP3i+zSJmhMqG8cnZaUwE5V4P21vQSclhhd2Hv/C4SVKNqA2i
+oZbHj2vAkuzTTL075AoANebEjPGqwsKZi5mWUE5Pa931JeiXxWZlEB7rkgQ1PAB
fsDBhYLt4MxCWAhifLMA6uQ4BhXu2RuXOqNfSbqa8jVF6DB6cD8eAHGpPKfJOl30
LtZnq+n4SfeNbZjD2FQWZR4CrA==
=lHfs
-----END PGP PUBLIC KEY BLOCK-----
```

4. Importieren Sie den öffentlichen Schlüssel des Unterzeichners in Ihren Schlüsselbund.

```
$ gpg --import signer-public-key.txt

gpg: key 4094ABB1BEDFDAB4: public key "AWS SAM CLI Team <aws-sam-cli-
signer@amazon.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
gpg: no ultimately trusted keys found
```

Notieren Sie sich den Schlüsselwert aus der Ausgabe. Beispiel, *4094ABB1BEDFDAB4*.

5. Verwenden Sie den Schlüsselwert, um den Fingerabdruck des öffentlichen Schlüssels des Unterzeichners abzurufen und zu überprüfen.

```
$ gpg --fingerprint 4094ABB1BEDFDAB4

pub   rsa4096 2025-05-19 [SCEA] [expires: 2027-05-19]
      EF46 3E86 CA31 933B B688  CC1A 4094 ABB1 BEDF DAB4
uid           [ unknown] AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>
```

Der Fingerabdruck sollte folgenden Angaben entsprechen:

```
EF46 3E86 CA31 933B B688  CC1A 4094 ABB1 BEDF DAB4
```

Wenn die Zeichenfolge für den Fingerabdruck nicht übereinstimmt, verwenden Sie das AWS SAMCLI Installationsprogramm nicht. Wenden Sie sich an das AWS SAM Team, indem Sie ein Problem im GitHub aws-sam-cli Repository erstellen.

6. Überprüfen Sie die Signaturen des öffentlichen Schlüssels des Unterzeichners:

```
$ gpg --check-sigs 4094ABB1BEDFDAB4

pub   rsa4096 2025-05-19 [SCEA] [expires: 2027-05-19]
      EF463E86CA31933BB688CC1A4094ABB1BEDFDAB4
```

```
uid           [ unknown] AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>
sig!3        4094ABB1BEDFDAB4 2025-05-19  [self-signature]
sig!         42FD5F7A73AD885A 2025-05-19 AWS SAM CLI Primary <aws-sam-cli-
primary@amazon.com>
```

Wenn Sie Folgendes sehen`1 signature not checked due to a missing key`, wiederholen Sie die vorherigen Schritte, um die öffentlichen Schlüssel und den öffentlichen Schlüssel des Unterzeichners in Ihren Schlüsselbund zu importieren.

Sie sollten die Schlüsselwerte sowohl für den öffentlichen Primärschlüssel als auch für den öffentlichen Schlüssel des Unterzeichners aufgelistet sehen.

Nachdem Sie nun die Integrität des öffentlichen Schlüssels des Unterzeichners überprüft haben, können Sie den öffentlichen Schlüssel des Unterzeichners verwenden, um das Paketinstallationsprogramm zu überprüfen. AWS SAMCLI

Um die Integrität des zu überprüfen AWS SAM CLI Paket-Installer

1. Besorgen Sie sich die AWS SAMCLI Paketsignaturdatei — Laden Sie die Signaturdatei für das AWS SAMCLI Paketinstallationsprogramm mit dem folgenden Befehl herunter:

   ```
   $ wget https://github.com/aws/aws-sam-cli/releases/latest/download/aws-sam-cli-linux-arm64.zip.sig
   ```

2. Überprüfen Sie die Signaturdatei — Übergeben Sie sowohl die heruntergeladenen `.sig` `.zip` Dateien als auch die Dateien als Parameter an den gpg Befehl. Im Folgenden wird ein Beispiel gezeigt:

   ```
   $ gpg --verify aws-sam-cli-linux-arm64.zip.sig aws-sam-cli-linux-arm64.zip
   ```

   Die Ausgabe sollte folgendermaßen oder ähnlich aussehen:

   ```
   gpg: Signature made Mon 19 May 2025 01:21:57 AM UTC using RSA key ID
    4094ABB1BEDFDAB4
   gpg: Good signature from "AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>"
   gpg: WARNING: This key is not certified with a trusted signature!
   gpg:          There is no indication that the signature belongs to the owner.
   Primary key fingerprint: EF46 3E86 CA31 933B B688  CC1A 4094 ABB1 BEDF DAB4
   ```

- Die `WARNING: This key is not certified with a trusted signature!` Nachricht kann ignoriert werden. Dies liegt daran, dass zwischen Ihrem persönlichen PGP-Schlüssel (falls Sie einen haben) und dem AWS SAM CLI-PGP-Schlüssel keine Vertrauenskette besteht. Weitere Informationen finden Sie unter Web of Trust.
- Wenn die Ausgabe den Ausdruck enthältBAD `signature`, überprüfen Sie, ob Sie das Verfahren korrekt ausgeführt haben. Wenn Sie weiterhin diese Antwort erhalten, wenden Sie sich an das AWS SAM Team, indem Sie ein Problem im aws-sam-cli GitHub Repository erstellen und die heruntergeladene Datei nicht verwenden.

Die `Good signature from "AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>"` Meldung bedeutet, dass die Signatur verifiziert wurde und Sie mit der Installation fortfahren können.

x86_64 — Befehlszeilen-Installationsprogramm

AWS SAM benutzt GnuPG, um das ZIP-Installationsprogramm zu signieren AWS SAMCLI. Die Überprüfung erfolgt in den folgenden Schritten:

1. Verwenden Sie den öffentlichen Primärschlüssel, um den öffentlichen Schlüssel des Unterzeichners zu überprüfen.
2. Verwenden Sie den öffentlichen Schlüssel des Unterzeichners, um das Installationsprogramm des AWS SAMCLI Pakets zu verifizieren.

Um die Integrität des öffentlichen Schlüssels des Unterzeichners zu überprüfen

1. Kopieren Sie den öffentlichen Primärschlüssel und speichern Sie ihn als `.txt` Datei auf Ihrem lokalen Computer. Beispiel, *`primary-public-key.txt`*.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2.0.22 (GNU/Linux)

mQINBGRuSzMBEADsqiwOy78w7F4+sshaMFRIwRGNRm94p5Qey2KMZBxekFtoryVD
D9jEOnvupx4tvhfBHz5EcUHCEOdl4MTqdBy6vVAshozgxVb9RE8JpECn5lw7XC69
4Y7Gy1TKKQMEWtDXElkGxIFdUWvWjSnPlzfnoXwQYGeE93CUS3h5dImP22Yk1Ct6
eGGhlcbg1X4L8EpFMj7GvcsU8f7ziVI/PyC1Xwy39Q8/I67ip5eU5ddxO/xHqrbL
YC7+8pJPbRMej2twT2LrcpWWYAbprMtRoa6WfE0/thoo3xhHpIMHdPfAA86ZNGIN
kRLjGUg7jnPTRW4Oin3pCc8nT4Tfc1QERkHm641gTC/jUvpmQsM6h/FUVP2i5iE/
```

```
JHpJcMuL2Mg6zDo3x+3gTCf+Wqz3rZzxB+wQT3yryZs6efcQy7nROiRxYBxCSXX0
2cNYzsYLb/bYaW8yqWIHD5IqKhw269gp2E5Khs60zgS3CorMb5/xHgXjUCVgcu8a
a8ncdf9fjl3WS5p0ohetPbO2ZjWv+MaqrZOmUIgKbA4RpWZ/fU97P5BW9ylwmIDB
sWy0cMxg8MlvSdLytPieogaM0qMg3u5qXRGBr6Wmevkty0qgnmpGGc5zPiUbtOE8
CnFFqyxBpj5IOnG0KZGVihvn+iRxrv6GO7WWO92+Dc6m94U0EEiBR7QiOwARAQAB
tDRBV1MgU0FNIENMSSBQcmltYXJ5IDxhd3Mtc2FtLWNsaS1wcmltYXJ5QGFtYXpv
bi5jb20+iQI/BBMBCQApBQJkbkszAhsvBQkHhM4ABwsJCAcDAgEGFQgCCQoLBBYC
AwECHgECF4AACgkQQv1fenOtiFqTuhAAzi5+ju5UVOWqHKevOJSO08T4QB8HcqAE
SVO3mY6/j29knkcL8ubZP/DbpV7QpHPI2PB5qSXsiDTP3IYPbeY78zHSDjljaIK3
njJLMScFeGPyfPpwMsuY4nzrRIgAtXShPA8N/k4ZJcafnpNqKj7QnPxiC1KaIQWm
pOtvb8msUF3/s0UTa5Ys/lNRhVC0eGg32ogXGdojZA2kHZWdm9udLo4CDrDcrQT7
NtDcJASapXSQL63XfAS3snEc4e1941YxcjfYZ33rel8K9juyDZfi1slWR/L3AviI
QFIaqSHzyOtP1oinUkoVwL8ThevKD3Ag9CZflZLzNCV7yqlF8RlhEZ4zcE/3s9El
WzCFsozb5HfE1AZonmrDh3SyOEIBMcS6vG5dWnvJrAuSYv2rX38++K5Pr/MIAfOX
DOI1rtA+XDsHNv9lSwSy0lt+iClawZANO9IXCiN1rOYcVQlwzDFwCNWDgkwdOqS0
gOA2f8NF9lE5nBbeEuYquoOl1Vy8+ICbgOFs9LoWZlnVh7/RyY6ssowiU9vGUnHI
L8f9jqRspIz/Fm3JD86ntZxLVGkeZUz62FqErdohYfkFIVcv7GONTEyrz5HLlnpv
FJ0MR0HjrMrZrnOVZnwBKhpbLocTsH+3t5It4ReYEX0f1DIOL/KRwPvjMvBVkXY5
hblRVDQoOWc=
=d9oG
-----END PGP PUBLIC KEY BLOCK-----
```

2. Importieren Sie den öffentlichen Primärschlüssel in Ihren Schlüsselbund.

```
$ gpg --import primary-public-key.txt

gpg: directory `/home/.../.gnupg' created
gpg: new configuration file `/home/.../.gnupg/gpg.conf' created
gpg: WARNING: options in `/home/.../.gnupg/gpg.conf' are not yet active during this
 run
gpg: keyring `/home/.../.gnupg/secring.gpg' created
gpg: keyring `/home/.../.gnupg/pubring.gpg' created
gpg: /home/.../.gnupg/trustdb.gpg: trustdb created
gpg: key 73AD885A: public key "AWS SAM CLI Primary <aws-sam-cli-
primary@amazon.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
```

3. Kopieren Sie den öffentlichen Schlüssel des Unterzeichners und speichern Sie ihn als Datei auf Ihrem lokalen Computer. `.txt` Beispiel, *signer-public-key.txt*.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2.0.22 (GNU/Linux)
```

```
mQINBGgrxIgBEADGCTudveeeVbWpZDGX9Ni57mBRMVSJwQJ6F/PC34jw0DozxTtd
H+ZPsXLvLwerN/DVXbK8E1qNZ5RGptak8j7MPz+MC3n4txibEJpB61vpjJJM+9cC
7whaMLDT/SbykHYXdrnHqa8KsUJl7rPLJcaRN722NSxvYVMIOA9ffVXV7cfEyZi5
MbYF2Gc9LNbKaknImIva7EKeeh2/wI6YCqC5yytyfWU5dL6oHXsgTnFL9mhziMxv
WhyzawyJG6EJZsJ3WLlbIKApN6XZSXyCxOvlBrebYZjD5v0nA+TJaQ7is8atjtOI
DGe0AViw7kO8ChTpjA7YG/Uu7n/Fy7qLF/3Nz0b6cBNjemjBazQ3A3KNCpi5hqFM
Uo1WpoVLr5CXQnc0B3fBUnTIoxi0Sk5MKjH9AbYxfgqEX0ZJB9hAlc6LIEy0Yru6
MMBrIHE86IMl1NfE/DeLnCdPG23+1PttwyOt3+9z5QwmPe3VPpEfCySPcdxHKZSP
rLile8qDznEvlPDvQ0qkBxdMtVa2yct5VJkdqy6UrN2xa0dpspHjRUjHh/EY/xMt
fwMUjOKohaZ/1pjotCcksAsZWUxCNcFvLYxuxeytVk4F09Es1hj4ihhLUI+43/ic
3DHSEiext7Q8/UccNArkhSCT7UOvvL7QTuP+pjYTyiC8Vx6g/Y5Ht5+qywARAQAB
tDBBV1MgU0FNIENMSSBUZWFtIDxhd3Mtc2FtLWNsaS1zaWduZXJAYW1hem9uLmNv
bT6JAj8EEwEJACkFAmgrxIgCGy8FCQPCZwAHCwkIBwMCAQYVCAIJCgsEFgIDAQIe
AQIXgAAKCRBAlKuxvt/atJo6EAC/5C8uJs76W5f5V5XNAMzwBFiZuYpop3DRReCo
P68ZZylokAC9ShRZnIOujpDJtlNS7T/G00BzmcpspkYYE531ALaXcHWmb9XV0Ajg
J8iboAVBLY0C7mhL/cbJ3v9QlpXXjyTuhexkJCV8rdHVX/0H8WqTZplEaRuZ7p8q
PMxddg4ClwstYuH3O/dmNdlGqfb4Fqy8MnV1yGSXRs5Jf+sDlN2UO4mbpyk/mr1c
f/jFxmx86IkCWJVvdXWCVTe2AFy3NHCdLtdnEvFhokCOQd9wibUWX0j9vq4cVRZT
qamnpAQaOlH3lXOwrjqo8b1AIPoRWSfMtCYvh6kA8MAJv4cAznzXILSLtOE0mzaU
qp5qoy37wNIjeztX6c/q4wss05qTlJhnNu4s3nh5VHultooaYpmDxp+ala5TWeuM
KZDI4KdAGF4z0Raif+N53ndOYIiXkY0goUbsPCnVrCwoK9PjjyoJncq7c14wNl5O
IQUZEjyYAQDGZqs5XSfY4zW2cCXatrfozKF7R1kSU14DfJwPUyksoNAQEQezfXyq
kr0gfIWK1r2nMdqS7WgSx/ypS5kdyrHuPZdaYfEVtuezpoT2lQQxOSZqqlp5hI4R
nqmPte53WXJhbC0tgTIJWn+Uy/d5Q/aSIfD6o8gNLS1BDs1j1ku0XKu1sFCHUcZG
aerdsIkCHAQQAQkABgUCaCvFeAAKCRBC/V96c62IWt3/D/9gOLzWtz62lqJRCsri
wcA/yz88ayKb/GUv3FCT5Nd9JZt8y1tW+AE3SPTdcpfZmt5UN2sRzljO61mpKJzp
eBvYQ9og/34ZrRQqeg8bz02u34LKYl1gD0xY0bWtB7TGIxIZZYqZECoPR0Dp6ZzB
abzkRSsJkEk0vbZzJhfWFYs98qfp/G0suFSBE79O8Am33DB2jQ/Sollh1VmNE6Sv
EOgR6+2yEkS2D0+msJMa/V82v9gBTPnxSlNV1d8Dduvt9rbM3LoxiNXUgx/s52yY
U6H3bwUcQ3UY6uRe1UWo5QnMFcDwfg43+q5rmjB4xQyX/BaQyF5K0hZyG+42/pH1
EMwl8qN617FTxo3hvQUi/cBahlhQ8EVYsGnHDVxLCisbq5iZvp7+XtmMy1Q417gT
EQRo8feJh31elGWlccVR2pZgIm1PQ69dzzseHnnKkGhifik0bDGo5/IH2EgI1KFn
SG399RMU/qRzOPLVP3i+zSJmhMqG8cnZaUwE5V4P21vQSclhhd2Hv/C4SVKNqA2i
+oZbHj2vAkuzTTL075AoANebEjPGqwsKZi5mWUE5Pa931JeiXxWZlEB7rkgQ1PAB
fsDBhYLt4MxCWAhifLMA6uQ4BhXu2RuXOqNfSbqa8jVF6DB6cD8eAHGpPKfJOl30
LtZnq+n4SfeNbZjD2FQWZR4CrA==
=lHfs
-----END PGP PUBLIC KEY BLOCK-----
```

4. Importieren Sie den öffentlichen Schlüssel des Unterzeichners in Ihren Schlüsselbund.

```
$ gpg --import signer-public-key.txt
```

```
gpg: key 4094ABB1BEDFDAB4: public key "AWS SAM CLI Team <aws-sam-cli-
signer@amazon.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
gpg: no ultimately trusted keys found
```

Notieren Sie sich den Schlüsselwert aus der Ausgabe. Beispiel, *4094ABB1BEDFDAB4*.

5. Verwenden Sie den Schlüsselwert, um den Fingerabdruck des öffentlichen Schlüssels des Unterzeichners abzurufen und zu überprüfen.

```
$ gpg --fingerprint 4094ABB1BEDFDAB4

pub   rsa4096 2025-05-19 [SCEA] [expires: 2027-05-19]
      EF46 3E86 CA31 933B B688  CC1A 4094 ABB1 BEDF DAB4
uid           [ unknown] AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>
```

Der Fingerabdruck sollte folgenden Angaben entsprechen:

```
EF46 3E86 CA31 933B B688  CC1A 4094 ABB1 BEDF DAB4
```

Wenn die Zeichenfolge für den Fingerabdruck nicht übereinstimmt, verwenden Sie das AWS SAMCLI Installationsprogramm nicht. Wenden Sie sich an das AWS SAM Team, indem Sie ein Problem im GitHub aws-sam-cli Repository erstellen.

6. Überprüfen Sie die Signaturen des öffentlichen Schlüssels des Unterzeichners:

```
$ gpg --check-sigs 4094ABB1BEDFDAB4

pub   rsa4096 2025-05-19 [SCEA] [expires: 2027-05-19]
      EF463E86CA31933BB688CC1A4094ABB1BEDFDAB4
uid           [ unknown] AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>
sig!3        4094ABB1BEDFDAB4 2025-05-19  [self-signature]
sig!         42FD5F7A73AD885A 2025-05-19 AWS SAM CLI Primary <aws-sam-cli-
primary@amazon.com>
```

Wenn Sie Folgendes sehen`1 signature not checked due to a missing key`, wiederholen Sie die vorherigen Schritte, um die öffentlichen Schlüssel und den öffentlichen Schlüssel des Unterzeichners in Ihren Schlüsselbund zu importieren.

Sie sollten die Schlüsselwerte sowohl für den öffentlichen Primärschlüssel als auch für den öffentlichen Schlüssel des Unterzeichners aufgelistet sehen.

Nachdem Sie nun die Integrität des öffentlichen Schlüssels des Unterzeichners überprüft haben, können Sie den öffentlichen Schlüssel des Unterzeichners verwenden, um das Paketinstallationsprogramm zu überprüfen. AWS SAMCLI

Um die Integrität des zu überprüfen AWS SAM CLI Paket-Installer

1. Besorgen Sie sich die AWS SAMCLI Paketsignaturdatei — Laden Sie die Signaturdatei für das AWS SAMCLI Paketinstallationsprogramm mit dem folgenden Befehl herunter:

   ```
   $ wget https://github.com/aws/aws-sam-cli/releases/latest/download/aws-sam-cli-linux-x86_64.zip.sig
   ```

2. Überprüfen Sie die Signaturdatei — Übergeben Sie sowohl die heruntergeladenen `.sig` `.zip` Dateien als auch die Dateien als Parameter an den gpg Befehl. Im Folgenden wird ein Beispiel gezeigt:

   ```
   $ gpg --verify aws-sam-cli-linux-x86_64.zip.sig aws-sam-cli-linux-x86_64.zip
   ```

   Die Ausgabe sollte folgendermaßen oder ähnlich aussehen:

   ```
   gpg: Signature made Mon 19 May 2025 01:21:57 AM UTC using RSA key ID
    4094ABB1BEDFDAB4
   gpg: Good signature from "AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>"
   gpg: WARNING: This key is not certified with a trusted signature!
   gpg:          There is no indication that the signature belongs to the owner.
   Primary key fingerprint: EF46 3E86 CA31 933B B688  CC1A 4094 ABB1 BEDF DAB4
   ```

   - Die `WARNING: This key is not certified with a trusted signature!` Nachricht kann ignoriert werden. Dies liegt daran, dass zwischen Ihrem persönlichen PGP-Schlüssel (falls Sie einen haben) und dem AWS SAM CLI-PGP-Schlüssel keine Vertrauenskette besteht. Weitere Informationen finden Sie unter Web of Trust.
   - Wenn die Ausgabe den Ausdruck enthältBAD `signature`, überprüfen Sie, ob Sie das Verfahren korrekt ausgeführt haben. Wenn Sie weiterhin diese Antwort erhalten, wenden

Sie sich an das AWS SAM Team, indem Sie ein Problem im aws-sam-cli GitHub Repository erstellen und die heruntergeladene Datei nicht verwenden.

Die `Good signature from "AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>"` Meldung bedeutet, dass die Signatur verifiziert wurde und Sie mit der Installation fortfahren können.

macOS

GUI- und Befehlszeilen-Installer

Sie können die Integrität der Signaturdatei des AWS SAMCLI Paketinstallers mithilfe des `pkgutil` Tools oder manuell überprüfen.

Um dies mit pkgutil zu überprüfen

1. Führen Sie den folgenden Befehl aus und geben Sie den Pfad zum heruntergeladenen Installationsprogramm auf Ihrem lokalen Computer an:

   ```
   $ pkgutil --check-signature /path/to/aws-sam-cli-installer.pkg
   ```

   Im Folgenden wird ein Beispiel gezeigt:

   ```
   $ pkgutil --check-signature /Users/user/Downloads/aws-sam-cli-macos-arm64.pkg
   ```

2. Suchen Sie in der Ausgabe nach dem SHA256 fingerprint fürDeveloper ID Installer: AMZN Mobile LLC. Im Folgenden wird ein Beispiel gezeigt:

   ```
   Package "aws-sam-cli-macos-arm64.pkg":
      Status: signed by a developer certificate issued by Apple for distribution
      Notarization: trusted by the Apple notary service
      Signed with a trusted timestamp on: 2026-01-28 07:39:16 +0000
      Certificate Chain:
       1. Developer ID Installer: AMZN Mobile LLC (94KV3E626L)
          Expires: 2030-09-26 00:18:06 +0000
          SHA256 Fingerprint:
              5C 45 BE 63 FD 52 10 07 2D 66 56 77 5C A9 FF 25 91 6D 3F 01 F7 0E
              9A 8A 05 F6 2D 62 B2 88 8D A9
          ------------------------------------------------------------------------
       2. Developer ID Certification Authority
   ```

```
        Expires: 2031-09-17 00:00:00 +0000
        SHA256 Fingerprint:
            F1 6C D3 C5 4C 7F 83 CE A4 BF 1A 3E 6A 08 19 C8 AA A8 E4 A1 52 8F
            D1 44 71 5F 35 06 43 D2 DF 3A
        ------------------------------------------------------------------------
    3. Apple Root CA
        Expires: 2035-02-09 21:40:36 +0000
        SHA256 Fingerprint:
            B0 B1 73 0E CB C7 FF 45 05 14 2C 49 F1 29 5E 6E DA 6B CA ED 7E 2C
            68 C5 BE 91 B5 A1 10 01 F0 24
```

3. Das Developer ID Installer: AMZN Mobile LLC SHA256 fingerprint sollte dem folgenden Wert entsprechen:

```
5C 45 BE 63 FD 52 10 07 2D 66 56 77 5C A9 FF 25 91 6D 3F 01 F7 0E 9A 8A 05 F6 2D 62
 B2 88 8D A9
```

   Wenn die Zeichenfolge für den Fingerabdruck nicht übereinstimmt, verwenden Sie das AWS SAMCLI Installationsprogramm nicht. Wenden Sie sich an das AWS SAM Team, indem Sie ein Problem im GitHub aws-sam-cli Repository erstellen. Wenn die Zeichenfolge für den Fingerabdruck übereinstimmt, können Sie mit der Verwendung des Paketinstallationsprogramms fortfahren.

Um das Paketinstallationsprogramm manuell zu überprüfen

- Weitere Informationen finden Sie auf der Apple-Support-Website unter So überprüfen Sie die Echtheit manuell heruntergeladener Apple-Softwareupdates.

Windows

Das AWS SAMCLI Installationsprogramm ist als MSI Dateien für das Windows Betriebssystem verpackt.

Um die Integrität des Installationsprogramms zu überprüfen

1. Right-click klicken Sie auf das Installationsprogramm und öffnen Sie das Eigenschaftenfenster.
2. Wählen Sie die Registerkarte Digital Signatures aus.
3. Wählen Sie in der Signaturliste Amazon Web Services, Inc. und dann Details aus.

4. Falls die Registerkarte General nicht bereits ausgewählt ist, klicken Sie darauf und dann auf View Certificate.
5. Wählen Sie die Registerkarte Details aus, und anschließend die Option All (Alle) in der Dropdown-Liste Show (Zeigen), wenn diese nicht bereits ausgewählt ist.
6. Scrollen Sie nach unten zum Feld Thumbprint und wählen Sie Thumbprint aus. Der gesamte Thumbprint-Wert wird im unteren Fenster angezeigt.
7. Ordnen Sie den Fingerabdruckwert dem folgenden Wert zu. Wenn der Wert übereinstimmt, fahren Sie mit der Installation fort. Falls nicht, wenden Sie sich an das AWS SAM Team, indem Sie ein Problem im GitHub aws-sam-cli Repository erstellen.

   ```
   6c 4e f0 59 67 db 0a 2c 69 87 17 31 ec 9e 51 5d 7c d6 e9 52
   ```

Überprüfen Sie den Hashwert

Linux

x86_64 — Befehlszeilen-Installationsprogramm

Überprüfen Sie die Integrität und Authentizität der heruntergeladenen Installationsdateien, indem Sie mit dem folgenden Befehl einen Hashwert generieren:

```
$ sha256sum aws-sam-cli-linux-x86_64.zip
```

Die Ausgabe sollte wie im folgenden Beispiel aussehen:

```
<64-character SHA256 hash value> aws-sam-cli-linux-x86_64.zip
```

Vergleichen Sie den SHA-256 64-stelligen Hashwert mit dem Hashwert für Ihre gewünschte AWS SAMCLI Version in den AWS SAMCLIVersionshinweisen unterGitHub.

macOS

GUI- und Befehlszeilen-Installer

Überprüfen Sie die Integrität und Authentizität des heruntergeladenen Installationsprogramms, indem Sie mit dem folgenden Befehl einen Hashwert generieren:

```
$ shasum -a 256 path-to-pkg-installer/name-of-pkg-installer

# Examples
$ shasum -a 256 ~/Downloads/aws-sam-cli-macos-arm64.pkg
$ shasum -a 256 ~/Downloads/aws-sam-cli-macos-x86_64.pkg
```

Vergleichen Sie Ihren SHA-256 64-stelligen Hashwert mit dem entsprechenden Wert im AWS SAMCLIGitHubVersionshinweis-Repository.

Tutorial: Stellen Sie eine Hello World-Anwendung bereit mit AWS SAM

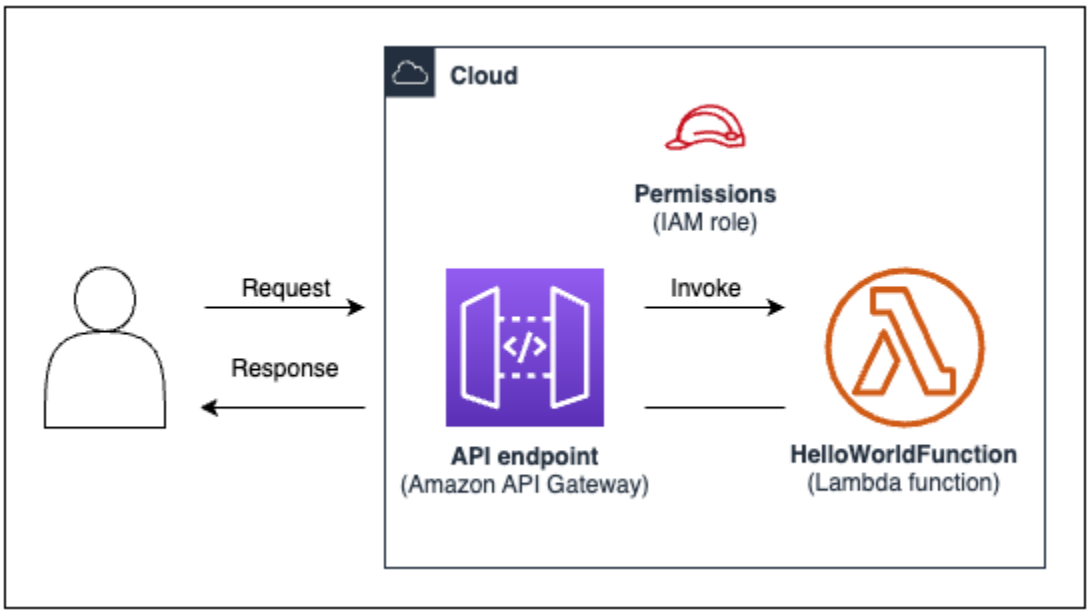
In diesem Tutorial verwenden Sie die AWS Serverless Application Model Befehlszeilenschnittstelle (AWS SAMCLI), um Folgendes auszuführen:

- Initialisieren, erstellen und implementieren Sie eine Hello World-Beispielanwendung.
- Nehmen Sie lokale Änderungen vor und synchronisieren Sie mit AWS CloudFormation.
- Führen Sie lokale Tests auf Ihrem Entwicklungshost durch.
- Löschen Sie die Beispielanwendung aus dem AWS Cloud.

Die Hello World-Beispielanwendung implementiert ein grundlegendes API-Backend. Sie besteht aus den folgenden Ressourcen:

- Amazon API Gateway — API-Endpunkt, den Sie zum Aufrufen Ihrer Funktion verwenden werden.
- AWS Lambda— Funktion, die die HTTP-API-GET-Anfrage verarbeitet und eine `hello world` Nachricht zurückgibt.
- AWS Identity and Access Management (IAM) -Rolle — Gewährt den Diensten Berechtigungen zur sicheren Interaktion.

Das folgende Diagramm zeigt die Komponenten dieser Anwendung:



Themen

- Voraussetzungen
- Schritt 1: Initialisieren Sie die Hello World-Beispielanwendung
- Schritt 2: Erstellen Sie Ihre Anwendung
- Schritt 3: Stellen Sie Ihre Anwendung auf dem bereit AWS Cloud
- Schritt 4: Führen Sie Ihre Anwendung aus
- Schritt 5: Interagieren Sie mit Ihrer Funktion im AWS Cloud
- Schritt 6: Ändern und synchronisieren Sie Ihre Anwendung mit AWS Cloud
- Schritt 7: (Optional) Testen Sie Ihre Anwendung lokal
- Schritt 8: Löschen Sie Ihre Anwendung aus dem AWS Cloud
- Fehlerbehebung
- Weitere Informationen

Voraussetzungen

Vergewissern Sie sich, dass Sie Folgendes abgeschlossen haben:

- AWS SAM Voraussetzungen
- Installieren Sie das AWS SAMCLI

> **Note**
>
> Wenn Sie Visual Studio Code für die serverlose Entwicklung bevorzugen, finden Sie step-by-step Anweisungen zur Verwendung der VS Code-Oberfläche unter Erstellen serverloser Anwendungen im AWS Toolkit for VS Code-Benutzerhandbuch.

Schritt 1: Initialisieren Sie die Hello World-Beispielanwendung

In diesem Schritt verwenden Sie das, AWS SAMCLI um ein Hello World-Beispielanwendungsprojekt auf Ihrem lokalen Computer zu erstellen.

Um die Hello World-Beispielanwendung zu initialisieren

1. Führen Sie in Ihrer Befehlszeile den folgenden Befehl von einem Startverzeichnis Ihrer Wahl aus:

   ```
   $ sam init
   ```

   > **Note**
   >
   > Dieser Befehl initialisiert Ihre serverlose Anwendung und erstellt ein Projektverzeichnis. Das Verzeichnis enthält mehrere Dateien und Ordner, einschließlich der `template.yaml` Datei, die Ihre Vorlage ist. AWS SAM
   > Die installierte Python-Version muss der `Runtime` unter angegebenen Eigenschaft entsprechen`template.yaml`. Für Produktionsworkloads empfehlen wir die Verwendung von Python 3.12 oder früheren Versionen, die in AWS Umgebungen vollständig unterstützt werden. Wenn die Python-Versionen nicht übereinstimmen, treten Build-Fehler auf.
   > So beheben Sie Probleme mit der Versionskompatibilität:
   >
   > - Geben Sie bei der Initialisierung eine kompatible Laufzeit an: `sam init --runtime python3.9`
   > - Ändern Sie die `Runtime` Eigenschaft `template.yaml` nach der Initialisierung

2. Das AWS SAMCLI führt Sie durch die Initialisierung einer neuen Anwendung. Konfigurieren Sie Folgendes:

   1. Wählen Sie AWS Schnellstartvorlagen aus, um eine Startvorlage auszuwählen.

2. Wählen Sie die Vorlage Hello World Example und laden Sie sie herunter.
3. Verwenden Sie die Python Laufzeit und den `zip` Pakettyp.
4. Deaktivieren Sie für dieses Tutorial die AWS X-Ray Ablaufverfolgung. Weitere Informationen finden Sie unter Was ist AWS X-Ray? im AWS X-Ray Entwicklerhandbuch.
5. Deaktivieren Sie für dieses Tutorial die Überwachung mit Amazon CloudWatch Application Insights. Weitere Informationen finden Sie unter Amazon CloudWatch Application Insights im CloudWatch Amazon-Benutzerhandbuch.
6. Deaktivieren Sie für dieses Tutorial die Einstellung von Structured Logging im JSON-Format für Ihre Lambda-Funktionen.
7. Benennen Sie Ihre Anwendung als Sam-App.

Um den AWS SAMCLI interaktiven Flow zu verwenden:

- Klammern (`[ ]`) stehen für Standardwerte. Lassen Sie Ihre Antwort leer, um den Standardwert auszuwählen.
- Geben Sie **y** für Ja und **n** für Nein ein.

Im Folgenden finden Sie ein Beispiel für den `sam init` interaktiven Ablauf:

```
$ sam init
...
Which template source would you like to use?
    1 - AWS Quick Start Templates
    2 - Custom Template Location
Choice: 1

Choose an AWS Quick Start application template
    1 - Hello World Example
    2 - Data processing
    3 - Hello World Example with Powertools for AWS Lambda
    4 - Multi-step workflow
    5 - Scheduled task
    6 - Standalone function
    7 - Serverless API
    8 - Infrastructure event management
    9 - Lambda Response Streaming
   10 - GraphQLApi Hello World Example
   11 - Full Stack
```

```
    12 - Lambda EFS example
    13 - Serverless Connector Hello World Example
    14 - Multi-step workflow with Connectors
    15 - DynamoDB Example
    16 - Machine Learning
Template: 1

Use the most popular runtime and package type? (Python and zip) [y/N]: y

Would you like to enable X-Ray tracing on the function(s) in your application?  [y/
N]: ENTER

Would you like to enable monitoring using CloudWatch Application Insights?
For more info, please view https://docs.aws.amazon.com/AmazonCloudWatch/latest/
monitoring/cloudwatch-application-insights.html [y/N]: ENTER

Would you like to set Structured Logging in JSON format on your Lambda functions?
 [y/N]: ENTER

Project name [sam-app]: ENTER
```

3. Der AWS SAMCLI lädt Ihre Startvorlage herunter und erstellt die Verzeichnisstruktur des Anwendungsprojekts auf Ihrem lokalen Computer. Im Folgenden finden Sie ein Beispiel für die AWS SAMCLI Ausgabe:

```
Cloning from https://github.com/aws/aws-sam-cli-app-templates (process may take a
 moment)

    -----------------------
    Generating application:
    -----------------------
    Name: sam-app
    Runtime: python3.9
    Architectures: x86_64
    Dependency Manager: pip
    Application Template: hello-world
    Output Directory: .
    Configuration file: sam-app/samconfig.toml

    Next steps can be found in the README file at sam-app/README.md


Commands you can use next
```

```
=========================
[*] Create pipeline: cd sam-app && sam pipeline init --bootstrap
[*] Validate SAM template: cd sam-app && sam validate
[*] Test Function in the Cloud: cd sam-app && sam sync --stack-name {stack-name} --
watch
```

4. Gehen Sie von Ihrer Befehlszeile aus in das neu erstellte `sam-app` Verzeichnis. Das Folgende ist ein Beispiel für das, was der erstellt AWS SAMCLI hat:

> **Note**
>
> Wenn der `tree` Befehl nicht auto installiert wird, führen Sie diesen Befehl aus: `brew install tree`

```
$ cd sam-app

$ tree

### README.md
### __init__.py
### events
#   ### event.json
### hello_world
#   ### __init__.py
#   ### app.py
#   ### requirements.txt
### samconfig.toml
### template.yaml
### tests
    ### __init__.py
    ### integration
    #   ### __init__.py
    #   ### test_api_gateway.py
    ### requirements.txt
    ### unit
        ### __init__.py
        ### test_handler.py

6 directories, 14 files
```

Einige wichtige Dateien, die Sie hervorheben sollten:

- `hello_world/app.py`— Enthält Ihren Lambda-Funktionscode.
- `hello_world/requirements.txt`— Enthält alle Python Abhängigkeiten, die Ihre Lambda-Funktion benötigt.
- `samconfig.toml`— Konfigurationsdatei für Ihre Anwendung, in der die von der AWS SAMCLI verwendeten Standardparameter gespeichert werden.
- `template.yaml`— Die AWS SAM Vorlage, die Ihren Anwendungsinfrastrukturcode enthält.

Sie haben jetzt eine vollständig erstellte serverlose Anwendung auf Ihrem lokalen Computer!

Schritt 2: Erstellen Sie Ihre Anwendung

In diesem Schritt verwenden Sie die, AWS SAMCLI um Ihre Anwendung zu erstellen und die Bereitstellung vorzubereiten. Beim Erstellen AWS SAMCLI erstellt der ein `.aws-sam` Verzeichnis und organisiert dort Ihre Funktionsabhängigkeiten, Ihren Projektcode und Ihre Projektdateien.

Um Ihre Anwendung zu erstellen

- Führen Sie in Ihrer Befehlszeile im `sam-app` Projektverzeichnis Folgendes aus:

```
$ sam build
```

> **Note**
>
> Wenn Sie es nicht Python auf Ihrem lokalen Computer haben, verwenden Sie stattdessen den sam build --use-container Befehl. Dadurch AWS SAMCLI wird ein Docker Container erstellt, der die Laufzeit und die Abhängigkeiten Ihrer Funktion enthält. Dieser Befehl erfordert Docker auf Ihrem lokalen Computer. Informationen zur Installation Docker finden Sie unterInstallieren von Docker.

Im Folgenden finden Sie ein Beispiel für die AWS SAMCLI Ausgabe:

```
$ sam build
Starting Build use cache
Manifest file is changed (new hash: 3298f1304...d4d421) or dependency folder (.aws-
sam/deps/4d3dfad6-a267-47a6-a6cd-e07d6fae318c) is missing for (HelloWorldFunction),
 downloading dependencies and copying/building source
```

```
Building codeuri: /Users/.../Demo/sam-tutorial1/sam-app/hello_world runtime:
 python3.9 metadata: {} architecture: x86_64 functions: HelloWorldFunction
Running PythonPipBuilder:CleanUp
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource
Running PythonPipBuilder:CopySource

Build Succeeded

Built Artifacts  : .aws-sam/build
Built Template   : .aws-sam/build/template.yaml

Commands you can use next
=========================
[*] Validate SAM template: sam validate
[*] Invoke Function: sam local invoke
[*] Test Function in the Cloud: sam sync --stack-name {{stack-name}} --watch
[*] Deploy: sam deploy --guided
```

Das Folgende ist ein verkürztes Beispiel für das von der AWS SAM CLI erstellte `.aws-sam` Verzeichnis:

```
.aws-sam
### build
#   ### HelloWorldFunction
#   #   ### __init__.py
#   #   ### app.py
#   #   ### requirements.txt
#   ### template.yaml
### build.toml
```

Einige wichtige Dateien, die Sie hervorheben sollten:

- `build/HelloWorldFunction`— Enthält Ihren Lambda-Funktionscode und Abhängigkeiten. Das AWS SAMCLI erstellt ein Verzeichnis für jede Funktion in Ihrer Anwendung.
- `build/template.yaml`— Enthält eine Kopie Ihrer AWS SAM Vorlage, auf die CloudFormation bei der Bereitstellung verwiesen wird.
- `build.toml`— Konfigurationsdatei, in der Standardparameterwerte gespeichert werden, auf die AWS SAMCLI beim Erstellen und Bereitstellen Ihrer Anwendung verwiesen wird.

Sie sind jetzt bereit, Ihre Anwendung auf dem bereitzustellen AWS Cloud.

Schritt 3: Stellen Sie Ihre Anwendung auf dem bereit AWS Cloud

> **Note**
>
> Dieser Schritt erfordert die Konfiguration der AWS Anmeldeinformationen. Weitere Informationen finden Sie unter Schritt 2: Verwenden Sie den AWS CLI zur Konfiguration AWS Anmeldedaten in AWS SAM Voraussetzungen.

In diesem Schritt verwenden Sie die, AWS SAMCLI um Ihre Anwendung auf dem bereitzustellen AWS Cloud. Das AWS SAMCLI wird Folgendes tun:

- Führt Sie durch die Konfiguration Ihrer Anwendungseinstellungen für die Bereitstellung.
- Laden Sie Ihre Anwendungsdateien auf Amazon Simple Storage Service (Amazon S3) hoch.
- Verwandeln Sie Ihre AWS SAM Vorlage in eine CloudFormation Vorlage. Anschließend wird Ihre Vorlage in den CloudFormation Dienst hochgeladen, um Ihre AWS Ressourcen bereitzustellen.

So stellen Sie Ihre -Anwendung bereit

1. Führen Sie in Ihrer Befehlszeile im `sam-app` Projektverzeichnis Folgendes aus:

   ```
   $ sam deploy --guided
   ```

2. Folgen Sie dem AWS SAMCLI interaktiven Ablauf, um Ihre Anwendungseinstellungen zu konfigurieren. Konfigurieren Sie Folgendes:

   1. Der CloudFormation Stack-Name — Ein Stack ist eine Sammlung von AWS Ressourcen, die Sie als eine Einheit verwalten können. Weitere Informationen finden Sie unter Arbeiten mit Stacks im AWS CloudFormation Benutzerhandbuch.
   2. Das AWS-Region, auf dem Sie Ihren CloudFormation Stack bereitstellen möchten. Weitere Informationen finden Sie unter CloudFormation -Endpunkte im AWS CloudFormation -Benutzerhandbuch.
   3. Deaktivieren Sie für dieses Tutorial die Bestätigung von Änderungen vor der Bereitstellung.

4. IAM-Rollenerstellung zulassen — Auf diese Weise können AWS SAM Sie die IAM-Rolle erstellen, die für die Interaktion zwischen Ihrer API-Gateway-Ressource und der Lambda-Funktionsressource erforderlich ist.
5. Deaktivieren Sie für dieses Tutorial die Deaktivierung des Rollbacks.
6. HelloWorldFunction Ohne definierte Autorisierung zulassen — Diese Meldung wird angezeigt, weil Ihr API-Gateway-Endpunkt so konfiguriert ist, dass er ohne Autorisierung öffentlich zugänglich ist. Da dies die vorgesehene Konfiguration für Ihre Hello World-Anwendung ist, warten Sie, AWS SAMCLI bis der Vorgang abgeschlossen ist. Weitere Informationen zur Konfiguration der Autorisierung finden Sie unterKontrollieren Sie den API-Zugriff mit Ihrer AWS SAM Vorlage.
7. Argumente in Konfigurationsdatei speichern — Dadurch wird die `samconfig.toml` Datei Ihrer Anwendung mit Ihren Bereitstellungseinstellungen aktualisiert.
8. Wählen Sie den Namen der Standardkonfigurationsdatei aus.
9. Wählen Sie die Standardkonfigurationsumgebung aus.

Im Folgenden finden Sie ein Beispiel für die Ausgabe des `sam deploy --guided` interaktiven Ablaufs:

```
$ sam deploy --guided

Configuring SAM deploy
======================

    Looking for config file [samconfig.toml] :  Found
    Reading default arguments  :  Success

    Setting default arguments for 'sam deploy'
    =========================================
    Stack Name [sam-app]: ENTER
    AWS Region [us-west-2]: ENTER
    #Shows you resources changes to be deployed and require a 'Y' to initiate
 deploy
    Confirm changes before deploy [Y/n]: n
    #SAM needs permission to be able to create roles to connect to the resources in
 your template
    Allow SAM CLI IAM role creation [Y/n]: ENTER
    #Preserves the state of previously provisioned resources when an operation
 fails
```

```
    Disable rollback [y/N]: ENTER
    HelloWorldFunction may not have authorization defined, Is this okay? [y/N]: y
    Save arguments to configuration file [Y/n]: ENTER
    SAM configuration file [samconfig.toml]: ENTER
    SAM configuration environment [default]: ENTER
```

3. Der AWS SAMCLI stellt Ihre Anwendung wie folgt bereit:

   - Das AWS SAMCLI erstellt einen Amazon S3 S3-Bucket und lädt Ihr `.aws-sam` Verzeichnis hoch.
   - Das AWS SAMCLI wandelt Ihre AWS SAM Vorlage in den Service um CloudFormation und lädt sie in den Service hoch. CloudFormation
   - CloudFormation stellt Ihre Ressourcen bereit.

   Während der Bereitstellung AWS SAMCLI zeigt das Ihren Fortschritt an. Im Folgenden finden Sie ein Beispiel für eine Ausgabe:

```
Looking for resources needed for deployment:

    Managed S3 bucket: aws-sam-cli-managed-default-samclisam-s3-demo-
bucket-1a4x26zbcdkqr
    A different default S3 bucket can be set in samconfig.toml

    Parameter "stack_name=sam-app" in [default.deploy.parameters] is defined as a
 global parameter [default.global.parameters].
    This parameter will be only saved under [default.global.parameters] in /
Users/.../Demo/sam-tutorial1/sam-app/samconfig.toml.

    Saved arguments to config file
    Running 'sam deploy' for future deployments will use the parameters saved
 above.
    The above parameters can be changed by modifying samconfig.toml
    Learn more about samconfig.toml syntax at
    https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/
serverless-sam-cli-config.html

File with same data already exists at sam-app/da3c598813f1c2151579b73ad788cac8,
 skipping upload

    Deploying with following values
    ===============================
```

```
    Stack name                   : sam-app
    Region                       : us-west-2
    Confirm changeset            : False
    Disable rollback             : False
    Deployment s3 bucket         : aws-sam-cli-managed-default-samclisam-s3-demo-
bucket-1a4x26zbcdkqr
    Capabilities                 : ["CAPABILITY_IAM"]
    Parameter overrides          : {}
    Signing Profiles             : {}

Initiating deployment
=====================

File with same data already exists at sam-
app/2bebf67c79f6a743cc5312f6dfc1efee.template, skipping upload


Waiting for changeset to be created..

CloudFormation stack changeset
---------------------------------------------------------------------------------------------
Operation                              LogicalResourceId
 ResourceType                           Replacement
---------------------------------------------------------------------------------------------
* Modify                               HelloWorldFunction
 AWS::Lambda::Function                  False
* Modify                               ServerlessRestApi
 AWS::ApiGateway::RestApi               False
- Delete                               AwsSamAutoDependencyLayerNestedSt
 AWS::CloudFormation::Stack             N/A
                                       ack
---------------------------------------------------------------------------------------------


Changeset created successfully. arn:aws:cloudformation:us-
west-2:012345678910:changeSet/samcli-deploy1678917603/22e05525-08f9-4c52-a2c4-
f7f1fd055072


2023-03-15 12:00:16 - Waiting for stack create/update to complete

CloudFormation events from stack operations (refresh every 0.5 seconds)
---------------------------------------------------------------------------------------------
```

```
ResourceStatus                          ResourceType
 LogicalResourceId                       ResourceStatusReason
-------------------------------------------------------------------------------------------
UPDATE_IN_PROGRESS                      AWS::Lambda::Function
 HelloWorldFunction                      -
UPDATE_COMPLETE                         AWS::Lambda::Function
 HelloWorldFunction                      -
UPDATE_COMPLETE_CLEANUP_IN_PROGRE       AWS::CloudFormation::Stack          sam-app
                           -
SS
DELETE_IN_PROGRESS                      AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt       -
                                                                             ack
DELETE_COMPLETE                         AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt       -
                                                                             ack
UPDATE_COMPLETE                         AWS::CloudFormation::Stack          sam-app
                           -
-------------------------------------------------------------------------------------------

CloudFormation outputs from deployed stack
-------------------------------------------------------------------------------------------
Outputs
-------------------------------------------------------------------------------------------
Key                 HelloWorldFunctionIamRole
Description         Implicit IAM Role created for Hello World function
Value               arn:aws:iam::012345678910:role/sam-app-
HelloWorldFunctionRole-15GLOUR9LMT1W

Key                 HelloWorldApi
Description         API Gateway endpoint URL for Prod stage for Hello World
 function
Value               https://<restapiid>.execute-api.us-west-2.amazonaws.com/Prod/
hello/

Key                 HelloWorldFunction
Description         Hello World Lambda Function ARN
Value               arn:aws:lambda:us-west-2:012345678910:function:sam-app-
HelloWorldFunction-yQDNe17r9maD
-------------------------------------------------------------------------------------------


Successfully created/updated stack - sam-app in us-west-2
```

Ihre Anwendung ist jetzt bereitgestellt und läuft im AWS Cloud!

Schritt 4: Führen Sie Ihre Anwendung aus

In diesem Schritt senden Sie eine GET-Anfrage an Ihren API-Endpunkt und sehen die Ausgabe Ihrer Lambda-Funktion.

Um Ihren API-Endpunktwert zu erhalten

1. Suchen Sie anhand der Informationen, die AWS SAMCLI im vorherigen Schritt angezeigt wurden, den `Outputs` Abschnitt. Suchen Sie in diesem Abschnitt nach Ihrer `HelloWorldApi` Ressource, um Ihren HTTP-Endpunktwert zu ermitteln. Im Folgenden finden Sie ein Beispiel für eine Ausgabe:

```
---------------------------------------------------------------------------------------------
Outputs
---------------------------------------------------------------------------------------------
...
Key                 HelloWorldApi
Description         API Gateway endpoint URL for Prod stage for Hello World
 function
Value               https://ets1gv8lxi.execute-api.us-west-2.amazonaws.com/Prod/
hello/
...
---------------------------------------------------------------------------------------------
```

2. Alternativ können Sie den sam list endpoints --output json Befehl verwenden, um diese Informationen abzurufen. Im Folgenden finden Sie ein Beispiel für eine Ausgabe:

```
$ sam list endpoints --output json
2023-03-15 12:39:19 Loading policies from IAM...
2023-03-15 12:39:25 Finished loading policies from IAM.
[
  {
    "LogicalResourceId": "HelloWorldFunction",
    "PhysicalResourceId": "sam-app-HelloWorldFunction-yQDNe17r9maD",
    "CloudEndpoint": "-",
    "Methods": "-"
  },
  {
    "LogicalResourceId": "ServerlessRestApi",
    "PhysicalResourceId": "ets1gv8lxi",
```

```
    "CloudEndpoint": [
      "https://ets1gv8lxi.execute-api.us-west-2.amazonaws.com/Prod",
      "https://ets1gv8lxi.execute-api.us-west-2.amazonaws.com/Stage"
    ],
    "Methods": [
      "/hello['get']"
    ]
  }
]
```

Um Ihre Funktion aufzurufen

- Senden Sie mit Ihrem Browser oder der Befehlszeile eine GET-Anfrage an Ihren API-Endpunkt. Im Folgenden finden Sie ein Beispiel für die Verwendung des Befehls curl:

```
$ curl https://ets1gv8lxi.execute-api.us-west-2.amazonaws.com/Prod/hello/
{"message": "hello world"}
```

Schritt 5: Interagieren Sie mit Ihrer Funktion im AWS Cloud

In diesem Schritt verwenden Sie die, AWS SAMCLI um Ihre Lambda-Funktion in der aufzurufen. AWS Cloud

So rufen Sie Ihre Lambda-Funktion in der Cloud auf

1. Notieren Sie sich Ihre Funktionen `LogicalResourceId` aus dem vorherigen Schritt. Dies sollte `HelloWorldFunction` sein.
2. Führen Sie in Ihrer Befehlszeile aus dem `sam-app` Projektverzeichnis Folgendes aus:

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app
```

3. Das AWS SAMCLI ruft Ihre Funktion in der Cloud auf und gibt eine Antwort zurück. Im Folgenden finden Sie ein Beispiel für eine Ausgabe:

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app

Invoking Lambda Function HelloWorldFunction
START RequestId: d5ef494b-5f45-4086-86fd-d7322fa1a1f9 Version: $LATEST
END RequestId: d5ef494b-5f45-4086-86fd-d7322fa1a1f9
```

```
REPORT RequestId: d5ef494b-5f45-4086-86fd-d7322fa1a1f9  Duration: 6.62 ms
 Billed Duration: 7 ms     Memory Size: 128 MB     Max Memory Used: 67 MB  Init
 Duration: 164.06 ms
{"statusCode":200,"body":"{\"message\":\"hello world\"}"}%
```

Schritt 6: Ändern und synchronisieren Sie Ihre Anwendung mit AWS Cloud

In diesem Schritt verwenden Sie den AWS SAMCLI sam sync --watch Befehl, um lokale Änderungen mit dem zu synchronisieren AWS Cloud.

Um Sam Sync zu verwenden

1. Führen Sie in Ihrer Befehlszeile im `sam-app` Projektverzeichnis Folgendes aus:

   ```
   $ sam sync --watch
   ```

2. Sie AWS SAMCLI werden aufgefordert, zu bestätigen, dass Sie einen Entwicklungsstapel synchronisieren. Da der sam sync --watch Befehl automatisch lokale Änderungen AWS Cloud in Echtzeit bereitstellt, empfehlen wir ihn nur für Entwicklungsumgebungen.

   Der AWS SAMCLI führt eine erste Bereitstellung durch, bevor er mit der Überwachung auf lokale Änderungen beginnt. Im Folgenden finden Sie ein Beispiel für eine Ausgabe:

   ```
   $ sam sync --watch
   The SAM CLI will use the AWS Lambda, Amazon API Gateway, and AWS StepFunctions APIs
    to upload your code without
   performing a CloudFormation deployment. This will cause drift in your
    CloudFormation stack.
   **The sync command should only be used against a development stack**.

   Confirm that you are synchronizing a development stack.

   Enter Y to proceed with the command, or enter N to cancel:
    [Y/n]: y
   Queued infra sync. Waiting for in progress code syncs to complete...
   Starting infra sync.
   Manifest is not changed for (HelloWorldFunction), running incremental build
   Building codeuri: /Users/.../Demo/sam-tutorial1/sam-app/hello_world runtime:
    python3.9 metadata: {} architecture: x86_64 functions: HelloWorldFunction
   Running PythonPipBuilder:CopySource
   ```

```
Build Succeeded

Successfully packaged artifacts and wrote output template to file /var/
folders/45/5ct135bx3fn2551_ptl5g6_80000gr/T/tmpq3x9vh63.
Execute the following command to deploy the packaged template
sam deploy --template-file /var/folders/45/5ct135bx3fn2551_ptl5g6_80000gr/T/
tmpq3x9vh63 --stack-name <YOUR STACK NAME>


    Deploying with following values
    ===============================
    Stack name                   : sam-app
    Region                       : us-west-2
    Disable rollback             : False
    Deployment s3 bucket         : aws-sam-cli-managed-default-samclisam-s3-demo-
bucket-1a4x26zbcdkqr
    Capabilities                 : ["CAPABILITY_NAMED_IAM",
 "CAPABILITY_AUTO_EXPAND"]
    Parameter overrides          : {}
    Signing Profiles             : null

Initiating deployment
=====================


2023-03-15 13:10:05 - Waiting for stack create/update to complete

CloudFormation events from stack operations (refresh every 0.5 seconds)
---------------------------------------------------------------------------------------
ResourceStatus                      ResourceType
 LogicalResourceId                   ResourceStatusReason
---------------------------------------------------------------------------------------
UPDATE_IN_PROGRESS                  AWS::CloudFormation::Stack          sam-app
                        Transformation succeeded
CREATE_IN_PROGRESS                  AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt   -
                                                                        ack
CREATE_IN_PROGRESS                  AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt   Resource creation Initiated
                                                                        ack
CREATE_COMPLETE                     AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt   -
                                                                        ack
```

```
UPDATE_IN_PROGRESS                  AWS::Lambda::Function
 HelloWorldFunction                  -
UPDATE_COMPLETE                     AWS::Lambda::Function
 HelloWorldFunction                  -
UPDATE_COMPLETE_CLEANUP_IN_PROGRE   AWS::CloudFormation::Stack          sam-app
                          -
SS
UPDATE_COMPLETE                     AWS::CloudFormation::Stack          sam-app
                          -
-------------------------------------------------------------------------------------------

CloudFormation outputs from deployed stack
-------------------------------------------------------------------------------------------
Outputs
-------------------------------------------------------------------------------------------
Key                 HelloWorldFunctionIamRole
Description         Implicit IAM Role created for Hello World function
Value               arn:aws:iam::012345678910:role/sam-app-
HelloWorldFunctionRole-15GLOUR9LMT1W

Key                 HelloWorldApi
Description         API Gateway endpoint URL for Prod stage for Hello World
 function
Value               https://ets1gv8lxi.execute-api.us-west-2.amazonaws.com/Prod/
hello/

Key                 HelloWorldFunction
Description         Hello World Lambda Function ARN
Value               arn:aws:lambda:us-west-2:012345678910:function:sam-app-
HelloWorldFunction-yQDNe17r9maD
-------------------------------------------------------------------------------------------


Stack update succeeded. Sync infra completed.

Infra sync completed.
CodeTrigger not created as CodeUri or DefinitionUri is missing for
 ServerlessRestApi.
```

Als Nächstes ändern Sie Ihren Lambda-Funktionscode. Das AWS SAMCLI erkennt diese Änderung automatisch und synchronisiert Ihre Anwendung mit dem AWS Cloud.

Um Ihre Anwendung zu ändern und zu synchronisieren

1. Öffnen Sie die `sam-app/hello_world/app.py` Datei in der IDE Ihrer Wahl.
2. Ändern Sie die Datei `message` und speichern Sie sie. Im Folgenden wird ein Beispiel gezeigt:

```
import json
...
def lambda_handler(event, context):
    ...
    return {
        "statusCode": 200,
        "body": json.dumps({
            "message": "hello everyone!",
            ...
        }),
    }
```

3. Das AWS SAMCLI erkennt Ihre Änderung und synchronisiert Ihre Anwendung mit dem AWS Cloud. Im Folgenden finden Sie ein Beispiel für eine Ausgabe:

```
Syncing Lambda Function HelloWorldFunction...
Manifest is not changed for (HelloWorldFunction), running incremental build
Building codeuri: /Users/.../Demo/sam-tutorial1/sam-app/hello_world runtime:
 python3.9 metadata: {} architecture: x86_64 functions: HelloWorldFunction
Running PythonPipBuilder:CopySource
Finished syncing Lambda Function HelloWorldFunction.
```

4. Um Ihre Änderung zu überprüfen, senden Sie erneut eine GET-Anfrage an Ihren API-Endpunkt.

```
$ curl https://ets1gv8lxi.execute-api.us-west-2.amazonaws.com/Prod/hello/
{"message": "hello everyone!"}
```

Schritt 7: (Optional) Testen Sie Ihre Anwendung lokal

> **Note**
>
> Dieser Schritt ist optional.

> **Important**
>
> Dieser Schritt erfordert Docker auf Ihrem lokalen Computer. Sie müssen das Docker installiert und konfiguriert haben, um das AWS SAMCLI für lokale Tests verwenden zu können. Weitere Informationen finden Sie unter Installieren von Docker.

In diesem Schritt verwenden Sie den AWS SAMCLI sam local Befehl, um Ihre Anwendung lokal zu testen. Um dies zu erreichen, AWS SAMCLI erstellt der eine lokale Umgebung mitDocker. Diese lokale Umgebung emuliert die cloudbasierte Ausführungsumgebung Ihrer Lambda-Funktion.

Dafür müssen Sie Folgendes tun:

1. Erstellen Sie eine lokale Umgebung für Ihre Lambda-Funktion und rufen Sie sie auf.
2. Hosten Sie Ihren HTTP-API-Endpunkt lokal und verwenden Sie ihn, um Ihre Lambda-Funktion aufzurufen.

Um Ihre Lambda-Funktion lokal aufzurufen

1. Führen Sie in Ihrer Befehlszeile im `sam-app` Projektverzeichnis Folgendes aus:

   ```
   $ sam local invoke
   ```

2. Das AWS SAMCLI erstellt einen lokalen Docker Container und ruft Ihre Funktion auf. Im Folgenden finden Sie ein Beispiel für eine Ausgabe:

   ```
   $ sam local invoke
   Invoking app.lambda_handler (python3.9)
   Local image was not found.
   Removing rapid images for repo public.ecr.aws/sam/emulation-python3.9
   Building image.....................
   Using local image: public.ecr.aws/lambda/python:3.9-rapid-x86_64.

   Mounting /Users/.../Demo/sam-tutorial1/sam-app/.aws-sam/build/HelloWorldFunction
    as /var/task:ro,delegated inside runtime container
   START RequestId: b046db01-2a00-415d-af97-35f3a02e9eb6 Version: $LATEST
   END RequestId: b046db01-2a00-415d-af97-35f3a02e9eb6
   REPORT RequestId: b046db01-2a00-415d-af97-35f3a02e9eb6    Init Duration: 1.01 ms
       Duration: 633.45 ms    Billed Duration: 634 ms    Memory Size: 128 MB    Max
    Memory Used: 128 MB
   ```

```
{"statusCode": 200, "body": "{\"message\": \"hello world\"}"}
```

Um Ihre API lokal zu hosten

1. Führen Sie in Ihrer Befehlszeile im `sam-app` Projektverzeichnis Folgendes aus:

   ```
   $ sam local start-api
   ```

2. Der AWS SAMCLI erstellt einen lokalen Docker Container für Ihre Lambda-Funktion und erstellt einen lokalen HTTP-Server, um Ihren API-Endpunkt zu simulieren. Im Folgenden finden Sie ein Beispiel für eine Ausgabe:

   ```
   $ sam local start-api
   Initializing the lambda functions containers.
   Local image is up-to-date
   Using local image: public.ecr.aws/lambda/python:3.9-rapid-x86_64.

   Mounting /Users/.../Demo/sam-tutorial1/sam-app/.aws-sam/build/HelloWorldFunction
    as /var/task:ro,delegated inside runtime container
   Containers Initialization is done.
   Mounting HelloWorldFunction at http://127.0.0.1:3000/hello [GET]
   You can now browse to the above endpoints to invoke your functions. You do not
    need to restart/reload SAM CLI while working on your functions, changes will be
    reflected instantly/automatically. If you used sam build before running local
    commands, you will need to re-run sam build for the changes to be picked up. You
    only need to restart AWS SAM CLI if you update your AWS SAM template
   2023-03-15 14:25:21 WARNING: This is a development server. Do not use it in a
    production deployment. Use a production WSGI server instead.
    * Running on http://127.0.0.1:3000
   2023-03-15 14:25:21 Press CTRL+C to quit
   ```

3. Senden Sie mit Ihrem Browser oder der Befehlszeile eine GET-Anfrage an Ihren lokalen API-Endpunkt. Im Folgenden finden Sie ein Beispiel für die Verwendung des Befehls curl:

   ```
   $ curl http://127.0.0.1:3000/hello
   {"message": "hello world"}
   ```

Schritt 8: Löschen Sie Ihre Anwendung aus dem AWS Cloud

In diesem Schritt verwenden Sie den AWS SAMCLI sam delete Befehl, um Ihre Anwendung aus dem zu löschen AWS Cloud.

Um Ihre Anwendung aus dem zu löschen AWS Cloud

1. Führen Sie in Ihrer Befehlszeile im `sam-app` Projektverzeichnis Folgendes aus:

```
$ sam delete
```

2. Sie AWS SAMCLI werden aufgefordert, dies zu bestätigen. Anschließend werden der Amazon S3 S3-Bucket und der CloudFormation Stack Ihrer Anwendung gelöscht. Im Folgenden finden Sie ein Beispiel für eine Ausgabe:

```
$ sam delete
    Are you sure you want to delete the stack sam-app in the region us-west-2 ? [y/
N]: y
    Are you sure you want to delete the folder sam-app in S3 which contains the
 artifacts? [y/N]: y
    - Deleting S3 object with key c6ce8fa8b5a97dd022ecd006536eb5a4
    - Deleting S3 object with key 5d513a459d062d644f3b7dd0c8b56a2a.template
    - Deleting S3 object with key sam-app/2bebf67c79f6a743cc5312f6dfc1efee.template
    - Deleting S3 object with key sam-app/6b208d0e42ad15d1cee77d967834784b.template
    - Deleting S3 object with key sam-app/da3c598813f1c2151579b73ad788cac8
    - Deleting S3 object with key sam-app/f798cdd93cee188a71d120f14a035b11
    - Deleting Cloudformation stack sam-app

Deleted successfully
```

Fehlerbehebung

Informationen zur Behebung von Problemen finden Sie unterAWS SAMCLIProblembehandlung. AWS SAMCLI

Weitere Informationen

Weitere Informationen AWS SAM finden Sie in den folgenden Ressourcen:

- Der komplette AWS SAM Workshop — Ein Workshop, in dem Sie viele der wichtigsten Funktionen des Workshops kennenlernen AWS SAM möchten.
- Sessions with SAM — Videoserie, die von unserem AWS Serverless Developer Advocate-Team zur Verwendung AWS SAM erstellt wurde.
- Serverless Land — Website, die die neuesten Informationen, Blogs, Videos, Code und Lernressourcen für AWS Serverless zusammenfasst.

Lambda-Funktionen in Anwendungen umwandeln AWS SAM

Wenn Sie eine Funktion in der Lambda-Konsole haben, die Sie in VS Code als Infrastructure as Code (IaC) verwalten möchten, können Sie sie in VS Code übertragen und dann in eine AWS SAM Vorlage konvertieren. Wenn Ihre Funktion in eine AWS SAM Vorlage umgewandelt wurde, können Sie die Versionierung Ihrer Infrastruktur steuern, Bereitstellungen automatisieren, Funktionen aus der Ferne debuggen und konsistente Umgebungen während Ihres gesamten Entwicklungslebenszyklus aufrechterhalten. Mit lokalen und Remote-Debugging-Funktionen können Sie Probleme auch effektiver beheben, indem Sie Ihren Code schrittweise durchgehen, Variablen überprüfen und Breakpoints sowohl lokal als auch in der Cloud setzen. AWS

Anweisungen zum Verschieben Ihrer Funktionen aus der Konsole und zum Einrichten der lokalen Entwicklung finden Sie unter Lokales Entwickeln von Lambda-Funktionen mit VS Code im AWS Lambda Entwicklerhandbuch. Folgen Sie diesen Anweisungen, um eine Lambda-Funktion von der Konsole nach VS Code zu verschieben und sie dann in eine AWS SAM Vorlage zu konvertieren. Danach können Sie:

- Debuggen Sie Ihre Funktion aus der Ferne in der AWS Cloud. Einzelheiten finden Sie unter Ferndebuggen von Lambda-Funktionen mit VS Code im AWS Lambda Entwicklerhandbuch.
- Verwalten Sie die zugehörigen Ressourcen und Berechtigungen als Infrastruktur als Code. Einzelheiten finden Sie unter Using Lambda with Infrastructure as Code (IaC) im AWS Lambda Entwicklerhandbuch
- Stellen Sie Ihre Anwendung bereit und aktualisieren Sie sie mit dem. AWS SAMCLI Details hierzu finden Sie unter AWS SAM CLI.
- Entwerfen Sie Ihre serverlose Architektur visuell mit Infrastructure Composer. Einzelheiten finden Sie im Infrastructure Composer Developer Guide.

Weitere Informationen

Weitere Informationen AWS SAM finden Sie in den folgenden Ressourcen:

- Der komplette AWS SAM Workshop — Ein Workshop, in dem Sie viele der wichtigsten Funktionen des Workshops kennenlernen AWS SAM möchten.
- Sessions with SAM — Videoserie, die von unserem AWS Serverless Developer Advocate-Team zur Verwendung AWS SAM erstellt wurde.
- Serverless Land — Website, die die neuesten Informationen, Blogs, Videos, Code und Lernressourcen für AWS Serverless zusammenfasst.

Wie benutzt man AWS Serverless Application Model (AWS SAM)

Die wichtigsten Tools, die Sie zur Entwicklung Ihrer Anwendung verwenden, sind die AWS SAMCLIAWS SAM Vorlage und das AWS SAM Projekt (das ist Ihr Anwendungsprojektverzeichnis). Sie verwenden diese Tools, um:

1. Entwickeln Sie Ihre Anwendung(Dazu gehören die Initialisierung Ihrer Anwendung, die Definition Ihrer Ressourcen und die Erstellung Ihrer Anwendung).
2. Testen Sie Ihre Anwendung.
3. Debuggen Sie Ihre Anwendung.
4. Stellen Sie Ihre Anwendung und Ressourcen bereit.
5. Überwachen Sie Ihre Anwendung.

AWS SAM erstellt Ihr AWS SAM Projekt, nachdem Sie den sam init Befehl ausgeführt und den nachfolgenden Workflow abgeschlossen haben. Sie definieren Ihre serverlose Anwendung, indem Sie Ihrem AWS SAM Projekt Code hinzufügen. Ihr AWS SAM Projekt besteht zwar aus einer Reihe von Dateien und Ordnern, die wichtigste Datei darin ist jedoch Ihre AWS SAM Vorlage (benannt`template.yaml`). In dieser Vorlage schreiben Sie Ihren Code, um Ressourcen, Ereignisquellenzuordnungen und andere Eigenschaften auszudrücken, die Ihre serverlose Anwendung definieren.

Die AWS SAMCLI enthält ein Repository mit Befehlen, die Sie in Ihrem Projekt verwenden. AWS SAM Genauer gesagt AWS SAMCLI ist es das, was Sie verwenden, um Ihr Projekt zu erstellen, zu transformieren, bereitzustellen, zu debuggen, zu verpacken, zu initialisieren und zu synchronisieren AWS SAM . Mit anderen Worten, es ist das, was Sie verwenden, um Ihr AWS SAM Projekt in Ihre serverlose Anwendung zu verwandeln.

Weitere Informationen zu diesen Tools finden Sie in den folgenden Kapiteln:

AWS SAMCLI- Erfahren Sie mehr über AWS SAMCLI und wie Sie damit Ihre serverlosen Anwendungen erstellen und ausführen können.

AWS SAM Projekt und AWS SAM Vorlage — Erfahren Sie mehr über die AWS SAM Projektstruktur und wie Sie anhand der AWS SAM Vorlage Ihre serverlose Anwendung definieren können.

AWS SAM CLI

AWS Serverless Application Model Command Line Interface (AWS SAMCLI) ist ein Befehlszeilentool für die lokale Entwicklung und das Testen serverloser Anwendungen. AWS SAMCLIDamit können Sie Ihre serverlosen Anwendungen lokal erstellen, transformieren, bereitstellen, debuggen, verpacken, initialisieren und synchronisieren, bevor Sie sie in der Cloud bereitstellen.

AWS SAMCLIfunktioniert mit serverlosen Anwendungen, die mithilfe verschiedener Frameworks und Infrastructure-as-Code-Tools (IaC) definiert wurden und unterschiedliche Unterstützungsstufen bieten:

- AWS SAM Vorlagen — Bietet systemeigenen Support mit dem gesamten Funktionsumfang, einschließlich lokaler Test-, Debugging-, Paketierungs- und Bereitstellungsfunktionen.
- AWS CDK-Anwendungen — Unterstützt das lokale Testen von Lambda-Funktionen, nachdem Sie die AWS CDK-Anwendung mit dem Befehl cdk synth in CloudFormation Vorlagen synthetisiert haben.
- CloudFormation Vorlagen — Bietet direkte Kompatibilität, da sie AWS SAM erweitert und serverlose Ressourcen unterstützt CloudFormation, die in Standardvorlagen definiert sind. CloudFormation
- Terraform-Anwendungen — Bietet eingeschränkte Unterstützung für die Erstellung und das lokale Testen von Lambda-Funktionen. Erfordert, dass Sie AWS SAM Vorlagenartefakte generieren, die Ihre Terraform-definierten Lambda-Funktionen darstellen.

Für die umfassendste Funktionsunterstützung und ein optimiertes Entwicklererlebnis empfehlen wir die Verwendung nativer Vorlagen. AWS SAM

Themen

- Wie AWS SAMCLI werden Befehle dokumentiert
- Konfiguration der AWS SAMCLI
- AWS SAMCLIKernbefehle
- Lokales Testen mit AWS SAMCLI

Wie AWS SAMCLI werden Befehle dokumentiert

AWS SAMCLIBefehle werden im folgenden Format dokumentiert:

- LinuxEingabeaufforderung — Die Eingabeaufforderung ist standardmäßig dokumentiert und wird als ($) angezeigt. Bei Windows spezifischen Befehlen wird (>) als Eingabeaufforderung verwendet. Lassen Sie das Eingabeaufforderungssymbol weg, wenn Sie Befehle eingeben.
- Verzeichnis – Wenn Befehle in einem bestimmten Verzeichnis ausgeführt werden müssen, steht der Name des Verzeichnisses vor dem Eingabeaufforderungssymbol.
- Benutzereingabe – Befehlstext, den Sie in der Befehlszeile eingeben, ist als **`user input`** formatiert.
- Austauschbarer Text — Variabler Text, wie z. B. Dateinamen und Parameter, wird als *`replaceable text`* formatiert. Bei mehrzeiligen Befehlen oder Befehlen, für die eine bestimmte Tastatureingabe erforderlich ist, können Tastatureingaben auch als austauschbarer Text angezeigt werden. Beispiel, *ENTER*.
- Ausgabe — Die als Antwort auf den Befehl zurückgegebene Ausgabe ist formatiert als. `computer output`

Der folgende `sam deploy` Befehl und die folgende Ausgabe sind ein Beispiel:

```
$ sam deploy --guided --template template.yaml

Configuring SAM deploy
======================

    Looking for config file [samconfig.toml] :  Found
    Reading default arguments  :  Success

    Setting default arguments for 'sam deploy'
    =========================================
    Stack Name [sam-app]: ENTER
    AWS Region [us-west-2]: ENTER
    #Shows you resources changes to be deployed and require a 'Y' to initiate deploy
    Confirm changes before deploy [y/N]: ENTER
    #SAM needs permission to be able to create roles to connect to the resources in
 your template
    Allow SAM CLI IAM role creation [Y/n]: ENTER
    #Preserves the state of previously provisioned resources when an operation fails
    Disable rollback [y/N]: ENTER
    HelloWorldFunction may not have authorization defined, Is this okay? [y/N]: y
    Save arguments to configuration file [Y/n]: ENTER
    SAM configuration file [samconfig.toml]: ENTER
    SAM configuration environment [default]: ENTER
```

1. `sam deploy --guided --template template.yaml`ist der Befehl, den Sie in der Befehlszeile eingeben.
2. **`sam deploy --guided --template`**sollte unverändert bereitgestellt werden.
3. *`template.yaml`*kann durch Ihren spezifischen Dateinamen ersetzt werden.
4. Die Ausgabe beginnt um`Configuring SAM deploy`.
5. *`y`*Geben Sie in der Ausgabe die ersetzbaren Werte an, die Sie angeben. *`ENTER`*

Konfiguration der AWS SAMCLI

Einer der Vorteile von AWS SAM besteht darin, dass die Zeit eines Entwicklers optimiert wird, indem sich wiederholende Aufgaben entfallen. AWS SAMCLIenthält eine nach diesem `samconfig` Zweck benannte Konfigurationsdatei. Standardmäßig AWS SAMCLI ist keine Konfiguration für erforderlich, aber Sie können Ihre Konfigurationsdatei aktualisieren, damit Sie Befehle mit weniger Parametern ausführen können, indem AWS SAM Sie stattdessen Ihre benutzerdefinierten Parameter in Ihrer Konfigurationsdatei referenzieren. Die Beispiele in der folgenden Tabelle zeigen, wie Sie Ihre Befehle optimieren können:

Original	Optimiert mit **`samconfig`**
sam build --cached --parallel --use-containers	sam build
sam local invoke --env-vars locals.json	sam local invoke
sam local start-api --env-vars locals.json --warm-containers EAGER	sam local start-api

Das AWS SAMCLI bietet eine Reihe von Befehlen, mit denen Entwickler serverlose Anwendungen erstellen, entwickeln und bereitstellen können. Jeder dieser Befehle ist mit optionalen Flags konfigurierbar, die auf den Einstellungen der Anwendung und des Entwicklers basieren. Weitere Informationen finden Sie im AWS SAMCLIInhalt in GitHub

In den Themen in diesem Abschnitt erfahren Sie, wie Sie Ihre Standardeinstellungen erstellen AWS SAMCLIKonfigurationsdatei und anpassen können, um die Entwicklungszeit für Ihre serverlose Anwendung zu optimieren.

Themen

- So erstellen Sie Ihre Konfigurationsdatei (die samconfig Datei)
- Konfigurieren Sie die Projekteinstellungen
- Konfigurieren Sie Anmeldeinformationen und Grundeinstellungen

So erstellen Sie Ihre Konfigurationsdatei (die **`samconfig`** Datei)

Die AWS SAMCLI Konfigurationsdatei (Dateiname`samconfig`) ist eine Textdatei, die normalerweise die TOML-Struktur verwendet, aber auch in YAML sein kann. Wenn Sie eine AWS Schnellstartvorlage verwenden, wird diese Datei erstellt, wenn Sie den sam init Befehl ausführen. Sie können diese Datei aktualisieren, wenn Sie eine Anwendung mithilfe des sam deploy -\-guided Befehls bereitstellen.

Nach Abschluss der Bereitstellung enthält die `samconfig` Datei ein Profil mit dem Namen, `default` ob Sie die Standardwerte verwendet haben. Wenn Sie den deploy Befehl erneut ausführen, werden AWS SAM die gespeicherten Konfigurationseinstellungen aus diesem Profil übernommen.

Der Vorteil der `samconfig` Datei besteht darin, dass sie die Konfigurationseinstellungen für alle anderen Befehle AWS SAM speichert, die zusätzlich zum Befehl deploy verfügbar sind. Neben diesen Werten, die bei einer neuen Bereitstellung erstellt werden, gibt es eine Reihe von Attributen, die Sie in der `samconfig` Datei festlegen können, um andere Aspekte des Entwickler-Workflows zu vereinfachen AWS SAMCLI.

Konfigurieren Sie die Projekteinstellungen

Sie können projektspezifische Einstellungen, wie z. B. AWS SAMCLI Befehlsparameterwerte, in einer Konfigurationsdatei angeben, um sie mit dem zu verwenden. AWS SAMCLI Weitere Informationen zu dieser Konfigurationsdatei finden Sie unter. AWS SAMCLIKonfigurationsdatei

Verwenden von Konfigurationsdateien

Konfigurationsdateien sind nach Umgebung, Befehl und Parameterwert strukturiert. Weitere Informationen finden Sie unter Grundlagen der Konfigurationsdatei.

Um eine neue Umgebung zu konfigurieren

1. Geben Sie Ihre neue Umgebung in Ihrer Konfigurationsdatei an.

   Im Folgenden finden Sie ein Beispiel für die Angabe einer neuen `prod` Umgebung:

TOML

```
[prod.global.parameters]
```

YAML

```
prod:
  global:
    parameters:
```

2. Geben Sie Parameterwerte als Schlüssel-Wert-Paare im Parameterbereich der Konfigurationsdatei an.

   Im Folgenden finden Sie ein Beispiel für die Angabe des Stack-Namens Ihrer Anwendung für die `prod` Umgebung.

   TOML

   ```
   [prod.global.parameters]
   stack_name = "prod-app"
   ```

   YAML

   ```
   prod:
     global:
       parameters:
         stack_name: prod-app
   ```

3. Verwenden Sie die `--config-env` Option, um die zu verwendende Umgebung anzugeben.

   Im Folgenden wird ein Beispiel gezeigt:

   ```
   $ sam deploy --config-env "prod"
   ```

Um Parameterwerte zu konfigurieren

1. Geben Sie den AWS SAMCLI Befehl an, für den Sie Parameterwerte konfigurieren möchten. Verwenden Sie den `global` Bezeichner, um Parameterwerte für alle AWS SAMCLI Befehle zu konfigurieren.

Im Folgenden finden Sie ein Beispiel für die Angabe von Parameterwerten für den `sam deploy` Befehl der `default` Umgebung:

TOML

```
[default.deploy.parameters]
confirm_changeset = true
```

YAML

```
default:
  deploy:
    parameters:
      confirm_changeset: true
```

Das Folgende ist ein Beispiel für die Angabe von Parameterwerten für alle AWS SAMCLI Befehle in der `default` Umgebung:

TOML

```
[default.global.parameters]
stack_name = "sam-app"
```

YAML

```
default:
  global:
    parameters:
      stack_name: sam-app
```

2. Sie können auch Parameterwerte angeben und Ihre Konfigurationsdatei über den AWS SAMCLI interaktiven Ablauf ändern.

   Im Folgenden finden Sie ein Beispiel für den `sam deploy --guided` interaktiven Ablauf:

```
$ sam deploy --guided

Configuring SAM deploy
======================
```

```
    Looking for config file [samconfig.toml] :  Found
    Reading default arguments  :  Success

    Setting default arguments for 'sam deploy'
    =========================================
    Stack Name [sam-app]: ENTER
    AWS Region [us-west-2]: ENTER
    #Shows you resources changes to be deployed and require a 'Y' to initiate
deploy
    Confirm changes before deploy [Y/n]: n
    #SAM needs permission to be able to create roles to connect to the resources in
your template
    Allow SAM CLI IAM role creation [Y/n]: ENTER
    #Preserves the state of previously provisioned resources when an operation
fails
    Disable rollback [y/N]: ENTER
    HelloWorldFunction may not have authorization defined, Is this okay? [y/N]: y
    Save arguments to configuration file [Y/n]: ENTER
    SAM configuration file [samconfig.toml]: ENTER
    SAM configuration environment [default]: ENTER
```

Weitere Informationen finden Sie unter Konfigurationsdateien erstellen und ändern.

Beispiele

Einfaches TOML Beispiel

Das Folgende ist ein Beispiel für eine `samconfig.toml` Konfigurationsdatei:

```
...
version = 0.1

[default]
[default.global]
[default.global.parameters]
stack_name = "sam-app"

[default.build.parameters]
cached = true
parallel = true

[default.deploy.parameters]
```

```
capabilities = "CAPABILITY_IAM"
confirm_changeset = true
resolve_s3 = true

[default.sync.parameters]
watch = true

[default.local_start_api.parameters]
warm_containers = "EAGER"

[prod]
[prod.sync]
[prod.sync.parameters]
watch = false
```

Einfaches YAML Beispiel

Das Folgende ist ein Beispiel für eine `samconfig.yaml` Konfigurationsdatei:

```
version 0.1
default:
  global:
    parameters:
      stack_name: sam-app
  build:
    parameters:
      cached: true
      parallel: true
  deploy:
    parameters:
      capabilities: CAPABILITY_IAM
      confirm_changeset: true
      resolve_s3: true
  sync:
    parameters:
      watch: true
  local_start_api:
    parameters:
      warm_containers: EAGER
prod:
  sync:
    parameters:
      watch: false
```

Konfigurieren Sie Anmeldeinformationen und Grundeinstellungen

Verwenden Sie AWS Command Line Interface (AWS CLI), um grundlegende Einstellungen wie AWS Anmeldeinformationen, den Namen der Standardregion und das Standardausgabeformat zu konfigurieren. Nach der Konfiguration können Sie diese Einstellungen mit dem verwenden AWS SAMCLI. Weitere Informationen finden Sie im AWS Command Line Interface Benutzerhandbuch im Folgenden:

- Grundlagen der Konfiguration
- Einstellungen für die Konfiguration und die Anmeldeinformationsdatei
- Benannte Profile für AWS CLI
- Verwendung eines für IAM Identity Center aktivierten benannten Profils

Anweisungen zur schnellen Einrichtung finden Sie unterSchritt 2: Verwenden Sie den AWS CLI zur Konfiguration AWS Anmeldedaten.

AWS SAMCLIKernbefehle

AWS SAMCLIenthält einige grundlegende Befehle, die Sie zum Erstellen, Erstellen, Testen, Bereitstellen und Synchronisieren Ihrer serverlosen Anwendung verwenden. Die folgende Tabelle listet diese Befehle auf und enthält Links mit weiteren Informationen zu den einzelnen Befehlen.

Eine vollständige Liste der AWS SAMCLI Befehle finden Sie unterAWS SAMCLIBefehlsreferenz.

Befehl	Was es tut	Verwandte Themen
sam build	Bereitet eine Anwendung für nachfolgende Schritte im Entwickler-Workflow vor, z. B. lokales Testen oder Bereitstellen in der AWS Cloud.	• Einführung in das Bauen mit AWS SAM • gleicher Build
sam deploy	Stellt eine Anwendung in der AWS Cloud bereit unter Verwendung von CloudFormation.	• Einführung in die Bereitstellung mit AWS SAM • sam deploy

Befehl	Was es tut	Verwandte Themen
sam init	Bietet Optionen zum Initialisieren und Erstellen einer neuen serverlosen Anwendung.	• Erstellen Sie Ihre Bewerbung in AWS SAM • sam init
sam local	Stellt Unterbefehle bereit, um Ihre serverlosen Anwendungen lokal zu testen.	• Einführung in das Testen mit dem sam local Befehl • sam local generate-event • sam local invoke • Sam lokale Start-API • Sam Local Start-Lambda
sam remote invoke	Bietet eine Möglichkeit, mit unterstützten AWS Ressourcen in der AWS Cloud zu interagieren.	• Einführung in das Testen in der Cloud mit sam remote invoke • sam remote invoke
sam remote test-event	Bietet eine Möglichkeit, auf gemeinsam nutzbare Testereignisse für Ihre AWS Lambda-Funktionen zuzugreifen und diese zu verwalten.	• Einführung in Cloud-Tests mit sam remote test-event • sam remote test-event
sam sync	Bietet Optionen zum schnellen Synchronisieren lokaler Anwendungsänderungen mit der AWS Cloud.	• Einführung in die Verwendung von sam sync to sync to AWS Cloud • Sam Sync

Lokales Testen mit AWS SAMCLI

Das AWS SAMCLI ermöglicht es Ihnen, serverlose Anwendungen lokal in verschiedenen Infrastructure-as-Code-Tools (IaC) zu testen. In diesem Handbuch wird erklärt, wie Sie das AWS SAMCLI für lokale Tests mit verschiedenen IaC-Frameworks verwenden.

Zu den wichtigsten Vorteilen der Verwendung AWS SAMCLI für lokale Tests gehören:

- Schnelle Entwicklung — Testen Sie Codeänderungen ohne Bereitstellung auf AWS
- Kosteneffizienz — Entwickeln und testen, ohne dass Gebühren anfallen AWS
- Offline-Fähigkeit — Arbeiten Sie an Ihren Anwendungen ohne Internetverbindung
- Vereinfachtes Debuggen — Gehen Sie den Lambda-Funktionscode lokal mit einem Debugger durch
- Realistisches Testen — Testen Sie Ihre Anwendungen mithilfe der lokalen Emulation von Diensten AWS

Testen Sie AWS SAM Anwendungen lokal

Informationen zum Testen von Anwendungen, die mithilfe von AWS SAM Vorlagen definiert wurden, finden Sie unter Serverlose Anwendungen testen und debuggen in diesem Handbuch.

Testen Sie Vorlagen lokal CloudFormation

Um die lokalen Testfunktionen AWS SAMCLI von zu verwenden CloudFormation, fügen Sie die AWS SAM Transformation zu Ihrer CloudFormation Vorlage hinzu. Weitere Informationen finden Sie unter Anatomie der AWS SAM Vorlage für:

- AWS SAM Transformation zu CloudFormation Vorlagen hinzufügen
- Grundlegendes zur Kompatibilität von Vorlagen
- Erkunden Sie die Syntax serverloser Ressourcen

Testen Sie AWS CDK-Anwendungen lokal

Sie können den verwenden AWS SAMCLI, um AWS CDK-Anwendungen lokal zu testen, nachdem Sie die CDK-Anwendung mit dem Befehl AWS cdk synth zu CloudFormation Vorlagen synthetisiert haben.

Weitere Informationen finden Sie im CDK-Entwicklerhandbuch unter Lokales Testen und Erstellen von AWS CDK-Anwendungen mit dem AWS SAMCLI. AWS Dies umfasst:

- Erste Schritte mit lokalem Testen
- Lokales Testen von AWS CDK-Anwendungen mit AWS SAM

Testen Sie Terraform-Anwendungen lokal

Das AWS SAMCLI unterstützt Terraform-Projekte und Terraform Cloud. Sie können es verwenden, um lokales Debuggen und Testen von: Lambda-Funktionen und -Layern sowie dem Amazon API Gateway HTTP und REST durchzuführen. APIs

Informationen zur Einrichtung Ihrer Umgebung und zu allen verfügbaren Funktionen finden Sie unter Terraform-Support in diesem Handbuch. Dies umfasst:

- Erste Schritte mit der Terraform-Unterstützung für AWS SAMCLI
- Verwendung von AWS SAMCLI mit Terraform zum lokalen Debuggen und Testen

AWS SAM Vorlage

Nachdem Sie den sam init Befehl ausgeführt und den nachfolgenden Workflow abgeschlossen haben, AWS SAM erstellt das Projektverzeichnis Ihrer Anwendung, bei dem es sich um Ihr AWS SAM Projekt handelt. Sie definieren Ihre serverlose Anwendung, indem Sie Ihrem AWS SAM Projekt Code hinzufügen. Ihr AWS SAM Projekt besteht zwar aus einer Reihe von Dateien und Ordnern, aber die Datei, mit der Sie hauptsächlich arbeiten, ist Ihre AWS SAM Vorlage (benannt`template.yaml`). In dieser Vorlage schreiben Sie den Code, um Ressourcen, Ereignisquellenzuordnungen und andere Eigenschaften auszudrücken, die Ihre serverlose Anwendung definieren.

> **Note**
>
> Ein Schlüsselelement der AWS SAM Vorlage ist die Vorlagenspezifikation. AWS SAM Diese Spezifikation bietet die Kurzsyntax, mit der Sie im Vergleich zu CloudFormation weniger Codezeilen die Ressourcen, Zuordnungen von Ereignisquellen, Berechtigungen, APIs und anderen Eigenschaften Ihrer serverlosen Anwendung definieren können.

Dieser Abschnitt enthält Einzelheiten darüber, wie Sie Abschnitte in der AWS SAM Vorlage verwenden, um Ressourcentypen, Ressourceneigenschaften, Datentypen, Ressourcenattribute, systeminterne Funktionen und API-Gateway-Erweiterungen zu definieren.

AWS SAM Vorlagen sind eine Erweiterung von CloudFormation Vorlagen mit eindeutigen Syntaxtypen, die Kurzsyntax mit weniger Codezeilen als verwenden. CloudFormation Dies beschleunigt Ihre Entwicklung beim Erstellen einer serverlosen Anwendung. Weitere Informationen finden Sie unter AWS SAM Ressourcen und Immobilien. Die vollständige Referenz zu CloudFormation Vorlagen finden Sie unter CloudFormation Vorlagenreferenz im AWS CloudFormation Benutzerhandbuch.

Bei der Entwicklung wird es für Sie oft von Vorteil sein, Ihren Anwendungscode in separate Dateien aufzuteilen, um Ihre Anwendung besser organisieren und verwalten zu können. Ein einfaches Beispiel hierfür ist die Verwendung einer separaten Datei für Ihren AWS Lambda Funktionscode, anstatt diesen Code in Ihrer AWS SAM Vorlage zu haben. Organisieren Sie dazu Ihren Lambda-Funktionscode in einem Unterverzeichnis Ihres Projekts und verweisen Sie auf seinen lokalen Pfad in Ihrer AWS Serverless Application Model ()AWS SAM-Vorlage.

Themen

- AWS SAM Anatomie der Vorlage
- AWS SAM Ressourcen und Immobilien
- Generiert CloudFormation Ressourcen für AWS SAM
- Ressourcenattribute, unterstützt von AWS SAM
- API-Gateway-Erweiterungen für AWS SAM
- Intrinsische Funktionen für AWS SAM
- CloudFormation Unterstützung für Spracherweiterungen

AWS SAM Anatomie der Vorlage

Eine AWS SAM Vorlagendatei folgt weitgehend dem Format einer CloudFormation Vorlagendatei, das im AWS CloudFormation Benutzerhandbuch unter Anatomie der Vorlage beschrieben wird. Die Hauptunterschiede zwischen AWS SAM Vorlagendateien und CloudFormation Vorlagendateien sind die folgenden:

- Deklaration transformieren. Die Deklaration `Transform: AWS::Serverless-2016-10-31` ist für AWS SAM Vorlagendateien erforderlich. Diese Deklaration identifiziert eine CloudFormation Vorlagendatei als AWS SAM Vorlagendatei. Weitere Informationen zu Transformationen finden Sie unter Transform im AWS CloudFormation Benutzerhandbuch.
- Abschnitt „Globals“. Der `Globals` Abschnitt ist einzigartig für AWS SAM. Er definiert Eigenschaften, die allen Ihren serverlosen Funktionen gemeinsam sind, und APIs. Alle `AWS::Serverless::StateMachine` Ressourcen`AWS::Serverless::Function`,`AWS::Serverless::Api`,`AWS::Serverless::Capacit` `AWS::Serverless::HttpApi`AWS::Serverless::SimpleTable, und erben die Eigenschaften, die `Globals` im Abschnitt definiert sind. Weitere Informationen zu diesem Abschnitt finden Sie unterSektion Globales der AWS SAM Vorlage.
- Abschnitt „Ressourcen“. In AWS SAM Vorlagen kann der `Resources` Abschnitt eine Kombination aus CloudFormation Ressourcen und AWS SAM Ressourcen enthalten. Weitere Informationen zu CloudFormation Ressourcen finden Sie in der Referenz zu AWS Ressourcen- und Eigenschaftstypen im AWS CloudFormation Benutzerhandbuch. Weitere Informationen zu AWS SAM Ressourcen finden Sie unterAWS SAM Ressourcen und Immobilien.

Alle anderen Abschnitte einer AWS SAM Vorlagendatei entsprechen dem gleichnamigen Abschnitt der CloudFormation Vorlagendatei.

YAML

Das folgende Beispiel zeigt ein in YAML formatiertes Vorlagenfragment.

```
Transform: AWS::Serverless-2016-10-31

Globals:
  set of globals

Description:
  String

Metadata:
  template metadata

Parameters:
  set of parameters

Mappings:
  set of mappings

Conditions:
  set of conditions

Resources:
  set of resources

Outputs:
  set of outputs
```

Abschnitte einer Vorlage

AWS SAM Vorlagen können mehrere Hauptabschnitte enthalten. Nur die `Resources` Abschnitte `Transform` und sind erforderlich.

Sie können die Vorlagenabschnitte in beliebiger Reihenfolge einfügen. Wenn Sie jedoch Spracherweiterungen verwenden, sollten Sie `AWS::LanguageExtensions` sie vor der serverlosen Transformation (also vor`AWS::Serverless-2016-10-31`) hinzufügen, wie im folgenden Beispiel gezeigt:

```
Transform:
  - AWS::LanguageExtensions
```

```
  - AWS::Serverless-2016-10-31
```

Beim Erstellen Ihrer Vorlage kann es hilfreich sein, die logische Reihenfolge zu verwenden, die in der folgenden Liste aufgeführt ist. Das liegt daran, dass sich die Werte in einem Abschnitt möglicherweise auf Werte aus einem vorherigen Abschnitt beziehen.

Transformieren (erforderlich)

Bei AWS SAM Vorlagen müssen Sie diesen Abschnitt mit einem Wert von einschließen`AWS::Serverless-2016-10-31`.

Zusätzliche Transformationen sind optional. Weitere Informationen zu Transformationen finden Sie unter Transform im AWS CloudFormation Benutzerhandbuch.

Globale Werte (optional)

Eigenschaften, die all Ihren serverlosen Funktionen und einfachen Tabellen APIs gemeinsam sind. Alle `AWS::Serverless::StateMachine` Ressourcen`AWS::Serverless::Function`,`AWS::Serverless::Api`, `AWS::Serverless::CapacityProvider` `AWS::Serverless::HttpApi``AWS::Serverless::SimpleTable`, und erben die Eigenschaften, die `Globals` im Abschnitt definiert sind.

Dieser Abschnitt ist einzigartig für AWS SAM. In CloudFormation Vorlagen gibt es keinen entsprechenden Abschnitt.

Description (optional)

Gibt eine Textzeichenfolge als Beschreibung der Vorlage an.

Dieser Abschnitt entspricht direkt dem `Description` Abschnitt CloudFormation Vorlagen.

Metadata (optional)

Gibt Objekte an, die zusätzliche Informationen über die Vorlage liefern.

Dieser Abschnitt entspricht direkt dem `Metadata` Abschnitt der CloudFormation Vorlagen.

Parameters (optional)

Werte, die zur Laufzeit an die Vorlage übergeben werden sollen (also bei der Erstellung oder Aktualisierung eines Stacks). Hier können Sie Parameter aus den Vorlagenabschnitten `Resources` und `Outputs` referenzieren. Objekte, die in diesem `Parameters` Abschnitt

deklariert sind, veranlassen den sam deploy --guided Befehl, dem Benutzer zusätzliche Eingabeaufforderungen anzuzeigen.

Werte, die mithilfe des `--parameter-overrides` `sam deploy` Befehlsparameters übergeben werden, und Einträge in der Konfigurationsdatei haben Vorrang vor Einträgen in der Vorlagendatei. AWS SAM Weitere Informationen zu dem `sam deploy` Befehl finden Sie sam deploy in der Befehlsreferenz. AWS SAMCLI Weitere Informationen zur Konfigurationsdatei finden Sie unter AWS SAMCLIKonfigurationsdatei.

Mappings (optional)

Eine Sammlung von Zuweisungen ähnlich einer Lookup-Tabelle, die Schlüssel und die ihnen zugeordneten Werte enthält. Mit ihrer Hilfe lassen sich Werte für bedingte Parameter festlegen. Sie können einen Schlüssel einem entsprechenden Wert zuordnen, indem Sie die `Fn::FindInMap`systeminterne Funktion in den Abschnitten `Resources` und `Outputs` verwenden.

Dieser Abschnitt entspricht direkt dem `Mappings` Abschnitt mit den Vorlagen. CloudFormation

Conditions (optional)

Bedingungen, die steuern, ob während der Erstellung oder Aktualisierung eines Stacks bestimmte Ressourcen erstellt werden oder ein Wert für bestimmte Ressourceneigenschaften festgelegt wird. Beispielsweise könnten Sie über Bedingungen festlegen, wie eine Ressource erstellt wird, wenn der Stack für eine Produktionsumgebung gedacht ist, und wie sie erstellt wird, wenn der Stack für eine Testumgebung gedacht ist.

Dieser Abschnitt entspricht direkt dem `Conditions` Abschnitt der CloudFormation Vorlagen.

Resources (erforderlich)

Die Stack-Ressourcen und ihre Eigenschaften, z. B. eine Amazon Elastic Compute Cloud (Amazon EC2) -Instance oder ein Amazon Simple Storage Service (Amazon S3) -Bucket. Hier können Sie Ressourcen aus den Vorlagenabschnitten `Resources` und `Outputs` referenzieren.

Dieser Abschnitt ähnelt dem `Resources` Abschnitt mit CloudFormation Vorlagen. In AWS SAM Vorlagen kann dieser Abschnitt zusätzlich zu AWS SAM Ressourcen CloudFormation Ressourcen enthalten.

Outputs (optional)

Die Werte, die zurückgegeben werden, wenn Sie sich die Eigenschaften Ihres Stacks ansehen. Sie können beispielsweise eine Ausgabe für einen S3-Bucket-Namen deklarieren und dann den

Befehl `aws cloudformation describe-stacks` AWS Command Line Interface (AWS CLI) aufrufen, um den Namen anzuzeigen.

Dieser Abschnitt entspricht direkt dem `Outputs` Abschnitt der CloudFormation Vorlagen.

Nächste Schritte

Informationen zum Herunterladen und Bereitstellen einer serverlosen Beispielanwendung, die eine AWS SAM Vorlagendatei enthält, finden Erste Schritte mit AWS SAM Sie in Tutorial: Stellen Sie eine Hello World-Anwendung bereit mit AWS SAM den Anweisungen unter.

Sektion Globales der AWS SAM Vorlage

Manchmal haben Ressourcen, die Sie in einer AWS SAM Vorlage deklarieren, gemeinsame Konfigurationen. Beispielsweise könnten Sie eine Anwendung mit mehreren `AWS::Serverless::Function` Ressourcen haben, die identische `RuntimeMemory`, `VPCConfigEnvironment`, und `Cors` Konfigurationen haben. Anstatt diese Informationen in jeder Ressource zu duplizieren, können Sie sie einmal im `Globals` Abschnitt deklarieren und Ihre Ressourcen sie erben lassen.

Der `Globals` Abschnitt unterstützt die folgenden AWS SAM Ressourcentypen:

- `AWS::Serverless::Api`
- `AWS::Serverless::CapacityProvider`
- `AWS::Serverless::Function`
- `AWS::Serverless::HttpApi`
- `AWS::Serverless::SimpleTable`
- `AWS::Serverless::StateMachine`
- `AWS::Serverless::MicrovmImage`
- `AWS::Serverless::NetworkConnector`

Beispiel:

```
Globals:
  Function:
    Runtime: nodejs12.x
    Timeout: 180
```

```
    Handler: index.handler
    Environment:
      Variables:
        TABLE_NAME: data-table

Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      Environment:
        Variables:
          MESSAGE: "Hello From SAM"

  ThumbnailFunction:
    Type: AWS::Serverless::Function
    Properties:
      Events:
        Thumbnail:
          Type: Api
          Properties:
            Path: /thumbnail
            Method: POST
```

In diesem Beispiel `ThumbnailFunction` verwenden beide `HelloWorldFunction` und „nodejs12.x“ für`Runtime`, „180" Sekunden für und „index.handler“ für`Timeout`. `Handler` `HelloWorldFunction`fügt die MESSAGE-Umgebungsvariable zusätzlich zum geerbten TABLE_NAME hinzu. `ThumbnailFunction`erbt alle `Globals` Eigenschaften und fügt eine API-Ereignisquelle hinzu.

Unterstützte Ressourcen und Eigenschaften

AWS SAM unterstützt die folgenden Ressourcen und Eigenschaften.

```
Globals:
  Api:
    AccessLogSetting:
    Auth:
    BinaryMediaTypes:
    CacheClusterEnabled:
    CacheClusterSize:
    CanarySetting:
    Cors:
    DefinitionUri:
```

```
    Domain:
    EndpointConfiguration:
    EndpointAccessMode:
    GatewayResponses:
    MethodSettings:
    MinimumCompressionSize:
    Name:
    OpenApiVersion:
    PropagateTags:
    SecurityPolicy:
    TracingEnabled:
    Variables:

  CapacityProvider:
    InstanceRequirements:
    KmsKeyArn:
    LoggingConfig:
    ManagedResourceTags:
    OperatorRole:
    PropagateTags:
    ScalingConfig:
    Tags:
    VpcConfig:

  Function:
    Architectures:
    AssumeRolePolicyDocument:
    AutoPublishAlias:
    AutoPublishAliasAllProperties:
    CapacityProviderConfig:
    CodeSigningConfigArn:
    CodeUri:
    DeadLetterQueue:
    DeploymentPreference:
    Description:
    DurableConfig:
    Environment:
    EphemeralStorage:
    EventInvokeConfig:
    FileSystemConfigs:
    FunctionScalingConfig:
    FunctionUrlConfig:
    Handler:
    KmsKeyArn:
```

```
    Layers:
    LoggingConfig:
    MemorySize:
    PermissionsBoundary:
    PropagateTags:
    ProvisionedConcurrencyConfig:
    PublishToLatestPublished:
    RecursiveLoop:
    ReservedConcurrentExecutions:
    RolePath:
    Runtime:
    RuntimeManagementConfig:
    SnapStart:
    SourceKMSKeyArn:
    Tags:
    TenancyConfig:
    Timeout:
    Tracing:
    VersionDeletionPolicy:
    VpcConfig:

  HttpApi:
    AccessLogSettings:
    Auth:
    PropagateTags:
    StageVariables:
    Tags:

  SimpleTable:
    SSESpecification:

  StateMachine:
    PropagateTags:

  MicrovmImage:
    BaseImageArn:
    BaseImageVersion:
    BuildRoleArn:
    Hooks:
    EnvironmentVariables:
    EgressNetworkConnectors:
    CpuConfigurations:
    Resources:
    AdditionalOsCapabilities:
```

```
    Logging:
    Tags:
    PropagateTags:

  NetworkConnector:
    OperatorRole:
    Tags:
    PropagateTags:
```

> **Note**
>
> Alle Ressourcen und Eigenschaften, die nicht in der vorherigen Liste enthalten sind, werden nicht unterstützt. Sie werden u. a. aus folgenden Gründen nicht unterstützt: 1) Sie führen zu potenziellen Sicherheitsproblemen oder 2) Sie erschweren das Verständnis der Vorlage.

Implizite APIs

AWS SAM erstellt implizite APIs, wenn Sie eine API im Abschnitt deklarieren. `Events` Sie können es verwenden`Globals`, um alle Eigenschaften impliziter APIs zu überschreiben.

Überschreibbare Eigenschaften

Ressourcen können die Eigenschaften überschreiben, die Sie in dem `Globals` Abschnitt deklarieren. Sie können beispielsweise einer Umgebungsvariablenzuordnung neue Variablen hinzufügen oder global deklarierte Variablen überschreiben. Die Ressource kann jedoch keine Eigenschaft entfernen, die in dem `Globals` Abschnitt angegeben ist.

Allgemeiner ausgedrückt deklariert der `Globals` Abschnitt Eigenschaften, die alle Ihre Ressourcen gemeinsam haben. Einige Ressourcen können neue Werte für global deklarierte Eigenschaften bereitstellen, sie können sie jedoch nicht entfernen. Wenn einige Ressourcen eine Eigenschaft verwenden, andere jedoch nicht, dürfen Sie sie nicht im `Globals` Abschnitt deklarieren.

In den folgenden Abschnitten wird beschrieben, wie das Überschreiben für verschiedene Datentypen funktioniert.

Primitive Datentypen werden ersetzt

Zu den primitiven Datentypen gehören Zeichenketten, Zahlen, Boolesche Werte usw.

Der im `Resources` Abschnitt angegebene Wert ersetzt den Wert im `Globals` Abschnitt.

Beispiel:

```
Globals:
  Function:
    Runtime: nodejs12.x

Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      Runtime: python3.9
```

Das `Runtime` Formular `MyFunction` ist auf gesetzt`python3.9`.

Karten werden zusammengeführt

Karten werden auch als Wörterbücher oder Sammlungen von Schlüssel-Wert-Paaren bezeichnet.

Karteneinträge in `Resources` diesem Abschnitt werden mit globalen Karteneinträgen zusammengeführt. Wenn es Duplikate gibt, überschreibt der `Resource` Abschnittseintrag den `Globals` Abschnittseintrag.

Beispiel:

```
Globals:
  Function:
    Environment:
      Variables:
        STAGE: Production
        TABLE_NAME: global-table

Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      Environment:
        Variables:
          TABLE_NAME: resource-table
          NEW_VAR: hello
```

Die Umgebungsvariablen von `MyFunction` sind auf Folgendes festgelegt:

```
{
  "STAGE": "Production",
  "TABLE_NAME": "resource-table",
  "NEW_VAR": "hello"
}
```

Listen sind additiv

Listen werden auch als Arrays bezeichnet.

Listeneinträge im `Globals` Abschnitt werden der Liste im Abschnitt vorangestellt. `Resources`

Beispiel:

```
Globals:
  Function:
    VpcConfig:
      SecurityGroupIds:
        - sg-123
        - sg-456

Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      VpcConfig:
        SecurityGroupIds:
          - sg-first
```

Die `SecurityGroupIds` Felder `MyFunction` `VpcConfig` sind wie folgt gesetzt:

```
[ "sg-123", "sg-456", "sg-first" ]
```

AWS SAM Ressourcen und Immobilien

In diesem Abschnitt werden die Ressourcen- und Eigenschaftstypen beschrieben, die spezifisch für sind. AWS SAM Sie definieren diese Ressourcen und Eigenschaften mithilfe der AWS SAM Kurzsyntax. AWS SAM unterstützt auch CloudFormation Ressourcen- und Eigenschaftstypen. Referenzinformationen zu allen AWS Ressourcen- und Eigenschaftstypen CloudFormation

sowie deren AWS SAM Unterstützung finden Sie in der Referenz zu AWS Ressourcen- und Eigenschaftstypen im AWS CloudFormation Benutzerhandbuch.

Themen

- AWS::Serverless::Api
- AWS::Serverless::Application
- AWS::Serverless::CapacityProvider
- AWS::Serverless::Connector
- AWS::Serverless: :Funktion
- AWS::Serverless::GraphQLApi
- AWS::Serverless::HttpApi
- AWS::Serverless::WebSocketApi
- AWS::Serverless::LayerVersion
- AWS::Serverless::MicrovmImage
- AWS::Serverless::NetworkConnector
- AWS::Serverless::SimpleTable
- AWS::Serverless::StateMachine

AWS::Serverless::Api

Erstellt eine Sammlung von Amazon API Gateway Gateway-Ressourcen und -Methoden, die über HTTPS-Endpunkte aufgerufen werden können.

Eine AWS::Serverless::Api Ressource muss nicht explizit zu einer Vorlage für AWS serverlose Anwendungsdefinitionen hinzugefügt werden. Eine Ressource dieses Typs wird implizit aus der Vereinigung von API-Ereignissen erstellt, die für AWS::Serverless: :Funktion Ressourcen definiert sind, die in der Vorlage definiert sind und sich nicht auf eine AWS::Serverless::Api Ressource beziehen.

Eine AWS::Serverless::Api Ressource sollte verwendet werden, um die verwendete API zu definieren und zu dokumentieren OpenApi, was mehr Möglichkeiten bietet, die zugrunde liegenden Amazon API Gateway Gateway-Ressourcen zu konfigurieren.

Wir empfehlen, CloudFormation Hooks oder IAM-Richtlinien zu verwenden, um zu überprüfen, ob an API-Gateway-Ressourcen Autorisatoren angehängt sind, um den Zugriff darauf zu kontrollieren.

Weitere Informationen zur Verwendung von CloudFormation Hooks finden Sie unter Hooks registrieren im CloudFormation CLI-Benutzerhandbuch und im apigw-enforce-authorizer GitHub Repository.

Weitere Informationen zur Verwendung von IAM-Richtlinien finden Sie unter Erfordern, dass API-Routen autorisiert sind im API Gateway Developer Guide.

> **Note**
>
> Bei der Bereitstellung auf werden AWS CloudFormation Ihre AWS SAM Ressourcen in Ressourcen umgewandelt CloudFormation . AWS SAM Weitere Informationen finden Sie unter Generiert CloudFormation Ressourcen für AWS SAM.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer AWS Serverless Application Model (AWS SAM) -Vorlage zu deklarieren.

YAML

```
Type: AWS::Serverless::Api
Properties:
  AccessLogSetting: AccessLogSetting
  AlwaysDeploy: Boolean
  ApiKeySourceType: String
  Auth: ApiAuth
  BinaryMediaTypes: List
  CacheClusterEnabled: Boolean
  CacheClusterSize: String
  CanarySetting: CanarySetting
  Cors: String | CorsConfiguration
  DefinitionBody: JSON
  DefinitionUri: String | ApiDefinition
  Description: String
  DisableExecuteApiEndpoint: Boolean
  Domain: DomainConfiguration
  EndpointAccessMode: String
  EndpointConfiguration: EndpointConfiguration
  FailOnWarnings: Boolean
  GatewayResponses: Map
  MergeDefinitions: Boolean
```

```
  MethodSettings: MethodSettings
  MinimumCompressionSize: Integer
  Mode: String
  Models: Map
  Name: String
  OpenApiVersion: String
  PropagateTags: Boolean
  Policy: JSON
  SecurityPolicy: String
  StageName: String
  Tags: Map
  TracingEnabled: Boolean
  Variables: Map
```

Eigenschaften

`AccessLogSetting`

Konfiguriert die Einstellung für das Zugriffsprotokoll für eine Phase.

Typ: AccessLogSetting

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `AccessLogSetting` Eigenschaft einer `AWS::ApiGateway::Stage` Ressource übergeben.

`AlwaysDeploy`

Stellt die API immer bereit, auch wenn keine Änderungen an der API festgestellt wurden.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`ApiKeySourceType`

Die Quelle des API-Schlüssels für Messungsanforderungen nach einem Nutzungsplan. Gültige Werte sind `HEADER` und `AUTHORIZER`.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `ApiKeySourceType` Eigenschaft einer `AWS::ApiGateway::RestApi` Ressource übergeben.

`Auth`

Konfigurieren Sie die Autorisierung, um den Zugriff auf Ihre API-Gateway-API zu kontrollieren.

Weitere Informationen zur Konfiguration des Zugriffs mithilfe von AWS SAM finden Sie unterKontrollieren Sie den API-Zugriff mit Ihrer AWS SAM Vorlage. Ein Beispiel, das zeigt, wie Sie einen globalen Autorisierer überschreiben können, finden Sie unterthe section called "Überschreiben Sie einen globalen Autorisierer für Ihre Amazon API Gateway Gateway-REST-API".

Typ: ApiAuth

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`BinaryMediaTypes`

Liste der MIME-Typen, die Ihre API zurückgeben könnte. Verwenden Sie dies, um die Binärunterstützung für zu aktivierenAPIs.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `BinaryMediaTypes` Eigenschaft einer `AWS::ApiGateway::RestApi` Ressource. Die Liste von BinaryMediaTypes wird sowohl der CloudFormation Ressource als auch dem OpenAPI-Dokument hinzugefügt.

`CacheClusterEnabled`

Gibt an, ob das Caching für die Phase aktiviert ist. Um Antworten zwischenzuspeichern, müssen Sie auch `CachingEnabled` auf `true` unter `MethodSettings` setzen.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `CacheClusterEnabled` Eigenschaft einer `AWS::ApiGateway::Stage` Ressource übergeben.

`CacheClusterSize`

Die Cache-Cluster-Größe der Stufe.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `CacheClusterSize` Eigenschaft einer `AWS::ApiGateway::Stage` Ressource übergeben.

`CanarySetting`

Konfigurieren Sie eine Canary-Einstellung für eine Phase einer regulären Bereitstellung.

Typ: CanarySetting

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `CanarySetting` Eigenschaft einer `AWS::ApiGateway::Stage` Ressource übergeben.

`Cors`

Verwalten Sie Cross-Origin Resource Sharing (CORS) für Ihr gesamtes API Gateway. APIs Geben Sie die Domäne, die zugelassen werden soll, als Zeichenfolge an oder geben Sie ein Wörterbuch mit zusätzlicher Cors-Konfiguration an.

> **Note**
>
> CORS erfordert eine AWS SAM Änderung Ihrer OpenAPI-Definition. Erstellen Sie eine Inline-OpenAPI-Definition in`DefinitionBody`, um CORS zu aktivieren.

Weitere Informationen zu CORS finden Sie unter Aktivieren von CORS für eine API-Gateway-REST-API-Ressource im API Gateway Developer Guide.

Typ: Zeichenfolge | CorsConfiguration

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`DefinitionBody`

OpenAPI-Spezifikation, die Ihre API beschreibt. Wenn `DefinitionUri` weder noch angegeben `DefinitionBody` sind, generiert SAM auf der Grundlage Ihrer Vorlagenkonfiguration eine `DefinitionBody` für Sie.

Verwenden Sie die `AWS::Include` Transformation, um auf eine lokale OpenAPI Datei zu verweisen, die Ihre API definiert. Weitere Informationen hierzu finden Sie unter Wie werden AWS SAM lokale Dateien hochgeladen.

Type: JSON

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der Body Eigenschaft einer `AWS::ApiGateway::RestApi` Ressource. Wenn bestimmte Eigenschaften bereitgestellt werden, können Inhalte in die eingefügt oder geändert werden, DefinitionBody bevor sie an sie übergeben werden CloudFormation. Zu den Eigenschaften gehören `Auth` `BinaryMediaTypesCors`,`GatewayResponses`,`Models`, und eine `EventSource` vom Typ Api für eine entsprechende`AWS::Serverless::Function`.

`DefinitionUri`

Amazon S3 S3-URI, lokaler Dateipfad oder Speicherortobjekt des OpenAPI-Dokuments, das die API definiert. Das Amazon S3 S3-Objekt, auf das diese Eigenschaft verweist, muss eine gültige OpenAPI-Datei sein. Wenn `DefinitionUri` weder noch angegeben `DefinitionBody` sind, generiert SAM auf der Grundlage Ihrer Vorlagenkonfiguration eine `DefinitionBody` für Sie.

Wenn ein lokaler Dateipfad angegeben wird, muss die Vorlage den Workflow durchlaufen, der den `sam package` Befehl `sam deploy` oder enthält, damit die Definition ordnungsgemäß transformiert wird.

Systeminterne Funktionen werden in externen OpenApi Dateien, auf die von verwiesen wird, nicht unterstützt. `DefinitionUri` Verwenden Sie stattdessen die `DefinitionBody` Eigenschaft mit der Include-Transformation, um eine OpenApi Definition in die Vorlage zu importieren.

Typ: Zeichenfolge | ApiDefinition

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `BodyS3Location` Eigenschaft einer `AWS::ApiGateway::RestApi` Ressource. Die verschachtelten Amazon S3 S3-Eigenschaften sind unterschiedlich benannt.

`Description`

Eine Beschreibung der Api-Ressource.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Description` Eigenschaft einer `AWS::ApiGateway::RestApi` Ressource übergeben.

`DisableExecuteApiEndpoint`

Gibt an, ob Clients Ihre API mithilfe des `execute-api`-Standardendpunkts aufrufen können. Standardmäßig können Clients Ihre API mit der Standardeinstellung `https://{api_id}.execute-api.{region}.amazonaws.com` aufrufen. Wenn Sie erzwingen möchten, dass Kunden einen benutzerdefinierten Domänennamen verwenden, um Ihre API aufzurufen, geben Sie a `True`.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `DisableExecuteApiEndpoint` Eigenschaft einer `AWS::ApiGateway::RestApi` Ressource. Sie wird direkt an die `disableExecuteApiEndpoint` Eigenschaft einer `x-amazon-apigateway-endpoint-configuration` Erweiterung übergeben, die der `Body` Eigenschaft einer `AWS::ApiGateway::RestApi` Ressource hinzugefügt wird.

`Domain`

Konfiguriert eine benutzerdefinierte Domain für diese API-Gateway-API.

Typ: DomainConfiguration

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`EndpointConfiguration`

Der Endpunkttyp einer REST-API.

Typ: EndpointConfiguration

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `EndpointConfiguration` Eigenschaft einer `AWS::ApiGateway::RestApi` Ressource. Die verschachtelten Konfigurationseigenschaften sind unterschiedlich benannt.

`EndpointAccessMode`

Der Zugriffsmodus für den REST-API-Endpunkt. Erforderlich, wenn erweiterte Sicherheitsrichtlinien (solche mit dem Präfix`SecurityPolicy_`) verwendet werden.

Zulässige Werte: `STRICT` oder `BASIC`.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `EndpointAccessMode` Eigenschaft einer `AWS::ApiGateway::RestApi` Ressource übergeben.

`FailOnWarnings`

Gibt an, ob die API-Erstellung rückgängig gemacht werden soll (`true`) oder nicht (`false`), wenn eine Warnung auftritt. Der Standardwert ist `false`.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `FailOnWarnings` Eigenschaft einer `AWS::ApiGateway::RestApi` Ressource übergeben.

`GatewayResponses`

Konfiguriert Gateway-Antworten für eine API. Gateway-Antworten sind Antworten, die von API Gateway entweder direkt oder mithilfe von Lambda-Autorisierern zurückgegeben werden. Weitere Informationen finden Sie in der Dokumentation zur Api OpenApi Gateway-Erweiterung für Gateway Responses.

Typ: Karte

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`MergeDefinitions`

AWS SAM generiert eine OpenAPI Spezifikation aus Ihrer API-Ereignisquelle. Geben Sie `true` an, dass dies mit der in Ihrer `AWS::Serverless::Api` Ressource definierten OpenAPI Inline-Spezifikation AWS SAM zusammengeführt werden soll. Geben Sie `false` an, dass nicht zusammengeführt werden soll.

`MergeDefinitions`erfordert, dass die `DefinitionBody` Eigenschaft `AWS::Serverless::Api` für definiert wird. `MergeDefinitions`ist nicht kompatibel mit der `DefinitionUri` Eigenschaft für`AWS::Serverless::Api`.

Standardwert: `false`

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`MethodSettings`

Konfiguriert alle Einstellungen für die API-Phase, einschließlich Logging, Metrics, CacheTTL und Throttling.

Typ: Liste von MethodSetting

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `MethodSettings` Eigenschaft einer Ressource übergeben. `AWS::ApiGateway::Stage`

`MinimumCompressionSize`

Erlaubt die Komprimierung von Antworttexten auf der Grundlage des Accept-Encoding-Headers des Clients. Die Komprimierung wird ausgelöst, wenn die Größe des Antworttextes größer oder gleich dem konfigurierten Schwellenwert ist. Der maximale Schwellenwert für die Körpergröße beträgt 10 MB (10.485.760 Byte). - Die folgenden Komprimierungstypen werden unterstützt: gzip, deflate und identity.

Typ: Ganzzahl

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `MinimumCompressionSize` Eigenschaft einer Ressource übergeben. `AWS::ApiGateway::RestApi`

`Mode`

Diese Eigenschaft gilt nur, wenn Sie OpenAPI zur Definition Ihrer REST-API verwenden. Der `Mode` bestimmt, wie API Gateway mit Ressourcen-Updates umgeht. Weitere Informationen finden Sie unter Mode-Eigenschaft des AWS::ApiGateway::RestApiRessourcentyps.

Zulässige Werte: `overwrite` oder `merge`.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Mode` Eigenschaft einer `AWS::ApiGateway::RestApi` Ressource übergeben.

`Models`

Die Schemas, die von Ihren API-Methoden verwendet werden sollen. Diese Schemas können mit JSON oder YAML beschrieben werden. Im Abschnitt Beispiele unten auf dieser Seite finden Sie Beispielmodelle.

Typ: Karte

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Name`

Ein Name für die RestApi API-Gateway-Ressource

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Name` Eigenschaft einer `AWS::ApiGateway::RestApi` Ressource übergeben.

`OpenApiVersion`

Version von OpenApi , die verwendet werden soll. Dies kann entweder `2.0` für die Swagger-Spezifikation oder für eine der OpenApi 3.0-Versionen sein. `3.0.1` Weitere Informationen zu OpenAPI finden Sie in der OpenAPI-Spezifikation.

> **Note**
>
> AWS SAM erstellt eine Stage, die `Stage` standardmäßig aufgerufen wird. Wenn Sie diese Eigenschaft auf einen beliebigen gültigen Wert setzen, wird die Erstellung der Phase verhindert`Stage`.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`PropagateTags`

Geben Sie an, ob Tags von der `Tags` Eigenschaft an Ihre AWS::Serverless::Api generierten Ressourcen weitergegeben werden sollen oder nicht. Geben Sie `True` an, dass Tags in Ihren generierten Ressourcen verbreitet werden sollen.

Typ: Boolesch

Required: No

Standardwert: `False`

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Policy`

Ein Richtliniendokument, das die Berechtigungen für die API enthält. Um den ARN für die Richtlinie festzulegen, verwenden Sie die intrinsische `!Join`-Funktion mit `""` als Trennzeichen und die Werte von `"execute-api:/"` und `"*"`.

Type: JSON

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die Policy-Eigenschaft einer `AWS::ApiGateway::RestApi` Ressource übergeben.

`SecurityPolicy`

Die TLS-Version plus die Cipher Suite für die REST-API. Wenn Sie erweiterte Sicherheitsrichtlinien (solche mit dem Präfix`SecurityPolicy>`) verwenden, müssen Sie auch die festlegen. `EndpointAccessMode`

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `SecurityPolicy` Eigenschaft einer `AWS::ApiGateway::RestApi` Ressource übergeben.

`StageName`

Der Name der Stufe, die API Gateway als erstes Pfadsegment im Aufruf-URI (Uniform Resource Identifier) verwendet.

Um auf die Staging-Ressource zu verweisen, verwenden Sie*`<api-logical-id>`*`.Stage`. Weitere Hinweise zum Verweisen auf Ressourcen, die bei der Angabe einer AWS::Serverless::Api Ressource generiert werden, finden Sie unterCloudFormation Ressourcen, die generiert werdenAWS::Serverless::Api, wenn angegeben. Allgemeine Informationen zu generierten CloudFormation Ressourcen finden Sie unterGeneriert CloudFormation Ressourcen für AWS SAM.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `StageName` Eigenschaft einer `AWS::ApiGateway::Stage` Ressource. Es ist in SAM erforderlich, aber nicht in API Gateway

Zusätzliche Hinweise: Die implizite API hat den Stagennamen „Prod“.

`Tags`

Eine Zuordnung (Zeichenfolge zu Zeichenfolge), die die Tags angibt, die zu diesem API-Gateway-Schritt hinzugefügt werden sollen. Einzelheiten zu gültigen Schlüsseln und Werten für Tags finden Sie unter Resource-Tag im AWS CloudFormation Benutzerhandbuch.

Typ: Karte

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `Tags` Eigenschaft einer `AWS::ApiGateway::Stage` Ressource. Die Tags-Eigenschaft in SAM besteht aus Key:Value-Paaren; darin besteht CloudFormation sie aus einer Liste von Tag-Objekten.

`TracingEnabled`

Zeigt an, ob aktives Tracing mit X-Ray für die Phase aktiviert ist. Weitere Informationen zu X-Ray finden Sie unter Tracing user requests to REST APIs using X-Ray im API Gateway Developer Guide.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `TracingEnabled` Eigenschaft einer `AWS::ApiGateway::Stage` Ressource übergeben.

`Variables`

Eine Zuordnung (Zeichenfolge zu Zeichenfolge), die die Stufenvariablen definiert, wobei der Variablenname der Schlüssel und der Variablenwert der Wert ist. Variablennamen sind auf alphanumerische Zeichen beschränkt. Werte müssen dem folgenden regulären Ausdruck entsprechen: `[A-Za-z0-9._~:/?#&=,-]+`.

Typ: Karte

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Variables` Eigenschaft einer `AWS::ApiGateway::Stage` Ressource übergeben.

Rückgabewerte

Punkt

Wenn die logische ID dieser Ressource für die `Ref` intrinsische Funktion bereitgestellt wird, gibt sie die ID der zugrunde liegenden API-Gateway-API zurück.

Weitere Informationen zur Verwendung der `Ref` Funktion finden Sie `Ref`im AWS CloudFormation Benutzerhandbuch.

Fn:: GetAtt

`Fn::GetAtt` gibt einen Wert für ein angegebenes Attribut dieses Typs zurück. Im Folgenden sehen Sie die verfügbaren Attribute und Beispielrückgabewerte.

Weitere Informationen zur Verwendung `Fn::GetAtt` finden Sie `Fn::GetAtt`im AWS CloudFormation Benutzerhandbuch.

`RootResourceId`

Die Stamm-Ressourcen-ID für eine `RestApi`-Ressource, z. B. `a0bc123d4e`.

Beispiele

SimpleApiExample

Eine Hello AWS SAM World-Vorlagendatei, die eine Lambda-Funktion mit einem API-Endpunkt enthält. Dies ist eine vollständige AWS SAM Vorlagendatei für eine funktionierende serverlose Anwendung.

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: AWS SAM template with a simple API definition
Resources:
  ApiGatewayApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: prod
  ApiFunction: # Adds a GET method at the root resource via an Api event
    Type: AWS::Serverless::Function
    Properties:
      Events:
        ApiEvent:
          Type: Api
          Properties:
            Path: /
            Method: get
            RestApiId:
```

```
              Ref: ApiGatewayApi
      Runtime: python3.10
      Handler: index.handler
      InlineCode: |
        def handler(event, context):
            return {'body': 'Hello World!', 'statusCode': 200}
```

ApiCorsExample

Ein AWS SAM Vorlagenausschnitt mit einer API, die in einer externen Swagger-Datei definiert ist, zusammen mit Lambda-Integrationen und CORS-Konfigurationen. Dies ist nur ein Teil einer Vorlagendatei, der eine Definition zeigt. AWS SAM AWS::Serverless::Api

YAML

```
Resources:
  ApiGatewayApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      # Allows www.example.com to call these APIs
      # SAM will automatically add AllowMethods with a list of methods for this API
      Cors: "'www.example.com'"
      DefinitionBody: # Pull in an OpenApi definition from S3
        'Fn::Transform':
          Name: 'AWS::Include'
          # Replace "bucket" with your bucket name
          Parameters:
            Location: s3://bucket/swagger.yaml
```

ApiCognitoAuthExample

Ein AWS SAM Vorlagenausschnitt mit einer API, die Amazon Cognito verwendet, um Anfragen an die API zu autorisieren. Dies ist nur ein Teil einer AWS SAM Vorlagendatei, die eine Definition enthält. AWS::Serverless::Api

YAML

```
Resources:
  ApiGatewayApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
```

```
      Cors: "'*'"
      Auth:
        DefaultAuthorizer: MyCognitoAuthorizer
        Authorizers:
          MyCognitoAuthorizer:
            UserPoolArn:
              Fn::GetAtt: [MyCognitoUserPool, Arn]
```

ApiModelsExample

Ein AWS SAM Vorlagenausschnitt mit einer API, die ein Models-Schema enthält. Dies ist nur ein Teil einer AWS SAM Vorlagendatei, die eine AWS::Serverless::Api Definition mit zwei Modellschemas zeigt.

YAML

```
Resources:
  ApiGatewayApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      Models:
        User:
          type: object
          required:
            - username
            - employee_id
          properties:
            username:
              type: string
            employee_id:
              type: integer
            department:
              type: string
        Item:
          type: object
          properties:
            count:
              type: integer
            category:
              type: string
            price:
              type: integer
```

Beispiel für Caching

Eine Hello AWS SAM World-Vorlagendatei, die eine Lambda-Funktion mit einem API-Endpunkt enthält. In der API ist das Caching für eine Ressource und Methode aktiviert. Weitere Informationen zum Caching finden Sie unter Aktivieren von API-Caching zur Verbesserung der Reaktionsfähigkeit im API Gateway Developer Guide.

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: AWS SAM template with a simple API definition with caching turned on
Resources:
  ApiGatewayApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: prod
      CacheClusterEnabled: true
      CacheClusterSize: '0.5'
      MethodSettings:
        - ResourcePath: /
          HttpMethod: GET
          CachingEnabled: true
          CacheTtlInSeconds: 300
      Tags:
        CacheMethods: All

  ApiFunction: # Adds a GET method at the root resource via an Api event
    Type: AWS::Serverless::Function
    Properties:
      Events:
        ApiEvent:
          Type: Api
          Properties:
            Path: /
            Method: get
            RestApiId:
              Ref: ApiGatewayApi
      Runtime: python3.10
      Handler: index.handler
      InlineCode: |
        def handler(event, context):
            return {'body': 'Hello World!', 'statusCode': 200}
```

Beispiel für eine benutzerdefinierte Domain mit einer privaten API

Eine Hello AWS SAM World-Vorlagendatei, die eine Lambda-Funktion mit einem API-Endpunkt enthält, der einer privaten Domain zugeordnet ist. Die Vorlage erstellt eine Domänenzugriffszuordnung zwischen einem VPC-Endpunkt und der privaten Domain. Weitere Informationen finden Sie unter Benutzerdefinierte Domainnamen für private APIs in API Gateway.

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: AWS SAM template configured with a custom domain using a private API

Parameters:
    DomainName:
      Type: String
      Default: mydomain.example.com
    CertificateArn:
      Type: String
    HostedZoneId:
      Type: String
    VpcEndpointId:
      Type: String
    VpcEndpointDomainName:
      Type: String
    VpcEndpointHostedZoneId:
      Type: String

Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      InlineCode: |
        def handler(event, context):
            return {'body': 'Hello World!', 'statusCode': 200}
      Handler: index.handler
      Runtime: python3.13
      Events:
        Fetch:
          Type: Api
          Properties:
            RestApiId:
              Ref: MyApi
```

```
          Method: Get
          Path: /get
  MyApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      EndpointConfiguration:
        Type: PRIVATE
        VPCEndpointIds:
        - !Ref VpcEndpointId
      Policy:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal: '*'
          Action: execute-api:Invoke
          Resource: execute-api:/*
        - Effect: Deny
          Principal: '*'
          Action: execute-api:Invoke
          Resource: execute-api:/*
          Condition:
            StringNotEquals:
              aws:SourceVpce: !Ref VpcEndpointId
      Domain:
        DomainName: !Ref DomainName
        CertificateArn: !Ref CertificateArn
        EndpointConfiguration: PRIVATE
        BasePath:
        - /
        Route53:
          HostedZoneId: !Ref HostedZoneId
          VpcEndpointDomainName: !Ref VpcEndpointDomainName
          VpcEndpointHostedZoneId: !Ref VpcEndpointHostedZoneId
        AccessAssociation:
          VpcEndpointId: !Ref VpcEndpointId
        Policy:
          Version: '2012-10-17       '
          Statement:
          - Effect: Allow
            Principal: '*'
            Action: execute-api:Invoke
            Resource: execute-api:/*
          - Effect: Deny
```

```
            Principal: '*'
            Action: execute-api:Invoke
            Resource: execute-api:/*
            Condition:
              StringNotEquals:
                aws:SourceVpce: !Ref VpcEndpointId
```

ApiAuth

Konfigurieren Sie die Autorisierung, um den Zugriff auf Ihre API-Gateway-API zu kontrollieren.

Weitere Informationen und Beispiele für die Konfiguration des Zugriffs mithilfe von AWS SAM finden Sie unterKontrollieren Sie den API-Zugriff mit Ihrer AWS SAM Vorlage.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
  AddApiKeyRequiredToCorsPreflight: Boolean
  AddDefaultAuthorizerToCorsPreflight: Boolean
  ApiKeyRequired: Boolean
  Authorizers: CognitoAuthorizer | LambdaTokenAuthorizer | LambdaRequestAuthorizer |
AWS_IAM
  DefaultAuthorizer: String
  InvokeRole: String
  ResourcePolicy: ResourcePolicyStatement
  UsagePlan: ApiUsagePlan
```

> **Note**
>
> Die `Authorizers` Eigenschaft beinhaltet`AWS_IAM`, aber dafür ist keine zusätzliche Konfiguration erforderlich`AWS_IAM`. Ein Beispiel finden Sie unter AWS ICH BIN.

Eigenschaften

`AddApiKeyRequiredToCorsPreflight`

Wenn die `Cors` Eigenschaften `ApiKeyRequired` und festgelegt sind, führt `AddApiKeyRequiredToCorsPreflight` diese Einstellung dazu, dass der API-Schlüssel der `Options` Eigenschaft hinzugefügt wird.

Typ: Boolesch

Required: No

Standardwert: `True`

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`AddDefaultAuthorizerToCorsPreflight`

Wenn die `Cors` Eigenschaften `DefaultAuthorizer` und gesetzt sind, `AddDefaultAuthorizerToCorsPreflight` führt die Einstellung dazu, dass der Standard-Authorizer zur `Options` Eigenschaft im OpenAPI-Abschnitt hinzugefügt wird.

Typ: Boolesch

Required: No

Standard: Wahr

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`ApiKeyRequired`

Wenn auf true gesetzt, ist ein API-Schlüssel für alle API-Ereignisse erforderlich. Weitere Informationen zu API-Schlüsseln finden Sie unter Nutzungspläne mit API-Schlüsseln erstellen und verwenden im API Gateway Developer Guide.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Authorizers`

Der Autorisierer, der zur Steuerung des Zugriffs auf Ihre API-Gateway-API verwendet wird.

Weitere Informationen finden Sie unter Kontrollieren Sie den API-Zugriff mit Ihrer AWS SAM Vorlage.

Typ: CognitoAuthorizer| | LambdaTokenAuthorizer| LambdaRequestAuthorizer AWS_IAM

Required: No

Standard: Keiner

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Zusätzliche Hinweise: SAM fügt die Authorizer zur OpenApi Definition einer API hinzu.

`DefaultAuthorizer`

Geben Sie einen Standardautorisierer für eine API-Gateway-API an, der standardmäßig für die Autorisierung von API-Aufrufen verwendet wird.

> **Note**
>
> Wenn die API EventSource für die mit dieser API verknüpfte Funktion für die Verwendung von IAM-Berechtigungen konfiguriert ist, muss diese Eigenschaft auf gesetzt werden`AWS_IAM`, da andernfalls ein Fehler auftritt.

Typ: Zeichenfolge

Required: No

Standard: Keiner

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`InvokeRole`

Legt die Integrationsanmeldedaten für alle Ressourcen und Methoden auf diesen Wert fest.

`CALLER_CREDENTIAL`Sordnet zu`arn:aws:iam::*:<user>/`, wodurch die Anmeldeinformationen des Anrufers verwendet werden, um den Endpunkt aufzurufen.

Gültige Werte:`CALLER_CREDENTIALS`,, `NONE` `IAMRoleArn`

Typ: Zeichenfolge

Required: No

Standardwert: `CALLER_CREDENTIALS`

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`ResourcePolicy`

Konfigurieren Sie die Ressourcenrichtlinie für alle Methoden und Pfade auf einer API.

Typ: ResourcePolicyStatement

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Zusätzliche Hinweise: Diese Einstellung kann auch individuell `AWS::Serverless::Function` mit dem definiert werdenApiFunctionAuth. Dies ist für APIs with erforderlich`EndpointConfiguration: PRIVATE`.

`UsagePlan`

Konfiguriert einen mit dieser API verknüpften Nutzungsplan. Weitere Informationen zu Nutzungsplänen finden Sie unter Nutzungspläne mit API-Schlüsseln erstellen und verwenden im API Gateway Developer Guide.

Diese AWS SAM Eigenschaft generiert drei zusätzliche CloudFormation Ressourcen, wenn diese Eigenschaft festgelegt ist: an AWS::ApiGateway::UsagePlanAWS::ApiGateway::UsagePlanKey, an und an AWS::ApiGateway::ApiKey. Informationen zu diesem Szenario finden Sie unterUsagePlan Eigenschaft ist spezifiziert. Allgemeine Informationen zu generierten CloudFormation Ressourcen finden Sie unterGeneriert CloudFormation Ressourcen für AWS SAM.

Typ: ApiUsagePlan

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

CognitoAuth

Beispiel für Cognito Auth

YAML

```
Auth:
  Authorizers:
    MyCognitoAuth:
     UserPoolArn:
       Fn::GetAtt:
         - MyUserPool
         - Arn
     AuthType: "COGNITO_USER_POOLS"
  DefaultAuthorizer: MyCognitoAuth
  InvokeRole: CALLER_CREDENTIALS
  AddDefaultAuthorizerToCorsPreflight: false
  ApiKeyRequired: false
  ResourcePolicy:
    CustomStatements: [{
      "Effect": "Allow",
      "Principal": "*",
      "Action": "execute-api:Invoke",
      "Resource": "execute-api:/Prod/GET/pets",
      "Condition": {
          "IpAddress": {
              "aws:SourceIp": "1.2.3.4"
          }
        }
    }]
    IpRangeDenylist:
      - "10.20.30.40"
```

AWS ICH BIN

AWS IAM-Beispiel

YAML

```
Auth:
  Authorizers: AWS_IAM
```

ApiUsagePlan

Konfiguriert einen Nutzungsplan für eine API-Gateway-API. Weitere Informationen zu Nutzungsplänen finden Sie unter Nutzungspläne mit API-Schlüsseln erstellen und verwenden im API Gateway Developer Guide.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
  CreateUsagePlan: String
  Description: String
  Quota: QuotaSettings
  Tags: List
  Throttle: ThrottleSettings
  UsagePlanName: String
```

Eigenschaften

`CreateUsagePlan`

Legt fest, wie dieser Nutzungsplan konfiguriert ist. Gültige Werte sind `PER_API`, `SHARED` und `NONE`.

`PER_API`erstellt AWS::ApiGateway::UsagePlanAWS::ApiGateway::ApiKey, und AWS::ApiGateway::UsagePlanKeyRessourcen, die für diese API spezifisch sind. Diese Ressourcen haben IDs den logischen Wert von *<api-logical-id>*`UsagePlan` *<api-logical-id>*`ApiKey`*<api-logical-id>*`UsagePlanKey`, und.

`SHARED`erstellt AWS::ApiGateway::UsagePlanAWS::ApiGateway::ApiKey, und AWS::ApiGateway::UsagePlanKeyRessourcen, die von allen APIs gemeinsam genutzt werden, die auch `CreateUsagePlan: SHARED` in derselben AWS SAM Vorlage enthalten sind. Diese Ressourcen haben IDs den logischen Wert von `ServerlessUsagePlan`

`ServerlessApiKeyServerlessUsagePlanKey`, und. Wenn Sie diese Option verwenden, empfehlen wir, zusätzliche Konfigurationen für diesen Nutzungsplan nur für eine API-Ressource hinzuzufügen, um widersprüchliche Definitionen und einen unsicheren Status zu vermeiden.

`NONE`deaktiviert die Erstellung oder Verknüpfung eines Nutzungsplans mit dieser API. Dies ist nur erforderlich, wenn `SHARED` oder in der Sektion Globales der AWS SAM Vorlage angegeben `PER_API` ist.

Zulässige Werte: `PER_API`, `SHARED` und `NONE`

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Description`

Eine Beschreibung des Nutzungsplans.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Description` Eigenschaft einer `AWS::ApiGateway::UsagePlan` Ressource übergeben.

`Quota`

Konfiguriert die Anzahl von Anforderungen, die Benutzer innerhalb eines bestimmten Intervalls vornehmen können.

Typ: QuotaSettings

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Quota` Eigenschaft einer `AWS::ApiGateway::UsagePlan` Ressource übergeben.

`Tags`

Ein Array an beliebigen Tags (Schlüssel-Wert-Paaren), die dem Nutzungsplan zugewiesen werden sollen.

Diese Eigenschaft verwendet den CloudFormation Tag-Typ.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Tags` Eigenschaft einer `AWS::ApiGateway::UsagePlan` Ressource übergeben.

`Throttle`

Konfiguriert die gesamte Anforderungsrate (durchschnittliche Anforderungen pro Sekunde) und die Steigerungskapazität.

Typ: ThrottleSettings

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Throttle` Eigenschaft einer `AWS::ApiGateway::UsagePlan` Ressource übergeben.

`UsagePlanName`

Ein Name für den Nutzungsplan.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `UsagePlanName` Eigenschaft einer `AWS::ApiGateway::UsagePlan` Ressource übergeben.

Beispiele

UsagePlan

Im Folgenden finden Sie ein Beispiel für einen Nutzungsplan.

YAML

```
Auth:
  UsagePlan:
    CreateUsagePlan: PER_API
    Description: Usage plan for this API
    Quota:
```

```
      Limit: 500
      Period: MONTH
    Throttle:
      BurstLimit: 100
      RateLimit: 50
    Tags:
      - Key: TagName
        Value: TagValue
```

CognitoAuthorizer

Definieren Sie einen Amazon Cognito Cognito-Benutzerpool-Autorisierer.

Weitere Informationen und Beispiele finden Sie unter Kontrollieren Sie den API-Zugriff mit Ihrer AWS SAM Vorlage.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
AuthorizationScopes: List
Identity: CognitoAuthorizationIdentity
UserPoolArn: String | List
```

Eigenschaften

`AuthorizationScopes`

Liste der Autorisierungsbereiche für diesen Autorisierer.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Identity`

Diese Eigenschaft kann verwendet werden, um `IdentitySource` in einer eingehenden Anfrage einen Autorisierer anzugeben.

Typ: CognitoAuthorizationIdentity

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`UserPoolArn`

Die ARN (s) des Amazon Cognito Cognito-Benutzerpools, die für die Autorisierung verwendet werden sollen. Geben Sie einen einzelnen ARN als Zeichenfolge oder mehrere ARNs als Liste an, um mehrere Benutzerpools zu verwenden.

Typ: Zeichenfolge | Liste

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

CognitoAuth

Beispiel für Cognito Auth

YAML

```
Auth:
  Authorizers:
    MyCognitoAuth:
      AuthorizationScopes:
        - scope1
        - scope2
      UserPoolArn:
        Fn::GetAtt:
          - MyCognitoUserPool
          - Arn
      Identity:
        Header: MyAuthorizationHeader
        ValidationExpression: myauthvalidationexpression
```

CognitoAuthorizationIdentity

Diese Eigenschaft kann verwendet werden, um IdentitySource in einer eingehenden Anfrage nach einem Autorisierer ein anzugeben. Weitere Informationen dazu finden IdentitySource Sie unter der ApiGateway Authorizer-Erweiterung OpenApi .

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
Header: String
ReauthorizeEvery: Integer
ValidationExpression: String
```

Eigenschaften

`Header`

Geben Sie den Header-Namen für Authorization in der OpenApi Definition an.

Typ: Zeichenfolge

Required: No

Standard: Autorisierung

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`ReauthorizeEvery`

Der Zeitraum time-to-live (TTL) in Sekunden, der angibt, wie lange API Gateway Autorisierungsergebnisse zwischenspeichert. Wenn Sie einen Wert größer als 0 festlegen, speichert API Gateway die Genehmigerantworten im Cache. Standardmäßig legt API Gateway diese Eigenschaft auf 300 fest. Der maximale Wert ist 3600, oder 1 Stunde.

Typ: Ganzzahl

Required: No

Standard: 300

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`ValidationExpression`

Geben Sie einen Validierungsausdruck für die Validierung der eingehenden Identität an

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

CognitoAuthIdentity

YAML

```
Identity:
  Header: MyCustomAuthHeader
  ValidationExpression: Bearer.*
  ReauthorizeEvery: 30
```

LambdaRequestAuthorizer

Konfigurieren Sie einen Lambda Authorizer, um den Zugriff auf Ihre API mit einer Lambda-Funktion zu kontrollieren.

Weitere Informationen und Beispiele finden Sie unter Kontrollieren Sie den API-Zugriff mit Ihrer AWS SAM Vorlage.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
DisableFunctionDefaultPermissions: Boolean
FunctionArn: String
FunctionInvokeRole: String
```

```
FunctionPayloadType: String
Identity: LambdaRequestAuthorizationIdentity
```

Eigenschaften

`DisableFunctionDefaultPermissions`

Geben Sie `true` an, dass nicht AWS SAM automatisch eine `AWS::Lambda::Permissions` Ressource erstellt wird, um Berechtigungen zwischen Ihrer `AWS::Serverless::Api` Ressource und der Autorisierungs-Lambda-Funktion bereitzustellen.

Standardwert: `false`

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`FunctionArn`

Geben Sie den Funktions-ARN der Lambda-Funktion an, die die Autorisierung für die API bereitstellt.

> **Note**
>
> AWS SAM erstellt automatisch eine `AWS::Lambda::Permissions` Ressource, wenn dies angegeben `FunctionArn` ist. `AWS::Serverless::Api` Die `AWS::Lambda::Permissions` Ressource stellt Berechtigungen zwischen Ihrer API und der Autorisierungs-Lambda-Funktion bereit.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`FunctionInvokeRole`

Fügt der OpenApi Definition des Lambda-Autorisierers Autorisierer-Anmeldeinformationen hinzu.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein Äquivalent. CloudFormation

`FunctionPayloadType`

Diese Eigenschaft kann verwendet werden, um den Typ des Lambda Authorizers für eine API zu definieren.

Zulässige Werte: `TOKEN` oder `REQUEST`.

Typ: Zeichenfolge

Required: No

Standardwert: `TOKEN`

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig AWS SAM und hat kein CloudFormation Äquivalent.

`Identity`

Diese Eigenschaft kann verwendet werden, um `IdentitySource` in einer eingehenden Anfrage einen Autorisierer anzugeben. Diese Eigenschaft ist nur erforderlich, wenn die `FunctionPayloadType` Eigenschaft auf `REQUEST` gesetzt ist.

Typ: LambdaRequestAuthorizationIdentity

Required: Conditional

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

LambdaRequestAuth

YAML

```
Authorizers:
  MyLambdaRequestAuth:
```

```
    FunctionPayloadType: REQUEST
    FunctionArn:
      Fn::GetAtt:
        - MyAuthFunction
        - Arn
    FunctionInvokeRole:
      Fn::GetAtt:
        - LambdaAuthInvokeRole
        - Arn
    Identity:
      Headers:
        - Authorization1
```

LambdaRequestAuthorizationIdentity

Diese Eigenschaft kann verwendet werden, um IdentitySource in einer eingehenden Anfrage nach einem Autorisierer ein anzugeben. Weitere Informationen dazu finden IdentitySource Sie unter der ApiGateway Authorizer-Erweiterung OpenApi .

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
Context: List
Headers: List
QueryStrings: List
ReauthorizeEvery: Integer
StageVariables: List
```

Eigenschaften

`Context`

Konvertiert die angegebenen Kontextzeichenfolgen in die Mapping-Ausdrücke des Formats`context.contextString`.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Headers`

Konvertiert die Header in eine durch Kommas getrennte Zeichenfolge von Zuordnungsausdrücken des Formats. `method.request.header.name`

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein Äquivalent. CloudFormation

`QueryStrings`

Konvertiert die angegebenen Abfragezeichenfolgen in eine durch Kommas getrennte Zeichenfolge von Zuordnungsausdrücken des Formats. `method.request.querystring.queryString`

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`ReauthorizeEvery`

Der Zeitraum time-to-live (TTL) in Sekunden, der angibt, wie lange API Gateway Autorisierungsergebnisse zwischenspeichert. Wenn Sie einen Wert größer als 0 festlegen, speichert API Gateway die Genehmigerantworten im Cache. Standardmäßig legt API Gateway diese Eigenschaft auf 300 fest. Der maximale Wert ist 3600, oder 1 Stunde.

Typ: Ganzzahl

Required: No

Standard: 300

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`StageVariables`

Konvertiert die angegebenen Stufenvariablen in eine durch Kommas getrennte Zeichenfolge von Zuordnungsausdrücken des Formats. `stageVariables.stageVariable`

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

LambdaRequestIdentity

YAML

```
Identity:
  QueryStrings:
    - auth
  Headers:
    - Authorization
  StageVariables:
    - VARIABLE
  Context:
    - authcontext
  ReauthorizeEvery: 100
```

LambdaTokenAuthorizer

Konfigurieren Sie einen Lambda Authorizer, um den Zugriff auf Ihre API mit einer Lambda-Funktion zu kontrollieren.

Weitere Informationen und Beispiele finden Sie unter Kontrollieren Sie den API-Zugriff mit Ihrer AWS SAM Vorlage.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
DisableFunctionDefaultPermissions: Boolean
FunctionArn: String
FunctionInvokeRole: String
FunctionPayloadType: String
Identity: LambdaTokenAuthorizationIdentity
```

Eigenschaften

`DisableFunctionDefaultPermissions`

Geben Sie `true` an, dass nicht AWS SAM automatisch eine `AWS::Lambda::Permissions` Ressource erstellt wird, um Berechtigungen zwischen Ihrer `AWS::Serverless::Api` Ressource und der Autorisierungs-Lambda-Funktion bereitzustellen.

Standardwert: `false`

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`FunctionArn`

Geben Sie den Funktions-ARN der Lambda-Funktion an, die die Autorisierung für die API bereitstellt.

> **Note**
>
> AWS SAM erstellt automatisch eine `AWS::Lambda::Permissions` Ressource, wenn dies angegeben `FunctionArn` ist. `AWS::Serverless::Api` Die `AWS::Lambda::Permissions` Ressource stellt Berechtigungen zwischen Ihrer API und der Autorisierungs-Lambda-Funktion bereit.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`FunctionInvokeRole`

Fügt der OpenApi Definition des Lambda-Autorisierers Autorisierer-Anmeldeinformationen hinzu.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein Äquivalent. CloudFormation

`FunctionPayloadType`

Diese Eigenschaft kann verwendet werden, um den Typ des Lambda Authorizers für eine Api zu definieren.

Zulässige Werte: `TOKEN` oder `REQUEST`.

Typ: Zeichenfolge

Required: No

Standardwert: `TOKEN`

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig AWS SAM und hat kein CloudFormation Äquivalent.

`Identity`

Diese Eigenschaft kann verwendet werden, um `IdentitySource` in einer eingehenden Anfrage einen Autorisierer anzugeben. Diese Eigenschaft ist nur erforderlich, wenn die `FunctionPayloadType` Eigenschaft auf `REQUEST` gesetzt ist.

Typ: LambdaTokenAuthorizationIdentity

Required: Conditional

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

LambdaTokenAuth

YAML

```
Authorizers:
  MyLambdaTokenAuth:
    FunctionArn:
      Fn::GetAtt:
        - MyAuthFunction
        - Arn
    Identity:
      Header: MyCustomAuthHeader # OPTIONAL; Default: 'Authorization'
      ValidationExpression: mycustomauthexpression # OPTIONAL
      ReauthorizeEvery: 20 # OPTIONAL; Service Default: 300
```

BasicLambdaTokenAuth

YAML

```
Authorizers:
  MyLambdaTokenAuth:
    FunctionArn:
      Fn::GetAtt:
        - MyAuthFunction
        - Arn
```

LambdaTokenAuthorizationIdentity

Diese Eigenschaft kann verwendet werden, um IdentitySource in einer eingehenden Anfrage einen Autorisierer anzugeben. Weitere Informationen dazu finden IdentitySource Sie unter der ApiGateway Authorizer-Erweiterung OpenApi .

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
  Header: String
```

```
  ReauthorizeEvery: Integer
  ValidationExpression: String
```

Eigenschaften

`Header`

Geben Sie den Header-Namen für Authorization in der OpenApi Definition an.

Typ: Zeichenfolge

Required: No

Standard: Autorisierung

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`ReauthorizeEvery`

Der Zeitraum time-to-live (TTL) in Sekunden, der angibt, wie lange API Gateway Autorisierungsergebnisse zwischenspeichert. Wenn Sie einen Wert größer als 0 festlegen, speichert API Gateway die Genehmigerantworten im Cache. Standardmäßig legt API Gateway diese Eigenschaft auf 300 fest. Der maximale Wert ist 3600, oder 1 Stunde.

Typ: Ganzzahl

Required: No

Standard: 300

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`ValidationExpression`

Geben Sie einen Validierungsausdruck für die Validierung der eingehenden Identität an.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

LambdaTokenIdentity

YAML

```
Identity:
  Header: MyCustomAuthHeader
  ValidationExpression: Bearer.*
  ReauthorizeEvery: 30
```

ResourcePolicyStatement

Konfiguriert eine Ressourcenrichtlinie für alle Methoden und Pfade einer API. Weitere Informationen zu Ressourcenrichtlinien finden Sie unter Steuern des Zugriffs auf eine API mit API-Gateway-Ressourcenrichtlinien im API Gateway Developer Guide.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
  AwsAccountBlacklist: List
  AwsAccountWhitelist: List
  CustomStatements: List
  IntrinsicVpcBlacklist: List
  IntrinsicVpcWhitelist: List
  IntrinsicVpceBlacklist: List
  IntrinsicVpceWhitelist: List
  IpRangeBlacklist: List
  IpRangeWhitelist: List
  SourceVpcBlacklist: List
  SourceVpcWhitelist: List
```

Eigenschaften

`AwsAccountBlacklist`

Die zu blockierenden AWS Konten.

Typ: Liste von Zeichenfolgen

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig AWS SAM und hat kein CloudFormation Äquivalent.

`AwsAccountWhitelist`

Die AWS Konten, die zugelassen werden sollen. Ein Beispiel für die Verwendung dieser Eigenschaft finden Sie im Abschnitt Beispiele unten auf dieser Seite.

Typ: Liste von Zeichenfolgen

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`CustomStatements`

Eine Liste von benutzerdefinierten Ressourcenrichtlinien-Anweisungen, die auf diese API angewendet werden sollen. Ein Beispiel für die Verwendung dieser Eigenschaft finden Sie im Abschnitt Beispiele unten auf dieser Seite.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`IntrinsicVpcBlacklist`

Die Liste der zu blockierenden virtuellen privaten Clouds (VPCs), wobei jede VPC als Referenz angegeben ist, z. B. als dynamische Referenz oder als `Ref` intrinsische Funktion. Ein Beispiel für die Verwendung dieser Eigenschaft finden Sie im Abschnitt Beispiele unten auf dieser Seite.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`IntrinsicVpcWhitelist`

Die Liste der VPCs Zulässigen, wobei jede VPC als Referenz angegeben wird, z. B. als dynamische Referenz oder als `Ref` intrinsische Funktion.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig AWS SAM und hat kein Äquivalent. CloudFormation

`IntrinsicVpceBlacklist`

Die Liste der zu blockierenden VPC-Endpunkte, wobei jeder VPC-Endpunkt als Referenz angegeben ist, z. B. als dynamische Referenz oder als intrinsische Funktion. `Ref`

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein Äquivalent. CloudFormation

`IntrinsicVpceWhitelist`

Die Liste der VPC-Endpunkte, die zugelassen werden sollen, wobei jeder VPC-Endpunkt als Referenz angegeben ist, z. B. als dynamische Referenz oder als systemeigene Funktion. `Ref` Ein Beispiel für die Verwendung dieser Eigenschaft finden Sie im Abschnitt Beispiele unten auf dieser Seite.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`IpRangeBlacklist`

Die IP-Adressen oder Adressbereiche, die blockiert werden sollen. Ein Beispiel für die Verwendung dieser Eigenschaft finden Sie im Abschnitt Beispiele unten auf dieser Seite.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`IpRangeWhitelist`

Die IP-Adressen oder Adressbereiche, die zugelassen werden sollen.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`SourceVpcBlacklist`

Die Quell-VPC oder die VPC-Endpoints, die blockiert werden sollen. Quell-VPC-Namen müssen mit beginnen `"vpc-"` und Quell-VPC-Endpunktnamen müssen mit beginnen. `"vpce-"` Ein Beispiel für die Verwendung dieser Eigenschaft finden Sie im Abschnitt Beispiele unten auf dieser Seite.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`SourceVpcWhitelist`

Die Quell-VPC oder VPC-Endpoints, die zugelassen werden sollen. Quell-VPC-Namen müssen mit beginnen `"vpc-"` und Quell-VPC-Endpunktnamen müssen mit beginnen. `"vpce-"`

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

Beispiel für eine Ressourcenrichtlinie

Das folgende Beispiel blockiert zwei IP-Adressen und eine Quell-VPC und lässt ein AWS Konto zu.

YAML

```
Auth:
  ResourcePolicy:
    CustomStatements: [{
                         "Effect": "Allow",
                         "Principal": "*",
                         "Action": "execute-api:Invoke",
                         "Resource": "execute-api:/Prod/GET/pets",
                         "Condition": {
                           "IpAddress": {
                             "aws:SourceIp": "1.2.3.4"
                           }
                         }
                       }]
    IpRangeBlacklist:
      - "10.20.30.40"
      - "1.2.3.4"
    SourceVpcBlacklist:
      - "vpce-1a2b3c4d"
    AwsAccountWhitelist:
      - "111122223333"
    IntrinsicVpcBlacklist:
      - "{{resolve:ssm:SomeVPCReference:1}}"
      - !Ref MyVPC
    IntrinsicVpceWhitelist:
      - "{{resolve:ssm:SomeVPCEReference:1}}"
      - !Ref MyVPCE
```

ApiDefinition

Ein OpenAPI-Dokument, das die API definiert.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
  Bucket: String
  Key: String
  Version: String
```

Eigenschaften

`Bucket`

Der Name des Amazon S3 S3-Buckets, in dem die OpenAPI-Datei gespeichert ist.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Bucket` Eigenschaft des `AWS::ApiGateway::RestApi S3Location` Datentyps übergeben.

`Key`

Der Amazon S3 S3-Schlüssel der OpenAPI-Datei.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Key` Eigenschaft des `AWS::ApiGateway::RestApi S3Location` Datentyps übergeben.

`Version`

Für versionierte Objekte die Version der OpenAPI-Datei.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Version` Eigenschaft des `AWS::ApiGateway::RestApi S3Location` Datentyps übergeben.

Beispiele

Definition: Uri, Beispiel.

Beispiel für eine API-Definition

YAML

```
DefinitionUri:
  Bucket: amzn-s3-demo-bucket-name
```

```
  Key: mykey-name
  Version: 121212
```

CorsConfiguration

Verwalten Sie Cross-Origin Resource Sharing (CORS) für Ihr API Gateway. APIs Geben Sie die Domäne, die zugelassen werden soll, als Zeichenfolge an oder geben Sie ein Wörterbuch mit zusätzlicher Cors-Konfiguration an.

> **Note**
>
> CORS erfordert eine AWS SAM Änderung Ihrer OpenAPI-Definition. Erstellen Sie eine Inline-OpenAPI-Definition in`DefinitionBody`, um CORS zu aktivieren. Wenn `CorsConfiguration` das in der OpenAPI-Definition und auch auf Eigenschaftsebene festgelegt ist, werden sie AWS SAM zusammengeführt. Die Eigenschaftsebene hat Vorrang vor der OpenAPI-Definition.

Weitere Informationen zu CORS finden Sie unter Aktivieren von CORS für eine API-Gateway-REST-API-Ressource im API Gateway Developer Guide.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
  AllowCredentials: Boolean
  AllowHeaders: String
  AllowMethods: String
  AllowOrigin: String
  MaxAge: String
```

Eigenschaften

`AllowCredentials`

Boolescher Wert, der angibt, ob die Anfrage Anmeldeinformationen enthalten darf.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`AllowHeaders`

Zeichenfolge von Headern, die zugelassen werden sollen.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`AllowMethods`

Zeichenfolge, die die zuzulassenden HTTP-Methoden enthält.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`AllowOrigin`

Quellzeichenfolge, die zugelassen werden soll. Dies kann eine durch Kommas getrennte Liste im Zeichenkettenformat sein.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`MaxAge`

Zeichenfolge, die die Anzahl der Sekunden enthält, um die CORS-Preflight-Anfrage zwischenzuspeichern.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

CorsConfiguration

Beispiel für eine CORS-Konfiguration. Dies ist nur ein Teil einer AWS SAM Vorlagendatei, die eine AWS::Serverless::Api Definition mit konfiguriertem CORS und einem zeigt. AWS::Serverless: :Funktion Wenn Sie eine Lambda-Proxyintegration oder eine HTTP-Proxyintegration verwenden, muss Ihr Backend die Header `Access-Control-Allow-OriginAccess-Control-Allow-Methods`, und `Access-Control-Allow-Headers` zurückgeben.

YAML

```
Resources:
  ApiGatewayApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      Cors:
        AllowMethods: "'POST, GET'"
        AllowHeaders: "'X-Forwarded-For'"
        AllowOrigin: "'https://example.com'"
        MaxAge: "'600'"
        AllowCredentials: true
  ApiFunction: # Adds a GET method at the root resource via an Api event
    Type: AWS::Serverless::Function
    Properties:
      Events:
        ApiEvent:
          Type: Api
          Properties:
            Path: /
            Method: get
            RestApiId:
              Ref: ApiGatewayApi
      Runtime: python3.10
      Handler: index.handler
```

```
      InlineCode: |
        import json
        def handler(event, context):
          return {
          'statusCode': 200,
          'headers': {
            'Access-Control-Allow-Headers': 'Content-Type',
            'Access-Control-Allow-Origin': 'https://example.com',
            'Access-Control-Allow-Methods': 'POST, GET'
            },
          'body': json.dumps('Hello from Lambda!')
          }
```

DomainConfiguration

Konfiguriert eine benutzerdefinierte Domain für eine API.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
AccessAssociation: DomainAccessAssociation
BasePath: List
CertificateArn: String
DomainName: String
EndpointAccessMode: String
EndpointConfiguration: String
MutualTlsAuthentication: MutualTlsAuthentication
NormalizeBasePath: Boolean
OwnershipVerificationCertificateArn: String
Policy: Json
Route53: Route53Configuration
SecurityPolicy: String
```

Eigenschaften

`AccessAssociation`

Die Konfiguration, die zum Generieren der `AWS::ApiGateway::DomainNameAccessAssociation` Ressource erforderlich ist.

AWS SAM generiert eine AWS::ApiGateway::DomainNameAccessAssociationRessource, wenn diese Eigenschaft festgelegt ist. Hinweise zu generierten CloudFormation Ressourcen finden Sie unterGeneriert CloudFormation Ressourcen für AWS SAM.

Typ: DomainAccessAssociation

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`BasePath`

Eine Liste der Basepfade, die mit dem Amazon API Gateway Gateway-Domänennamen konfiguriert werden sollen.

Typ: Liste

Required: No

Standard:/

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `BasePath` Eigenschaft einer `AWS::ApiGateway::BasePathMapping` Ressource. AWS SAM erstellt mehrere `AWS::ApiGateway::BasePathMapping` Ressourcen, eine pro, die in dieser Eigenschaft `BasePath` angegeben ist.

`CertificateArn`

Der Amazon-Ressourcenname (ARN) eines AWS verwalteten Zertifikats, der Endpunkt dieses Domainnamens. AWS Certificate Manager ist die einzige unterstützte Quelle.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `CertificateArn` Eigenschaft einer `AWS::ApiGateway::DomainName` Ressource. Wenn auf `REGIONAL` (der Standardwert) gesetzt `EndpointConfiguration` ist, `CertificateArn` wird RegionalCertificateArnin zugeordnet`AWS::ApiGateway::DomainName`. Wenn der auf gesetzt `EndpointConfiguration` ist`EDGE`, `CertificateArn` wird CertificateArnin

zugeordnetAWS::ApiGateway::DomainName. Wenn auf gesetzt EndpointConfiguration istPRIVATE, wird diese Eigenschaft an die AWS::ApiGateway::DomainNameV2-Ressource übergeben.

Zusätzliche Hinweise: Für einen EDGE Endpunkt müssen Sie das Zertifikat in der us-east-1 AWS Region erstellen.

DomainName

Der benutzerdefinierte Domainname für Ihre API-Gateway-API. Großbuchstaben werden nicht unterstützt.

AWS SAM generiert eine AWS::ApiGateway::DomainNameRessource, wenn diese Eigenschaft festgelegt ist. Hinweise zu diesem Szenario finden Sie unterDomainName Eigenschaft ist angegeben. Hinweise zu generierten CloudFormation Ressourcen finden Sie unterGeneriert CloudFormation Ressourcen für AWS SAM.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die DomainName Eigenschaft einer AWS::ApiGateway::DomainName Ressource übergeben, oder an AWS::ApiGateway::DomainNameV2when, wenn sie auf gesetzt EndpointConfiguration istPRIVATE.

EndpointConfiguration

Definiert den Typ des API-Gateway-Endpunkts, der der benutzerdefinierten Domain zugeordnet werden soll. Der Wert dieser Eigenschaft bestimmt, wie die CertificateArn Eigenschaft zugeordnet wird. CloudFormation

Gültige Werte: EDGE, REGIONAL, oder PRIVATE

Typ: Zeichenfolge

Required: No

Standardwert: REGIONAL

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`EndpointAccessMode`

Der Zugriffsmodus für den Endpunkt des benutzerdefinierten Domainnamens. Erforderlich, wenn erweiterte Sicherheitsrichtlinien (solche mit dem Präfix`SecurityPolicy_`) verwendet werden.

Zulässige Werte: `STRICT` oder `BASIC`.

Typ: Zeichenfolge

Required: Conditional

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `EndpointAccessMode` Eigenschaft einer `AWS::ApiGateway::DomainName` Ressource übergeben, oder an `AWS::ApiGateway::DomainNameV2`when `EndpointConfiguration` ist auf `PRIVATE` gesetzt.

`MutualTlsAuthentication`

Die TLS-Authentifizierungskonfiguration (Mutual Transport Layer Security) für einen benutzerdefinierten Domainnamen.

Typ: MutualTlsAuthentication

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `MutualTlsAuthentication` Eigenschaft einer `AWS::ApiGateway::DomainName` Ressource übergeben.

`NormalizeBasePath`

Gibt an, ob nicht-alphanumerische Zeichen in den durch die Eigenschaft definierten Basepfaden zulässig sind. `BasePath` Wenn auf gesetzt, werden nicht-alphanumerische Zeichen aus `True` Basepfaden entfernt.

Wird zusammen mit der Eigenschaft verwendet. `NormalizeBasePath` `BasePath`

Typ: Boolesch

Required: No

Standard: Wahr

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`OwnershipVerificationCertificateArn`

Die ARN des öffentlichen Zertifikats, das von ACM ausgestellt wurde, um den Besitz Ihrer benutzerdefinierten Domain zu überprüfen. Nur erforderlich, wenn Sie Mutual TLS konfigurieren und einen von ACM importierten oder privaten CA-Zertifikat-ARN für angeben. `CertificateArn`

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `OwnershipVerificationCertificateArn` Eigenschaft einer `AWS::ApiGateway::DomainName` Ressource übergeben.

`Policy`

Die IAM-Richtlinie, die an den API-Gateway-Domänennamen angehängt werden soll. Gilt nur, wenn auf `PRIVATE` eingestellt `EndpointConfiguration` ist.

Type: Json

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Policy` Eigenschaft einer `AWS::ApiGateway::DomainNameV2` Ressource übergeben, wenn sie auf gesetzt `EndpointConfiguration` ist`PRIVATE`. Beispiele für gültige Richtliniendokumente finden Sie unter AWS::ApiGateway::DomainNameV2.

`Route53`

Definiert eine Amazon Route 53-Konfiguration.

Typ: Route53-Konfiguration

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein Äquivalent. CloudFormation

`SecurityPolicy`

Die TLS-Version plus die Cipher Suite für diesen Domainnamen.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `SecurityPolicy` Eigenschaft einer `AWS::ApiGateway::DomainName` Ressource übergeben, oder an `AWS::ApiGateway::DomainNameV2`when `EndpointConfiguration` ist auf gesetzt. `PRIVATE` Für `PRIVATE` Endgeräte wird nur TLS_1_2 unterstützt.

Beispiele

DomainName

DomainName Beispiel

YAML

```
Domain:
  DomainName: www.example.com
  CertificateArn: arn-example
  EndpointConfiguration: EDGE
  Route53:
    HostedZoneId: Z1PA6795UKMFR9
  BasePath:
    - foo
    - bar
```

DomainAccessAssociation

Konfiguriert eine Domänenzugriffszuordnung zwischen einer Zugriffszuordnungsquelle und einem privaten benutzerdefinierten Domänennamen. Die einzige unterstützte Quelle für die Zugriffszuweisung ist eine VPC-Endpunkt-ID. Weitere Informationen finden Sie unter Benutzerdefinierte Domainnamen für private APIs in API Gateway.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
VpcEndpointId: String
```

Eigenschaften

`VpcEndpointId`

Die Endpunkt-ID des VPC-Schnittstellenendpunkts, der dem API Gateway VPC-Dienst zugeordnet ist.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `AccessAssociationSource` Eigenschaft einer `AWS::ApiGateway::DomainNameAccessAssociation` Ressource übergeben.

Route53Configuration

Konfiguriert die Route53-Datensätze für eine API.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer AWS Serverless Application Model (AWS SAM) -Vorlage zu deklarieren.

YAML

```
DistributionDomainName: String
EvaluateTargetHealth: Boolean
HostedZoneId: String
HostedZoneName: String
IpV6: Boolean
Region: String
SetIdentifier: String
VpcEndpointDomainName: String
VpcEndpointHostedZoneId: String
```

Eigenschaften

`DistributionDomainName`

Konfiguriert eine benutzerdefinierte Verteilung des benutzerdefinierten API-Domänennamens.

Typ: Zeichenfolge

Required: No

Standard: Verwenden Sie die API Gateway Gateway-Distribution.

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `DNSName` Eigenschaft einer `AWS::Route53::RecordSetGroup AliasTarget` Ressource übergeben.

Zusätzliche Hinweise: Der Domainname einer CloudFrontDistribution.

`EvaluateTargetHealth`

Wenn EvaluateTargetHealth wahr, erbt ein Aliaseintrag den Zustand der referenzierten AWS Ressource, z. B. eines Elastic Load Balancing Balancing-Load Balancers oder eines anderen Datensatzes in der Hosting-Zone.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `EvaluateTargetHealth` Eigenschaft einer `AWS::Route53::RecordSetGroup AliasTarget` Ressource übergeben.

Zusätzliche Hinweise: Sie können den Wert nicht auf „true“ EvaluateTargetHealth setzen, wenn es sich bei dem Alias-Ziel um eine CloudFront Distribution handelt.

`HostedZoneId`

Die ID der gehosteten Zone, in der Sie Datensätze erstellen möchten.

Geben Sie `HostedZoneName` oder `HostedZoneId` an, jedoch nicht beides. Wenn Sie mehrere gehostete Zonen mit dem gleichen Domainnamen haben, müssen Sie die gehostete Zone mit der `HostedZoneId` angeben.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `HostedZoneId` Eigenschaft einer `AWS::Route53::RecordSetGroup RecordSet` Ressource übergeben.

`HostedZoneName`

Der Name der gehosteten Zone, in der Sie Datensätze erstellen möchten.

Geben Sie `HostedZoneName` oder `HostedZoneId` an, jedoch nicht beides. Wenn Sie mehrere gehostete Zonen mit dem gleichen Domainnamen haben, müssen Sie die gehostete Zone mit der `HostedZoneId` angeben.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `HostedZoneName` Eigenschaft einer `AWS::Route53::RecordSetGroup RecordSet` Ressource übergeben.

`IpV6`

Wenn diese Eigenschaft gesetzt ist, wird eine `AWS::Route53::RecordSet` Ressource AWS SAM erstellt und Type AAAA für die angegebene Ressource auf gesetzt HostedZone.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Region`

Nur latenzbasierte Ressourcendatensätze: Die EC2 Amazon-Region, in der Sie die Ressource erstellt haben, auf die sich dieser Ressourcendatensatz bezieht. Bei der Ressource handelt es sich in der Regel um eine AWS Ressource, z. B. eine EC2 Instance oder einen ELB-Load Balancer, auf die je nach Datensatztyp mit einer IP-Adresse oder einem DNS-Domainnamen verwiesen wird.

Wenn Amazon Route 53 eine DNS-Abfrage für einen Domainnamen und einen Domain-Typ empfängt, für den Sie Latenzressourcendatensätze erstellt haben, wählt Route 53 den Latenzressourcendatensatz aus, der die niedrigste Latenz zwischen dem Endbenutzer und der zugehörigen EC2 Amazon-Region aufweist. Route 53 gibt dann den Wert zurück, der dem ausgewählten Ressourcendatensatz zugeordnet ist.

Beachten Sie Folgendes:

- Sie können nur einen `ResourceRecord` pro Latenz-Ressourcendatensatz angeben.
- Sie können nur einen Latenzressourcendatensatz für jede EC2 Amazon-Region erstellen.

- Sie müssen keine Latenzressourcendatensätze für alle EC2 Amazon-Regionen erstellen. Route 53 wählt die Region mit der besten Latenz aus den Regionen aus, für die Sie Latenz-Ressourcendatensätze erstellen.
- Sie können keine Nicht-Latenz-Ressourcendatensätze erstellen, die die gleichen Werte für die Elemente `Name` und `Type` haben wie Latenz-Ressourcendatensätze.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Region` Eigenschaft eines `AWS::Route53::RecordSetGroup` `RecordSet` Datentyps übergeben.

`SetIdentifier`

Ressourcendatensätze, die eine andere Routing-Richtlinie als "einfach" haben: Ein Bezeichner, der zwischen mehreren Ressourcendatensätzen unterscheidet, die die gleiche Kombination aus Name und Typ haben, wie beispielsweise mehrere gewichtete Ressourcendatensätze namens acme.example.com, die einen Typ A haben. In einer Gruppe von Ressourcendatensätzen, die den gleichen Namen und Typ haben, muss der Wert von `SetIdentifier` für jeden Ressourcen-Datensatz eindeutig sein.

Informationen zu Routing-Richtlinien finden Sie unter Auswahl einer Routing-Richtlinie im Amazon Route 53-Entwicklerhandbuch.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `SetIdentifier` Eigenschaft eines `AWS::Route53::RecordSetGroup` `RecordSet` Datentyps übergeben.

`VpcEndpointDomainName`

Ein DNS-Name des VPC-Schnittstellenendpunkts, der dem API Gateway VPC-Dienst zugeordnet ist. Diese Eigenschaft ist nur für private Domänen erforderlich.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `DNSName` Eigenschaft eines `AWS::Route53::RecordSet` `AliasTarget` Feldes übergeben.

`VpcEndpointHostedZoneId`

Die Hosting-Zonen-ID des VPC-Schnittstellenendpunkts, der dem API Gateway VPC-Dienst zugeordnet ist. Diese Eigenschaft ist nur für private Domänen erforderlich.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `HostedZoneId` Eigenschaft eines `AWS::Route53::RecordSet` `AliasTarget` Feldes übergeben.

Beispiele

Einfaches Beispiel

In diesem Beispiel konfigurieren wir eine benutzerdefinierte Domain und Route 53-Datensätze für unsere API.

YAML

```
Resources:
  MyApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      Domain:
        DomainName: www.example.com
        CertificateArn: arn:aws:acm:us-east-1:123456789012:certificate/
abcdef12-3456-7890-abcd-ef1234567890
        EndpointConfiguration: REGIONAL
        Route53:
          HostedZoneId: ABCDEFGHIJKLMNOP
```

EndpointConfiguration

Der Endpunkttyp einer REST-API.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
  IpAddressType: String
  Type: String
  VPCEndpointIds: List
```

Eigenschaften

`IpAddressType`

Die IP-Adresstypen, die eine API (RestApi) aufrufen können.

Zulässige Werte: `ipv4` oder `dualstack`.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `IpAddressType` Eigenschaft des `AWS::ApiGateway::RestApi EndpointConfiguration` Datentyps übergeben.

`Type`

Der Endpunkttyp einer REST-API.

Gültige Werte: `EDGE` oder `REGIONAL` oder `PRIVATE`

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Types` Eigenschaft des `AWS::ApiGateway::RestApi EndpointConfiguration` Datentyps übergeben.

`VPCEndpointIds`

Eine Liste der VPC-Endpunkte IDs einer REST-API, für die Route53-Aliase erstellt werden sollen.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `VpcEndpointIds` Eigenschaft des Datentyps übergeben. `AWS::ApiGateway::RestApi EndpointConfiguration`

Beispiele

EndpointConfiguration

Beispiel für eine Endpunktkonfiguration

YAML

```
EndpointConfiguration:
  Type: PRIVATE
  VPCEndpointIds:
    - vpce-123a123a
    - vpce-321a321a
```

AWS::Serverless::Application

Bettet eine serverlose Anwendung aus dem AWS Serverless Application Repositoryoder aus einem Amazon S3 S3-Bucket als verschachtelte Anwendung ein. Verschachtelte Anwendungen werden als verschachtelte AWS::CloudFormation::StackRessourcen bereitgestellt, die mehrere andere Ressourcen, einschließlich anderer Ressourcen, enthalten können. AWS::Serverless::Application

> **Note**
>
> Bei der Bereitstellung auf werden AWS CloudFormation Ihre AWS SAM Ressourcen in AWS SAM Ressourcen umgewandelt. CloudFormation Weitere Informationen finden Sie unter Generiert CloudFormation Ressourcen für AWS SAM.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
Type: AWS::Serverless::Application
```

```
Properties:
  Location: String | ApplicationLocationObject
  NotificationARNs: List
  Parameters: Map
  Tags: Map
  TimeoutInMinutes: Integer
```

Eigenschaften

`Location`

Vorlagen-URL, Dateipfad oder Standortobjekt einer verschachtelten Anwendung.

Wenn eine Vorlagen-URL bereitgestellt wird, muss sie dem in der CloudFormation TemplateUrl Dokumentation angegebenen Format entsprechen und eine gültige CloudFormation SAM-Vorlage enthalten. An ApplicationLocationObject kann verwendet werden, um eine Anwendung anzugeben, die auf der veröffentlicht wurde AWS Serverless Application Repository.

Wenn ein lokaler Dateipfad angegeben wird, muss die Vorlage den Workflow durchlaufen, der den `sam package` Befehl `sam deploy` oder enthält, damit die Anwendung ordnungsgemäß transformiert werden kann.

Typ: Zeichenfolge | ApplicationLocationObject

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `TemplateURL` Eigenschaft einer `AWS::CloudFormation::Stack` Ressource. Die CloudFormation Version benötigt keine ZeitApplicationLocationObject, um eine Anwendung von der abzurufen AWS Serverless Application Repository.

`NotificationARNs`

Eine Liste vorhandener Amazon SNS SNS-Themen, an die Benachrichtigungen über Stack-Ereignisse gesendet werden.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `NotificationARNs` Eigenschaft einer `AWS::CloudFormation::Stack` Ressource übergeben.

`Parameters`

Werte der Anwendungsparameter.

Typ: Karte

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Parameters` Eigenschaft einer `AWS::CloudFormation::Stack` Ressource übergeben.

`Tags`

Eine Zuordnung (Zeichenfolge zu Zeichenfolge), die die Tags angibt, die dieser Anwendung hinzugefügt werden sollen. Schlüssel und Werte sind auf alphanumerische Zeichen beschränkt. Schlüssel können 1 bis 127 Unicode-Zeichen lang sein und ihnen darf nicht das Präfix aws: vorangestellt werden. Werte können 1 bis 255 Unicode-Zeichen lang sein.

Typ: Karte

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `Tags` Eigenschaft einer `AWS::CloudFormation::Stack` Ressource. Die Tags-Eigenschaft in SAM besteht aus Key:Value-Paaren; darin besteht CloudFormation sie aus einer Liste von Tag-Objekten. Wenn der Stack erstellt ist, fügt SAM dieser Anwendung automatisch ein `lambda:createdBy:SAM` Tag hinzu. Wenn diese Anwendung aus dem stammt AWS Serverless Application Repository, dann wird SAM außerdem automatisch die beiden zusätzlichen Tags `serverlessrepo:applicationId:`*`ApplicationId`* und`serverlessrepo:semanticVersion:`*`SemanticVersion`*.

`TimeoutInMinutes`

Die Zeitspanne in Minuten, die darauf CloudFormation wartet, dass der verschachtelte Stapel den `CREATE_COMPLETE` Status erreicht. Standardmäßig gibt es keine Zeitüberschreitung. Wenn CloudFormation erkannt wird, dass der verschachtelte Stapel den `CREATE_COMPLETE` Status erreicht hat, markiert es die Ressource des verschachtelten Stacks als `CREATE_COMPLETE` im übergeordneten Stapel und setzt die Erstellung des übergeordneten Stacks fort. Wenn der Timeout-Zeitraum abläuft, bevor der verschachtelte Stack den Wert erreicht`CREATE_COMPLETE`, wird der verschachtelte Stack als ausgefallen CloudFormation markiert und sowohl der verschachtelte Stack als auch der übergeordnete Stack zurückgesetzt.

Typ: Ganzzahl

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `TimeoutInMinutes` Eigenschaft einer Ressource übergeben. `AWS::CloudFormation::Stack`

Rückgabewerte

Punkt

Wenn die logische ID dieser Ressource der `Ref` systemeigenen Funktion zur Verfügung gestellt wird, gibt sie den Ressourcennamen der zugrunde liegenden `AWS::CloudFormation::Stack` Ressource zurück.

Weitere Informationen zur Verwendung der `Ref` Funktion finden Sie `Ref`im AWS CloudFormation Benutzerhandbuch.

Fn:: GetAtt

`Fn::GetAtt` gibt einen Wert für ein angegebenes Attribut dieses Typs zurück. Im Folgenden sehen Sie die verfügbaren Attribute und Beispielrückgabewerte.

Weitere Informationen zur Verwendung `Fn::GetAtt` finden Sie `Fn::GetAtt`im AWS CloudFormation Benutzerhandbuch.

`Outputs.ApplicationOutputName`

Der Wert der Stack-Ausgabe mit dem Namen*`ApplicationOutputName`*.

Beispiele

SAR-Anwendung

Anwendung, die eine Vorlage aus dem Serverless Application Repository verwendet

YAML

```
Type: AWS::Serverless::Application
Properties:
  Location:
```

```
    ApplicationId: 'arn:aws:serverlessrepo:us-east-1:012345678901:applications/my-
application'
    SemanticVersion: 1.0.0
  Parameters:
    StringParameter: parameter-value
    IntegerParameter: 2
```

Normale Anwendung

Anwendung von einer S3-URL

YAML

```
Type: AWS::Serverless::Application
Properties:
  Location: https://s3.amazonaws.com/sam-s3-demo-bucket/template.yaml
```

ApplicationLocationObject

Eine Anwendung, die auf dem veröffentlicht wurde AWS Serverless Application Repository.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
  ApplicationId: String
  SemanticVersion: String
```

Eigenschaften

`ApplicationId`

Der Amazon-Ressourcenname (ARN) der Anwendung.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`SemanticVersion`

Die semantische Version der Anwendung.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

meine Anwendung

Beispiel für ein Objekt für den Speicherort einer Anwendung

YAML

```
Location:
  ApplicationId: 'arn:aws:serverlessrepo:us-east-1:012345678901:applications/my-
application'
  SemanticVersion: 1.0.0
```

AWS::Serverless::CapacityProvider

Erstellt einen Kapazitätsanbieter für AWS Lambda Funktionen, die den Betrieb von Lambda Managed Instances auf kundeneigenen Amazon Elastic Compute Cloud-Instances ermöglichen. Diese Ressource ist Teil der Lambda Managed Instances-Funktion, die mithilfe von Amazon EC2-Preismodellen eine Kostenoptimierung für umfangreiche Lambda-Workloads ermöglicht.

Der Kapazitätsanbieter verwaltet den Lebenszyklus von Amazon EC2-Instances und stellt die notwendige Infrastruktur bereit, damit Lambda-Funktionen auf kundeneigenen Rechenressourcen ausgeführt werden können, während das serverlose Programmiermodell beibehalten wird.

> **Note**
>
> Wenn Sie es bereitstellen AWS CloudFormation, AWS SAM wandelt er Ihre Ressourcen in Ressourcen um. AWS SAM CloudFormation Weitere Informationen finden Sie unter Generiert CloudFormation Ressourcen für AWS SAM.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer AWS Serverless Application Model (AWS SAM) -Vorlage zu deklarieren.

YAML

```
Type: AWS::Serverless::CapacityProvider
Properties:
  CapacityProviderName: String
  VpcConfig: VpcConfig
  OperatorRole: String
  Tags: Map
  PropagateTags: Boolean
  InstanceRequirements: InstanceRequirements
  ScalingConfig: ScalingConfig
  LoggingConfig: CapacityProviderLoggingConfig
  KmsKeyArn: String
  ManagedResourceTags: ManagedResourceTags
```

Eigenschaften

`CapacityProviderName`

Der Name des Kapazitätsanbieters. Dieser Name muss innerhalb Ihres AWS Kontos und Ihrer Region eindeutig sein.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `CapacityProviderName` Eigenschaft einer `AWS::Lambda::CapacityProvider` Ressource übergeben.

`VpcConfig`

Die VPC-Konfiguration für den Kapazitätsanbieter. Gibt die VPC-Subnetze und Sicherheitsgruppen an, in denen Amazon EC2-Instances gestartet werden.

Typ: VpcConfig

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `VpcConfig` Eigenschaft einer Ressource übergeben. `AWS::Lambda::CapacityProvider`

`OperatorRole`

Der ARN der Operatorrolle für Lambda mit Berechtigungen zum Erstellen und Verwalten von Amazon EC2-Instances und zugehörigen Ressourcen im Kundenkonto. Falls nicht angegeben, AWS SAM wird automatisch eine Rolle mit den erforderlichen Berechtigungen generiert.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `CapacityProviderOperatorRoleArn` Eigenschaft `PermissionsConfig` einer `AWS::Lambda::CapacityProvider` Ressource übergeben.

`Tags`

Eine Zuordnung von Schlüssel-Wert-Paaren, die auf den Kapazitätsanbieter und die zugehörigen Ressourcen angewendet werden sollen.

Typ: Karte

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Tags` Eigenschaft einer `AWS::Lambda::CapacityProvider` Ressource übergeben. Die `Tags` Eigenschaft in AWS SAM besteht aus Schlüssel-Wert-Paaren (wohingegen CloudFormation diese Eigenschaft aus einer Liste von Tag-Objekten besteht). Fügt dieser Lambda-Funktion und den für diese Funktion generierten Standardrollen außerdem AWS SAM automatisch ein `lambda:createdBy:SAM` Tag hinzu.

`PropagateTags`

Gibt an, ob Tags aus der Tags-Eigenschaft an Ihre `AWS::Serverless::CapacityProvider` generierten Ressourcen übergeben werden sollen oder nicht. Setzen Sie diesen Wert auf`True`, um Tags in Ihren generierten Ressourcen zu verbreiten.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat keine CloudFormation Entsprechung.

`InstanceRequirements`

Spezifikationen für die Typen von Recheninstanzen, die der Kapazitätsanbieter verwenden kann. Dazu gehören Architekturanforderungen `allowed` und/oder `excluded` Instanztypen.

Typ: InstanceRequirements

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `InstanceRequirements` Eigenschaft einer `AWS::Lambda::CapacityProvider` Ressource übergeben.

`ScalingConfig`

Die Skalierungskonfiguration für den Kapazitätsanbieter. Definiert, wie der Kapazitätsanbieter Amazon EC2-Instances je nach Bedarf skaliert.

Typ: ScalingConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `CapacityProviderScalingConfig` Eigenschaft einer `AWS::Lambda::CapacityProvider` Ressource übergeben.

`LoggingConfig`

Wird verwendet`LoggingConfig`, um die Systemprotokollierung für den Kapazitätsanbieter zu konfigurieren. Systemprotokolle erfassen Skalierungsaktivitäten und Betriebsereignisse und senden sie auf konfigurierbarer CloudWatch Protokollebene an eine Amazon-Protokollgruppe.

Typ: CapacityProviderLoggingConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `LoggingConfig` Eigenschaft `TelemetryConfig` einer `AWS::Lambda::CapacityProvider` Ressource übergeben.

`KmsKeyArn`

Der ARN des AWS KMS Schlüssels, der zum Verschlüsseln von Daten im Ruhezustand und bei der Übertragung für den Kapazitätsanbieter verwendet wird.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `KmsKeyArn` Eigenschaft einer `AWS::Lambda::CapacityProvider` Ressource übergeben.

`ManagedResourceTags`

Wendet explizite Tags auf die Amazon Elastic Compute Cloud-Instances und andere verwaltete Ressourcen an, die der Kapazitätsanbieter in Ihrem Namen bereitstellt.

Typ: ManagedResourceTags

Required: No

CloudFormation Kompatibilität: AWS SAM verwendet diese Eigenschaft, um die `PropagateTags` Eigenschaft einer `AWS::Lambda::CapacityProvider` Ressource zu erstellen.

Rückgabewerte

Punkt

Wenn die logische ID dieser Ressource für die `Ref` intrinsische Funktion bereitgestellt wird, gibt sie den Namen des Kapazitätsanbieters zurück.

Weitere Informationen zur Verwendung der `Ref` Funktion finden Sie `Ref` im AWS CloudFormation Benutzerhandbuch.

Fan:: GetAtt

`Fn::GetAtt` gibt einen Wert für ein angegebenes Attribut dieses Typs zurück. Im Folgenden sehen Sie die verfügbaren Attribute und Beispielrückgabewerte.

Weitere Informationen zur Verwendung `Fn::GetAtt` finden Sie `Fn::GetAtt` im AWS CloudFormation Benutzerhandbuch.

`Arn`

Der ARN des Kapazitätsanbieters.

Beispiele

Anbieter von Basiskapazität

Im folgenden Beispiel wird ein Basiskapazitätsanbieter mit VPC-Konfiguration erstellt.

```
MyCapacityProvider:
  Type: AWS::Serverless::CapacityProvider
  Properties:
    CapacityProviderName: my-capacity-provider
    VpcConfig:
      SubnetIds:
        - subnet-12345678
        - subnet-87654321
      SecurityGroupIds:
        - sg-12345678
    Tags:
      Environment: Production
      Team: ServerlessTeam
```

Erweiterter Kapazitätsanbieter mit Skalierung

Im folgenden Beispiel wird ein Kapazitätsanbieter mit benutzerdefinierten Instanzanforderungen und Skalierungskonfiguration erstellt.

```
AdvancedCapacityProvider:
  Type: AWS::Serverless::CapacityProvider
  Properties:
    CapacityProviderName: advanced-capacity-provider
    VpcConfig:
      SubnetIds:
        - subnet-12345678
        - subnet-87654321
      SecurityGroupIds:
        - sg-12345678
    OperatorRole: arn:aws:iam::123456789012:role/MyCapacityProviderRole
    PropagateTags: true
    InstanceRequirements:
      Architectures:
        - x86_64
      ExcludedTypes:
        - t2.micro
    ScalingConfig:
```

```
      MaxInstanceCount: 10
      ManualScalingPolicies:
        AverageCPUUtilization: 70.0
    KmsKeyArn: arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012
    LoggingConfig:
      LogGroup: /aws/lambda/capacity-provider/advanced-capacity-provider
      SystemLogLevel: INFO
    ManagedResourceTags:
      Tags:
        CostCenter: cc-1234
        Environment: Production
    Tags:
      Environment: Production
      CostCenter: Engineering
```

VpcConfig

Konfiguriert die VPC-Einstellungen für einen Kapazitätsanbieter, einschließlich der Subnetze und Sicherheitsgruppen, in denen EC2 Instances gestartet werden.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
SubnetIds: List
SecurityGroupIds: List
```

Eigenschaften

`SubnetIds`

Eine Liste der Subnetze IDs , in denen EC2 Instances gestartet werden. Es muss mindestens ein Subnetz angegeben werden.

Typ: Liste

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `SubnetIds` Eigenschaft `VpcConfig` einer `AWS::Lambda::CapacityProvider` Ressource übergeben.

`SecurityGroupIds`

Eine Liste der Sicherheitsgruppen IDs , die den EC2 Instanzen zugeordnet werden sollen. Wenn nicht angegeben, wird die Standardsicherheitsgruppe für die VPC verwendet.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `SecurityGroupIds` Eigenschaft oder `VpcConfig` Eigenschaft einer `AWS::Lambda::CapacityProvider` Ressource übergeben.

Beispiele

VPC-Konfiguration

Das folgende Beispiel zeigt eine VPC-Konfiguration mit mehreren Subnetzen und Sicherheitsgruppen.

```
VpcConfig:
  SubnetIds:
    - subnet-12345678
    - subnet-87654321
  SecurityGroupIds:
    - sg-12345678
    - sg-87654321
```

InstanceRequirements

Spezifiziert die Anforderungen für EC2 Instances, die vom Kapazitätsanbieter gestartet werden, einschließlich Architekturen und Einschränkungen des Instanztyps.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
Architectures: List
AllowedTypes: List
ExcludedTypes: List
```

> **Note**
>
> Sie können wählen, ob Sie `ExcludedTypes` bei der Definition der Instanzanforderungen für Ihren Kapazitätsanbieter entweder eines `AllowedTypes` oder angeben möchten, aber nicht beide.

Eigenschaften

`Architectures`

Die Befehlssatzarchitekturen für die Kapazitätsanbieter-Instances.

Zulässige Werte: `x86_64` oder `arm64`.

Typ: Liste

Required: No

Standardwert: `x86_64`

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Architectures` Eigenschaft `InstanceRequirements` einer `AWS::Lambda::CapacityProvider` Ressource übergeben.

`AllowedTypes`

Eine Liste der zulässigen EC2 Instanztypen für die Kapazitätsanbieter-Instanz.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `AllowedInstanceTypes` Eigenschaft `InstanceRequirements` einer `AWS::Lambda::CapacityProvider` Ressource übergeben.

`ExcludedTypes`

Eine Liste von EC2 Instance-Typen, die vom Kapazitätsanbieter ausgeschlossen werden sollen.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `ExcludedInstanceTypes` Eigenschaft `InstanceRequirements` einer `AWS::Lambda::CapacityProvider` Ressource übergeben.

Beispiele

Konfiguration der Instanzanforderungen

Das folgende Beispiel zeigt Instanzanforderungen mit spezifischen Einschränkungen für Architektur und Instanztyp.

```
InstanceRequirements:
  Architectures:
    - x86_64
  ExcludedTypes:
    - t2.micro
```

ScalingConfig

Konfiguriert, wie der Kapazitätsanbieter EC2 Instances je nach Bedarf skaliert, einschließlich maximaler Instanzlimits und Skalierungsrichtlinien.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
MaxVCpuCount: Integer
AverageCPUUtilization: Double
```

Eigenschaften

`MaxVCpuCount`

Die maximale Anzahl von vCPUs , die der Kapazitätsanbieter für alle Recheninstanzen bereitstellen kann.

Typ: Ganzzahl

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `MaxVCpuCount` Eigenschaft `CapacityProviderScalingConfig` einer `AWS::Lambda::CapacityProvider` Ressource übergeben.

`AverageCPUUtilization`

Der angestrebte durchschnittliche Prozentsatz der CPU-Auslastung (0-100) für Skalierungsentscheidungen. Wenn die durchschnittliche CPU-Auslastung diesen Schwellenwert überschreitet, skaliert der Kapazitätsanbieter EC2 Amazon-Instances. Falls angegeben, AWS SAM Konstrukte `CapacityProviderScalingConfig` einer `AWS::Lambda::CapacityProvider` Ressource mit den Werten „auf“ `'Manual'` und `ScalingMode` „`ScalingPolicies`gesetzt auf`[{PredefinedMetricType: 'LambdaCapacityProviderAverageCPUUtilization', TargetValue: <this value>}]`“.

Type: Double

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

Skalierungskonfiguration

Das folgende Beispiel zeigt eine Skalierungskonfiguration mit maximaler VCpu Anzahl und durchschnittlicher CPU-Auslastung.

```
ScalingConfig:
```

```
  MaxVCpuCount: 10
  AverageCPUUtilization: 70.0
```

ManagedResourceTags

Wendet explizite Tags auf die Amazon Elastic Compute Cloud-Instances und andere Ressourcen an, die der Kapazitätsanbieter in Ihrem Namen bereitstellt und verwaltet.

> **Note**
>
> Die `ManagedResourceTags` Eigenschaft steuert das Tagging der Amazon EC2-Instances und anderer Ressourcen, die Lambda für den Kapazitätsanbieter verwaltet. Diese Eigenschaft unterscheidet sich von der PropagateTags Eigenschaft, die angibt, ob die ??? Tags-Eigenschaft an die CloudFormation Ressourcen AWS SAM weitergegeben wird, die anhand Ihrer Vorlage AWS SAM generiert werden.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer AWS Serverless Application Model (AWS SAM) -Vorlage zu deklarieren.

YAML

```
Tags: Map
```

Eigenschaften

Die folgenden Eigenschaften werden für unterstütztManagedResourceTags.

`Tags`

Eine Zuordnung von Schlüssel-Wert-Paaren, die auf die Amazon EC2-Instances und andere Ressourcen angewendet werden sollen, die der Kapazitätsanbieter verwaltet.

Typ: Karte

Required: No

CloudFormation Kompatibilität: AWS SAM verwendet diese Eigenschaft, um die `PropagateTags` Eigenschaft einer `AWS::Lambda::CapacityProvider` Ressource zu konstruieren. AWS

SAM setzt `Mode` auf `Explicit` und konvertiert die Schlüssel-Wert-Paare in eine Liste von Tag-Objekten in. `ExplicitTags`

Beispiele

Wenden Sie explizite Tags auf verwaltete Ressourcen an

Im folgenden Beispiel wird ein expliziter Satz von Tags auf die Amazon EC2-Instances angewendet, die der Kapazitätsanbieter verwaltet.

```
ManagedResourceTags:
  Tags:
    CostCenter: cc-1234
    Environment: Production
```

AWS::Serverless::Connector

Konfiguriert Berechtigungen zwischen zwei Ressourcen. Eine Einführung in Konnektoren finden Sie unterVerwaltung von Ressourcenberechtigungen mit AWS SAM Steckverbinder.

Weitere Informationen zu generierten AWS CloudFormation Ressourcen finden Sie unterCloudFormation Ressourcen, die generiert werden, wenn Sie angeben AWS::Serverless::Connector.

Wenn Sie Feedback zu Konnektoren geben möchten, reichen Sie eine neue Ausgabe im serverless-application-model AWS GitHub Repository ein.

> **Note**
>
> Wenn Sie auf bereitstellen AWS CloudFormation, werden Ihre AWS SAM Ressourcen in AWS SAM CloudFormation Ressourcen umgewandelt. Weitere Informationen finden Sie unter Generiert CloudFormation Ressourcen für AWS SAM.

Syntax

Verwenden Sie eine der folgenden Syntaxen, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

> **Note**
>
> Für die meisten Anwendungsfälle empfehlen wir, die Syntax für eingebettete Konnektoren zu verwenden. Da sie in die Quellressource eingebettet ist, ist sie im Laufe der Zeit einfacher zu lesen und zu verwalten. Wenn Sie auf eine Quellressource verweisen müssen, die sich nicht in derselben AWS SAM Vorlage befindet, z. B. eine Ressource in einem verschachtelten Stapel oder eine gemeinsam genutzte Ressource, verwenden Sie die `AWS::Serverless::Connector` Syntax.

Integrierte Konnektoren

```
<source-resource-logical-id>:
  Connectors:
    <connector-logical-id:
      Properties:
        Destination: ResourceReference | List of ResourceReference
        Permissions: List
        SourceReference: SourceReference
```

AWS::Serverless::Connector

```
Type: AWS::Serverless::Connector
Properties:
  Destination: ResourceReference | List of ResourceReference
  Permissions: List
  Source: ResourceReference
```

Eigenschaften

`Destination`

Die Zielressource.

Typ: ResourceReference| Liste von ResourceReference

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Permissions`

Der Berechtigungstyp, den die Quellressource für die Zielressource ausführen darf.

`Read`beinhaltet AWS Identity and Access Management (IAM) -Aktionen, die das Lesen von Daten aus der Ressource ermöglichen.

`Write`beinhaltet IAM-Aktionen, die das Initiieren und Schreiben von Daten in eine Ressource ermöglichen.

Zulässige Werte: `Read` oder `Write`.

Typ: Liste

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein Äquivalent. CloudFormation

`Source`

Die Quellressource. Erforderlich, wenn die `AWS::Serverless::Connector` Syntax verwendet wird.

Typ: ResourceReference

Required: Conditional

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`SourceReference`

Die Quellressource.

> **Note**
>
> Wird zusammen mit der Syntax der eingebetteten Konnektoren verwendet, wenn Sie zusätzliche Eigenschaften für die Quellressource definieren.

Typ: SourceReference

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

Eingebettete Anschlüsse

Das folgende Beispiel verwendet eingebettete Konnektoren, um eine `Write` Datenverbindung zwischen einer AWS Lambda Funktion und einer Amazon DynamoDB-Tabelle zu definieren:

```
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyTable:
    Type: AWS::Serverless::SimpleTable
  MyFunction:
    Type: AWS::Serverless::Function
    Connectors:
      MyConn:
        Properties:
          Destination:
            Id: MyTable
          Permissions:
            - Write
    ...
```

Im folgenden Beispiel werden eingebettete Konnektoren verwendet, um Berechtigungen zu definieren`Read`: `Write`

```
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Connectors:
      MyConn:
        Properties:
          Destination:
            Id: MyTable
          Permissions:
            - Read
            - Write
```

```
  MyTable:
    Type: AWS::DynamoDB::Table
    ...
```

Im folgenden Beispiel werden eingebettete Konnektoren verwendet, um eine Quellressource mit einer anderen Eigenschaft als zu definieren`Id`:

```
Transform: AWS::Serverless-2016-10-31
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyApi:
    Type: AWS::Serverless::Api
    Connectors:
      ApitoLambdaConn:
        Properties:
          SourceReference:
            Qualifier: Prod/GET/foobar
          Destination:
            Id: MyTable
          Permissions:
            - Read
            - Write
  MyTable:
    Type: AWS::DynamoDB::Table
    ...
```

AWS::Serverless::Connector

Im folgenden Beispiel wird die AWS::Serverless::Connector Ressource verwendet, damit eine AWS Lambda Funktion aus einer Amazon DynamoDB-Tabelle liest und in diese schreibt:

```
MyConnector:
  Type: AWS::Serverless::Connector
  Properties:
    Source:
      Id: MyFunction
    Destination:
      Id: MyTable
    Permissions:
      - Read
      - Write
```

Im folgenden Beispiel wird die AWS::Serverless::Connector Ressource verwendet, damit eine Lambda-Funktion in ein Amazon SNS SNS-Thema schreibt, wobei sich beide Ressourcen in derselben Vorlage befinden:

```
MyConnector:
  Type: AWS::Serverless::Connector
  Properties:
    Source:
      Id: MyLambda
    Destination:
      Id: MySNSTopic
    Permissions:
      - Write
```

Im folgenden Beispiel wird die AWS::Serverless::Connector Ressource verwendet, um ein Amazon SNS SNS-Thema in eine Lambda-Funktion schreiben zu lassen, die dann in eine Amazon DynamoDB-Tabelle schreibt, wobei sich alle Ressourcen in derselben Vorlage befinden:

```
Transform: AWS::Serverless-2016-10-31
Resources:
  Topic:
    Type: AWS::SNS::Topic
    Properties:
      Subscription:
        - Endpoint: !GetAtt Function.Arn
          Protocol: lambda

  Function:
    Type: AWS::Serverless::Function
    Properties:
      Runtime: nodejs16.x
      Handler: index.handler
      InlineCode: |
        const AWS = require('aws-sdk');
        exports.handler = async (event, context) => {
          const docClient = new AWS.DynamoDB.DocumentClient();
          await docClient.put({
            TableName: process.env.TABLE_NAME,
            Item: {
              id: context.awsRequestId,
              event: JSON.stringify(event)
            }
```

```
          }).promise();
        };
      Environment:
        Variables:
          TABLE_NAME: !Ref Table

  Table:
    Type: AWS::Serverless::SimpleTable

  TopicToFunctionConnector:
    Type: AWS::Serverless::Connector
    Properties:
      Source:
        Id: Topic
      Destination:
        Id: Function
      Permissions:
        - Write

  FunctionToTableConnector:
    Type: AWS::Serverless::Connector
    Properties:
      Source:
        Id: Function
      Destination:
        Id: Table
      Permissions:
        - Write
```

Im Folgenden sehen Sie die transformierte AWS CloudFormation Vorlage aus dem obigen Beispiel:

```
"FunctionToTableConnectorPolicy": {
  "Type": "AWS::IAM::ManagedPolicy",
  "Metadata": {
    "aws:sam:connectors": {
      "FunctionToTableConnector": {
        "Source": {
          "Type": "AWS::Lambda::Function"
        },
        "Destination": {
          "Type": "AWS::DynamoDB::Table"
        }
      }
```

```
    }
  },
  "Properties": {
    "PolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": [
            "dynamodb:PutItem",
            "dynamodb:UpdateItem",
            "dynamodb:DeleteItem",
            "dynamodb:BatchWriteItem",
            "dynamodb:PartiQLDelete",
            "dynamodb:PartiQLInsert",
            "dynamodb:PartiQLUpdate"
          ],
          "Resource": [
            {
              "Fn::GetAtt": [
                "MyTable",
                "Arn"
              ]
            },
            {
              "Fn::Sub": [
                "${DestinationArn}/index/*",
                {
                  "DestinationArn": {
                    "Fn::GetAtt": [
                      "MyTable",
                      "Arn"
                    ]
                  }
                }
              ]
            }
          ]
        }
      ]
    },
    "Roles": [
      {
        "Ref": "MyFunctionRole"
```

```
      }
    ]
  }
}
```

ResourceReference

Ein Verweis auf eine Ressource, die der AWS::Serverless::Connector Ressourcentyp verwendet.

> **Note**
>
> Geben Sie für Ressourcen in derselben Vorlage den an`Id`. Verwenden Sie für Ressourcen, die nicht in derselben Vorlage enthalten sind, eine Kombination anderer Eigenschaften. Weitere Informationen finden Sie unter AWS SAM Steckverbinderreferenz.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
Arn: String
Id: String
Name: String
Qualifier: String
QueueUrl: String
ResourceId: String
RoleName: String
Type: String
```

Eigenschaften

`Arn`

Der ARN einer Ressource.

Typ: Zeichenfolge

Required: Conditional

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Id`

Die logische ID einer Ressource in derselben Vorlage.

> **Note**
>
> Wenn angegeben, `Id` wird, wenn der Connector AWS Identity and Access Management (IAM-) Richtlinien generiert, die diesen Richtlinien zugeordnete IAM-Rolle aus der Ressource abgeleitet. `Id` Wenn nicht angegeben, `Id` wird die Ressource für `RoleName` Connectors bereitgestellt, um generierte IAM-Richtlinien an eine IAM-Rolle anzuhängen.

Typ: Zeichenfolge

Required: Conditional

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Name`

Der Name einer -Ressource.

Typ: Zeichenfolge

Required: Conditional

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Qualifier`

Ein Qualifikator für eine Ressource, der ihren Umfang einschränkt. `Qualifier`ersetzt den `*` Wert am Ende einer Ressourcenbeschränkung ARN. Ein Beispiel finden Sie unter API Gateway, das eine Lambda-Funktion aufruft.

> **Note**
>
> Die Definition des Qualifikators variiert je nach Ressourcentyp. Eine Liste der unterstützten Quell- und Zielressourcentypen finden Sie unterAWS SAM Steckverbinderreferenz.

Typ: Zeichenfolge

Required: Conditional

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`QueueUrl`

Die URL der Amazon SQS SQS-Warteschlange. Diese Eigenschaft gilt nur für Amazon SQS SQS-Ressourcen.

Typ: Zeichenfolge

Required: Conditional

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`ResourceId`

Die ID einer Ressource. Zum Beispiel die API-Gateway-API-ID.

Typ: Zeichenfolge

Required: Conditional

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`RoleName`

Der Rollenname, der einer Ressource zugeordnet ist.

> **Note**
>
> Wenn angegeben, `Id` wird, wenn der Connector IAM-Richtlinien generiert, die diesen Richtlinien zugeordnete IAM-Rolle aus der Ressource abgeleitet. `Id` Wenn nicht

angegeben, `Id` wird die Ressource für Connectoren bereitgestelltRoleName, um generierte IAM-Richtlinien an eine IAM-Rolle anzuhängen.

Typ: Zeichenfolge

Required: Conditional

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Type`

Der CloudFormation Typ einer Ressource. Weitere Informationen finden Sie in der Referenz zu AWS Ressourcen- und Eigenschaftstypen.

Typ: Zeichenfolge

Required: Conditional

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

API Gateway, das eine Lambda-Funktion aufruft

Das folgende Beispiel verwendet die AWS::Serverless::Connector Ressource, um Amazon API Gateway das Aufrufen einer AWS Lambda Funktion zu ermöglichen.

YAML

```
Transform: AWS::Serverless-2016-10-31
Resources:
  MyRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Effect: Allow
            Action: sts:AssumeRole
            Principal:
              Service: lambda.amazonaws.com
```

```
      ManagedPolicyArns:
        - !Sub arn:${AWS::Partition}:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole

  MyFunction:
    Type: AWS::Lambda::Function
    Properties:
      Role: !GetAtt MyRole.Arn
      Runtime: nodejs16.x
      Handler: index.handler
      Code:
        ZipFile: |
          exports.handler = async (event) => {
            return {
              statusCode: 200,
              body: JSON.stringify({
                "message": "It works!"
              }),
            };
          };

  MyApi:
    Type: AWS::ApiGatewayV2::Api
    Properties:
      Name: MyApi
      ProtocolType: HTTP

  MyStage:
    Type: AWS::ApiGatewayV2::Stage
    Properties:
      ApiId: !Ref MyApi
      StageName: prod
      AutoDeploy: True

  MyIntegration:
    Type: AWS::ApiGatewayV2::Integration
    Properties:
      ApiId: !Ref MyApi
      IntegrationType: AWS_PROXY
      IntegrationUri: !Sub arn:aws:apigateway:${AWS::Region}:lambda:path/2015-03-31/
functions/${MyFunction.Arn}/invocations
      IntegrationMethod: POST
      PayloadFormatVersion: "2.0"
```

```
  MyRoute:
    Type: AWS::ApiGatewayV2::Route
    Properties:
      ApiId: !Ref MyApi
      RouteKey: GET /hello
      Target: !Sub integrations/${MyIntegration}

  MyConnector:
    Type: AWS::Serverless::Connector
    Properties:
      Source: # Use 'Id' when resource is in the same template
        Type: AWS::ApiGatewayV2::Api
        ResourceId: !Ref MyApi
        Qualifier: prod/GET/hello # Or "*" to allow all routes
      Destination: # Use 'Id' when resource is in the same template
        Type: AWS::Lambda::Function
        Arn: !GetAtt MyFunction.Arn
      Permissions:
        - Write

Outputs:
  Endpoint:
    Value: !Sub https://${MyApi}.execute-api.${AWS::Region}.${AWS::URLSuffix}/prod/
hello
```

SourceReference

Ein Verweis auf eine Quellressource, die der AWS::Serverless::Connector Ressourcentyp verwendet.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
Qualifier: String
```

Eigenschaften

`Qualifier`

Ein Qualifikator für eine Ressource, der ihren Geltungsbereich einschränkt. `Qualifier`ersetzt den * Wert am Ende einer Ressourcenbeschränkung ARN.

> **Note**
>
> Die Definition des Qualifikators variiert je nach Ressourcentyp. Eine Liste der unterstützten Quell- und Zielressourcentypen finden Sie unterAWS SAM Steckverbinderreferenz.

Typ: Zeichenfolge

Required: Conditional

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

Im folgenden Beispiel werden eingebettete Konnektoren verwendet, um eine Quellressource mit einer anderen Eigenschaft als zu definieren**`Id`**:

```
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyApi:
    Type: AWS::Serverless::Api
    Connectors:
      ApitoLambdaConn:
        Properties:
          SourceReference:
            Qualifier: Prod/GET/foobar
          Destination:
            Id: MyTable
          Permissions:
            - Read
            - Write
  MyTable:
```

```
    Type: AWS::DynamoDB::Table
    ...
```

AWS::Serverless: :Funktion

Erstellt eine AWS Lambda Funktion, eine AWS Identity and Access Management (IAM-) Ausführungsrolle und Ereignisquellenzuordnungen, die die Funktion auslösen.

Die AWS::Serverless: :Funktion Ressource unterstützt auch das `Metadata` Ressourcenattribut, sodass Sie anweisen können, benutzerdefinierte Laufzeiten AWS SAM zu erstellen, die Ihre Anwendung benötigt. Weitere Informationen zum Erstellen benutzerdefinierter Laufzeiten finden Sie unter. Erstellen von Lambda-Funktionen mit benutzerdefinierten Laufzeiten in AWS SAM

> **Note**
>
> Wenn Sie es bereitstellen AWS CloudFormation, AWS SAM wandelt es Ihre AWS SAM Ressourcen in CloudFormation Ressourcen um. Weitere Informationen finden Sie unter Generiert CloudFormation Ressourcen für AWS SAM.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer AWS Serverless Application Model (AWS SAM) -Vorlage zu deklarieren.

YAML

```
Type: AWS::Serverless::Function
Properties:
  Architectures: List
  AssumeRolePolicyDocument: JSON
  AutoPublishAlias: String
  AutoPublishAliasAllProperties: Boolean
  AutoPublishCodeSha256: String
  CapacityProviderConfig: CapacityProviderConfig
  CodeSigningConfigArn: String
  CodeUri: String | FunctionCode
  DeadLetterQueue: Map | DeadLetterQueue
  DeploymentPreference: DeploymentPreference
  Description: String
  DurableConfig: DurableConfig
```

```
  Environment: Environment
  EphemeralStorage: EphemeralStorage
  EventInvokeConfig: EventInvokeConfiguration
  Events: EventSource
  FileSystemConfigs: List
  FunctionName: String
  FunctionScalingConfig: FunctionScalingConfig
  FunctionUrlConfig: FunctionUrlConfig
  Handler: String
  ImageConfig: ImageConfig
  ImageUri: String
  InlineCode: String
  KmsKeyArn: String
  Layers: List
  LoggingConfig: LoggingConfig
  MemorySize: Integer
  PackageType: String
  PermissionsBoundary: String
  Policies: String | List | Map
  PublishToLatestPublished: Boolean
  PropagateTags: Boolean
  ProvisionedConcurrencyConfig: ProvisionedConcurrencyConfig
  RecursiveLoop: String
  ReservedConcurrentExecutions: Integer
  Role: String
  RolePath: String
  Runtime: String
  RuntimeManagementConfig: RuntimeManagementConfig
  SnapStart: SnapStart
  SourceKMSKeyArn: String
  Tags: Map
  TenancyConfig: TenancyConfig
  Timeout: Integer
  Tracing: String
  VersionDescription: String
  VersionDeletionPolicy: String
  VpcConfig: VpcConfig
```

Eigenschaften

`Architectures`

Die Befehlssatz-Architektur für die Funktion.

Weitere Informationen zu dieser Eigenschaft finden Sie unter Lambda-Befehlssatzarchitekturen im AWS Lambda Entwicklerhandbuch.

Gültige Werte: Einer von oder `x86_64` `arm64`

Typ: Liste

Required: No

Standardwert: `x86_64`

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Architectures` Eigenschaft einer `AWS::Lambda::Function` Ressource übergeben.

`AssumeRolePolicyDocument`

Fügt ein AssumeRolePolicyDocument für die Standardeinstellung hinzu, die `Role` für diese Funktion erstellt wurde. Wenn diese Eigenschaft nicht angegeben ist, AWS SAM fügt eine standardmäßige Übernahmerolle für diese Funktion hinzu.

Type: JSON

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `AssumeRolePolicyDocument` Eigenschaft einer `AWS::IAM::Role` Ressource. AWS SAM fügt diese Eigenschaft der generierten IAM-Rolle für diese Funktion hinzu. Wenn der Amazon-Ressourcenname (ARN) einer Rolle für diese Funktion angegeben wird, bewirkt diese Eigenschaft nichts.

`AutoPublishAlias`

Der Name des Lambda-Alias. Weitere Informationen zu Lambda-Aliasen finden Sie unter Lambda-Funktionsaliase im Entwicklerhandbuch. AWS Lambda Beispiele, die diese Eigenschaft verwenden, finden Sie unter. Schrittweise Bereitstellung serverloser Anwendungen mit AWS SAM

AWS SAM generiert AWS::Lambda::Version und nutzt AWS::Lambda::Alias Ressourcen, wenn diese Eigenschaft festgelegt ist. Hinweise zu diesem Szenario finden Sie unterAutoPublishAlias Eigenschaft ist angegeben. Allgemeine Informationen zu generierten CloudFormation Ressourcen finden Sie unterGeneriert CloudFormation Ressourcen für AWS SAM.

> **Versionsveröffentlichung und intrinsische Funktionen**
>
> AWS SAM bestimmt, ob eine neue Version veröffentlicht werden soll, indem Eigenschaftswerte in der Vorlage zwischen Bereitstellungen verglichen werden. AWS

SAM Vergleicht bei Werten, die über systeminterne Funktionen bereitgestellt werden, die ungelöste Referenz, nicht den aufgelösten Wert. Ein Beispiel ist eine Umgebungsvariable, die auf einen Vorlagenparameter `!Ref` gesetzt ist.
Wenn eine Bereitstellung nur den Wert eines Stackparameters ändert, bleibt die Vorlage selbst unverändert. In diesem Fall AWS SAM wird keine neue Version veröffentlicht und der Alias nicht aktualisiert. AWS SAM wendet die Änderung nur auf die `$LATEST` Version der Funktion an. Problemumgehungen finden Sie unter Beispiel mit. AWS::LanguageExtensions

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`AutoPublishAliasAllProperties`

Gibt an, wann ein neues erstellt `AWS::Lambda::Version` wird. Wenn`true`, wird eine neue Lambda-Version erstellt, wenn eine Eigenschaft in der Lambda-Funktion geändert wird. In `false` diesem Fall wird eine neue Lambda-Version nur erstellt, wenn eine der folgenden Eigenschaften geändert wird:

- `Environment`, `MemorySize`, oder `SnapStart`.
- Jede Änderung, die zu einer Aktualisierung der `Code` Eigenschaft führt, z. B. `CodeDictImageUri`, oder`InlineCode`.

Parameteränderungen und Versionsveröffentlichung

AWS SAM veröffentlicht nur dann eine neue Version, wenn sich der Wert einer Eigenschaft in der Vorlage ändert. Wenn ein Eigenschaftswert aus einer systeminternen Funktion stammt, wird die AWS SAM ungelöste Referenz angezeigt. Ein Beispiel ist eine Umgebungsvariable, die als Vorlagenparameter `!Ref` definiert ist. Bereitstellungen, die nur den Wert des Parameters ändern, veröffentlichen keine neue Version, auch nicht mit`AutoPublishAliasAllProperties: true`. AWS SAM wendet die Änderung `$LATEST` nur auf an.

Verwenden Sie eine der folgenden Problemumgehungen, um die Veröffentlichung einer neuen Version AWS SAM zu erzwingen, wenn sich nur ein Parameterwert ändert:

- Standard AWS SAM und CloudFormation Bereitstellungen: Fügen Sie die AWS::LanguageExtensions Transformation zuvor `AWS::Serverless-2016-10-31` im Abschnitt Ihrer Vorlage hinzu. `Transform` CloudFormation löst Verweise auf ihre tatsächlichen Werte auf, bevor der AWS SAM Versions-Hash berechnet wird. Eine Änderung des Parameterwerts veröffentlicht dann eine neue Version und aktualisiert den Alias:

```
Transform:
  - AWS::LanguageExtensions
  - AWS::Serverless-2016-10-31
```

- AWS Serverless Application Repository Anwendungen: Der AWS Serverless Application Repository unterstützt nur die `AWS::Serverless-2016-10-31` Transformation und lehnt `AWS::LanguageExtensions` deklarierte Vorlagen ab. Für Anwendungen, die Sie über den veröffentlichen AWS Serverless Application Repository, geben Sie den Wert direkt als Literal in der Vorlage an, anstatt auf einen Parameter zu verweisen. Dadurch wird die Änderung sichtbar, wenn der AWS SAM Versions-Hash berechnet wird.

Diese Eigenschaft `AutoPublishAlias` muss definiert werden.

Wenn `AutoPublishCodeSha256` auch angegeben, hat ihr Verhalten Vorrang `AutoPublishAliasAllProperties: true` vor.

Typ: Boolesch

Required: No

Standardwert: `false`

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und es gibt keine CloudFormation Entsprechung.

`AutoPublishCodeSha256`

Wenn diese Zeichenfolge verwendet wird, verwendet sie den `CodeUri` Wert, um zu ermitteln, ob eine neue Lambda-Version veröffentlicht werden muss. Diese Eigenschaft wird häufig verwendet, um das folgende Bereitstellungsproblem zu lösen: Ein Bereitstellungspaket wird an einem Amazon S3-Standort gespeichert und durch ein neues Bereitstellungspaket mit aktualisiertem Lambda-Funktionscode ersetzt, aber die `CodeUri` Eigenschaft bleibt unverändert (im Gegensatz

dazu, dass das neue Bereitstellungspaket an einen neuen Amazon S3-Standort hochgeladen und dann an den neuen Standort geändert `CodeUri` wird).

Dieses Problem wird durch eine AWS SAM Vorlage mit den folgenden Merkmalen gekennzeichnet:

- Das `DeploymentPreference` Objekt ist für schrittweise Bereitstellungen konfiguriert (wie unter beschriebenSchrittweise Bereitstellung serverloser Anwendungen mit AWS SAM)
- Die `AutoPublishAlias` Eigenschaft ist festgelegt und ändert sich nicht zwischen Bereitstellungen
- Die `CodeUri` Eigenschaft ist festgelegt und ändert sich nicht zwischen Bereitstellungen.

In diesem Szenario `AutoPublishCodeSha256` führt die Aktualisierung dazu, dass eine neue Lambda-Version erfolgreich erstellt wird. Neuer Funktionscode, der in Amazon S3 bereitgestellt wird, wird jedoch nicht erkannt. Um neuen Funktionscode zu erkennen, sollten Sie die Versionierung in Ihrem Amazon S3-Bucket in Betracht ziehen. Geben Sie die `Version` Eigenschaft für Ihre Lambda-Funktion an und konfigurieren Sie Ihren Bucket so, dass immer das neueste Bereitstellungspaket verwendet wird.

In diesem Szenario müssen Sie einen eindeutigen Wert für `AutoPublishCodeSha256` angeben, um die schrittweise Bereitstellung erfolgreich auszulösen.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`CapacityProviderConfig`

Konfiguriert den Kapazitätsanbieter, an den veröffentlichte Versionen der Funktion angehängt werden. Dadurch kann die Funktion auf kundeneigenen EC2-Instances ausgeführt werden, die von Lambda Managed Instances verwaltet werden.

Typ: CapacityProviderConfig

Required: No

CloudFormation Kompatibilität: SAM reduziert die an die Eigenschaft einer `AWS::Lambda::Function` Ressource übergebene Eigenschaft und rekonstruiert die `CapacityProviderConfig` verschachtelte Struktur.

`CodeSigningConfigArn`

Der ARN der AWS::Lambda::CodeSigningConfig Ressource, der verwendet wird, um die Codesignatur für diese Funktion zu aktivieren. Weitere Informationen zur Codesignatur finden Sie unter Richten Sie die Codesignatur für Ihre AWS SAM Anwendung ein.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `CodeSigningConfigArn` Eigenschaft einer `AWS::Lambda::Function` Ressource übergeben.

`CodeUri`

Der Code für die Funktion. Zu den akzeptierten Werten gehören:

- Die Amazon S3-URI der Funktion. Beispiel, `s3://bucket-123456789/sam-app/1234567890abcdefg`.
- Der lokale Pfad zur Funktion. Beispiel, `hello_world/`.
- Ein FunctionCode-Objekt.

> **Note**
>
> Wenn Sie die Amazon S3-URI oder das Amazon FunctionCode S3-Objekt einer Funktion angeben, müssen Sie auf ein gültiges Lambda-Bereitstellungspaket verweisen.
> Wenn Sie einen lokalen Dateipfad angeben, verwenden Sie den, AWS SAMCLI um die lokale Datei bei der Bereitstellung hochzuladen. Weitere Informationen hierzu finden Sie unter So werden lokale Dateien bei der Bereitstellung AWS SAM hochgeladen.
> Wenn Sie intrinsische Funktionen in `CodeUri` Property verwenden, können AWS SAM Sie die Werte nicht korrekt analysieren. Erwägen Sie stattdessen die Verwendung von transform AWS::LanguageExtensions .

Typ: [Zeichenfolge |FunctionCode]

Erforderlich: Bedingt. Wenn auf gesetzt `PackageType` ist`Zip`, `InlineCode` ist eines von `CodeUri` oder erforderlich.

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `Code` Eigenschaft einer `AWS::Lambda::Function` Ressource. Die verschachtelten Amazon S3-Eigenschaften sind unterschiedlich benannt.

`DeadLetterQueue`

Konfiguriert ein Amazon Simple Notification Service (Amazon SNS) -Thema oder eine Amazon Simple Queue Service (Amazon SQS) -Warteschlange, in die Lambda Ereignisse sendet, die es nicht verarbeiten kann. Weitere Informationen zur Funktion von Warteschlangen mit unlesbaren Buchstaben finden Sie im Entwicklerhandbuch unter Dead-letter Warteschlangen. AWS Lambda

> **Note**
>
> Wenn die Ereignisquelle Ihrer Lambda-Funktion eine Amazon SQS-Warteschlange ist, konfigurieren Sie eine Warteschlange mit unzustellbaren Buchstaben für die Quellwarteschlange, nicht für die Lambda-Funktion. Die Warteschlange mit leeren Buchstaben, die Sie für eine Funktion konfigurieren, wird für die asynchrone Aufruf-Warteschlange der Funktion verwendet, nicht für Warteschlangen mit Ereignisquellen.

Typ: Karte | DeadLetterQueue

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `DeadLetterConfig` Eigenschaft einer `AWS::Lambda::Function` Ressource. In wird CloudFormation der Typ von dem abgeleitet`TargetArn`, wohingegen AWS SAM Sie den Typ zusammen mit dem übergeben müssen`TargetArn`.

`DeploymentPreference`

Die Einstellungen zur Aktivierung schrittweiser Lambda-Bereitstellungen.

Wenn ein `DeploymentPreference` Objekt angegeben ist, AWS SAM erstellt es ein Aufgerufenes Objekt `ServerlessDeploymentApplication` (AWS::CodeDeploy::Applicationeines pro Stapel), ein Aufgerufenes und ein AWS::CodeDeploy::DeploymentGroup *`<function-logical-id>`*`DeploymentGroup` Aufgerufenes Objekt AWS::IAM::Role. `CodeDeployServiceRole`

Typ: DeploymentPreference

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Siehe auch: Weitere Informationen zu dieser Eigenschaft finden Sie unterSchrittweise Bereitstellung serverloser Anwendungen mit AWS SAM.

`Description`

Eine Beschreibung der Funktion.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Description` Eigenschaft einer `AWS::Lambda::Function` Ressource übergeben.

`DurableConfig`

Konfiguration für langlebige Funktionen. Ermöglicht die statusbehaftete Ausführung mit automatischen Checkpointing- und Wiedergabefunktionen.

Typ: DurableConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und es gibt keine Entsprechung. CloudFormation

`Environment`

Die Konfiguration für die Laufzeitumgebung.

Typ: Umgebung

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Environment` Eigenschaft einer `AWS::Lambda::Function` Ressource übergeben.

`EphemeralStorage`

Ein Objekt, das den Speicherplatz in MB angibt, der Ihrer Lambda-Funktion in `/tmp` zur Verfügung steht.

Weitere Informationen zu dieser Eigenschaft finden Sie unter Lambda-Ausführungsumgebung im AWS Lambda Entwicklerhandbuch.

Typ: EphemeralStorage

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `EphemeralStorage` Eigenschaft einer `AWS::Lambda::Function` Ressource übergeben.

`EventInvokeConfig`

Das Objekt, das die Konfiguration des Ereignisaufrufs für eine Lambda-Funktion beschreibt.

Typ: EventInvokeConfiguration

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Events`

Gibt die Ereignisse an, die diese Funktion auslösen. Ereignisse bestehen aus einem Typ und einer Reihe von Eigenschaften, die vom Typ abhängen.

Typ: EventSource

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und es gibt keine CloudFormation Entsprechung.

`FileSystemConfigs`

Liste von FileSystemConfig Objekten, die die Verbindungseinstellungen für ein Amazon Elastic File System (Amazon EFS) -Dateisystem oder ein Amazon S3 Files-Dateisystem angeben. Sie können entweder einen Amazon EFS-Zugangspunkt oder einen S3 Files-Zugriffspunkt anhängen, aber nicht beide.

Jedes `FileSystemConfig` Objekt enthält einen `Arn` (den Zugriffspunkt-ARN) und einen `LocalMountPath` (den Pfad, in dem das Dateisystem in der Funktion gemountet ist). Für Amazon EFS ist der ARN ein Amazon EFS-Zugriffspunkt-ARN. Für S3-Dateien ist das ARN ein `AWS::S3Files::AccessPoint` ARN.

Wenn Ihre Vorlage eine AWS::EFS::MountTarget Ressource enthält, müssen Sie auch ein `DependsOn` Ressourcenattribut angeben, um sicherzustellen, dass das Mount-Ziel vor der Funktion erstellt oder aktualisiert wird. Ebenso müssen Sie, wenn Ihre Vorlage eine `AWS::S3Files::MountTarget` Ressource enthält, ein `DependsOn` Attribut für das S3-Dateien-Mount-Ziel angeben.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `FileSystemConfigs` Eigenschaft einer `AWS::Lambda::Function` Ressource übergeben.

`FunctionName`

Ein Name für die Funktion. Wenn Sie keinen Namen angeben, wird ein eindeutiger Name für Sie generiert.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `FunctionName` Eigenschaft einer `AWS::Lambda::Function` Ressource übergeben.

`FunctionScalingConfig`

Konfiguriert das Skalierungsverhalten für Lambda-Funktionen, die auf Kapazitätsanbietern ausgeführt werden. Definiert die minimale und maximale Anzahl von Ausführungsumgebungen.

Typ: FunctionScalingConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `FunctionScalingConfig` Eigenschaft einer `AWS::Lambda::Function` Ressource übergeben.

`FunctionUrlConfig`

Das Objekt, das eine Funktions-URL beschreibt. Eine Funktions-URL ist ein HTTPS-Endpunkt, den Sie verwenden können, um Ihre Funktion aufzurufen.

Weitere Informationen finden Sie unter Funktions-URLs im AWS Lambda Entwicklerhandbuch.

Typ: FunctionUrlConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und es gibt keine CloudFormation Entsprechung.

`Handler`

Die Funktion in Ihrem Code, die aufgerufen wird, um mit der Ausführung zu beginnen. Diese Eigenschaft ist nur erforderlich, wenn die `PackageType` Eigenschaft auf festgelegt ist`Zip`.

Typ: Zeichenfolge

Required: Conditional

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Handler` Eigenschaft einer `AWS::Lambda::Function` Ressource übergeben.

`ImageConfig`

Das Objekt, das zur Konfiguration der Lambda-Container-Image-Einstellungen verwendet wird. Weitere Informationen finden Sie im AWS Lambda Entwicklerhandbuch unter Verwenden von Container-Images mit Lambda.

Typ: ImageConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `ImageConfig` Eigenschaft einer `AWS::Lambda::Function` Ressource übergeben.

`ImageUri`

Die URI des Amazon Elastic Container Registry (Amazon ECR) -Repositorys für das Container-Image der Lambda-Funktion. Diese Eigenschaft gilt nur, wenn die `PackageType` Eigenschaft auf gesetzt ist`Image`, andernfalls wird sie ignoriert. Weitere Informationen finden Sie unter Verwenden von Container-Images mit Lambda im AWS Lambda Entwicklerhandbuch.

> **Note**
>
> Wenn die `PackageType` Eigenschaft auf festgelegt ist`Image`, `ImageUri` ist entweder eine erforderlich, oder Sie müssen Ihre Anwendung mit den erforderlichen `Metadata`

Einträgen in der AWS SAM Vorlagendatei erstellen. Weitere Informationen finden Sie unter Standard-Build mit AWS SAM.

Wenn Sie Ihre Anwendung mit den erforderlichen `Metadata` Einträgen erstellen`ImageUri`, hat dies Vorrang. Wenn Sie also beide angeben, `ImageUri` wird dies ignoriert.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `ImageUri` Eigenschaft des `AWS::Lambda::Function` `Code` Datentyps übergeben.

`InlineCode`

Der Lambda-Funktionscode, der direkt in die Vorlage geschrieben ist. Diese Eigenschaft gilt nur, wenn die `PackageType` Eigenschaft auf gesetzt ist`Zip`, andernfalls wird sie ignoriert.

> **Note**
>
> Wenn die `PackageType` Eigenschaft auf `Zip` (Standard) gesetzt ist, `InlineCode` ist eine von `CodeUri` oder erforderlich.

Typ: Zeichenfolge

Required: Conditional

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `ZipFile` Eigenschaft des `AWS::Lambda::Function` `Code` Datentyps übergeben.

`KmsKeyArn`

Der ARN eines AWS Key Management Service (AWS KMS) -Schlüssels, den Lambda zum Verschlüsseln und Entschlüsseln der Umgebungsvariablen Ihrer Funktion verwendet.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `KmsKeyArn` Eigenschaft einer Ressource übergeben. `AWS::Lambda::Function`

`Layers`

Die Liste der `LayerVersion` ARNs, die diese Funktion verwenden sollte. Die hier angegebene Reihenfolge ist die Reihenfolge, in der sie importiert werden, wenn die Lambda-Funktion ausgeführt wird. Die Version ist entweder ein vollständiger ARN einschließlich der Version oder ein Verweis auf eine LayerVersion Ressource. Zum Beispiel `LayerVersion` wird ein Verweis auf a sein, `!Ref MyLayer` während es sich um einen vollständigen ARN einschließlich der Version handelt`arn:aws:lambda:`*`region`*`:`*`account-id`*`:layer:`*`layer-name`*`:`*`version`*.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Layers` Eigenschaft einer `AWS::Lambda::Function` Ressource übergeben.

`LoggingConfig`

Die Amazon CloudWatch Logs-Konfigurationseinstellungen der Funktion.

Typ: LoggingConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `LoggingConfig` Eigenschaft einer `AWS::Lambda::Function` Ressource übergeben.

`MemorySize`

Die Größe des Speichers in MB, der pro Aufruf der Funktion zugewiesen wird.

Typ: Ganzzahl

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `MemorySize` Eigenschaft einer `AWS::Lambda::Function` Ressource übergeben.

`PackageType`

Der Bereitstellungspakettyp der Lambda-Funktion. Weitere Informationen finden Sie unter Lambda-Bereitstellungspakete im AWS Lambda Entwicklerhandbuch.

Hinweise:

1. Wenn diese Eigenschaft auf `Zip` (Standard) gesetzt ist, `InlineCode` gilt entweder `CodeUri` oder und `ImageUri` wird ignoriert.

2. Wenn diese Eigenschaft auf gesetzt ist`Image`, `ImageUri` gilt nur, `CodeUri` und beide `InlineCode` werden ignoriert. Das Amazon ECR-Repository, das zum Speichern des Container-Images der Funktion erforderlich ist, kann von der AWS SAMCLI automatisch erstellt werden. Weitere Informationen finden Sie unter sam deploy.

Zulässige Werte: `Zip` oder `Image`.

Typ: Zeichenfolge

Required: No

Standardwert: `Zip`

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `PackageType` Eigenschaft einer `AWS::Lambda::Function` Ressource übergeben.

`PermissionsBoundary`

Der ARN einer Berechtigungsgrenze, die für die Ausführungsrolle dieser Funktion verwendet werden soll. Diese Eigenschaft funktioniert nur, wenn die Rolle für Sie generiert wurde.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `PermissionsBoundary` Eigenschaft einer `AWS::IAM::Role` Ressource übergeben.

`Policies`

Berechtigungsrichtlinien für diese Funktion. Richtlinien werden an die Standardausführungsrolle AWS Identity and Access Management (IAM) der Funktion angehängt.

Diese Eigenschaft akzeptiert einen einzelnen Wert oder eine Liste von Werten. Gültige Werte sind:

- AWS SAM Richtlinienvorlagen.

- Die ARN einer AWS verwalteten Richtlinie oder einer vom Kunden verwalteten Richtlinie.
- Der Name einer AWS verwalteten Richtlinie aus der folgenden Liste.
- Eine Inline-IAM-Richtlinie, die YAML als Map formatiert ist.

> **Note**
>
> Wenn Sie die `Role` Eigenschaft festlegen, wird diese Eigenschaft ignoriert.

Typ: Zeichenfolge | Liste | Karte

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `Policies` Eigenschaft einer `AWS::IAM::Role` Ressource.

`PublishToLatestPublished`

Gibt an, ob die neueste Funktionsversion veröffentlicht werden soll, wenn die Funktion aktualisiert wird.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `PublishToLatestPublished` Eigenschaft einer `AWS::Lambda::Function` Ressource übergeben.

`PropagateTags`

Geben Sie an, ob Tags von der `Tags` Eigenschaft an Ihre AWS::Serverless::Function generierten Ressourcen weitergegeben werden sollen oder nicht. Geben Sie `True` an, dass Tags in Ihren generierten Ressourcen weitergegeben werden sollen.

Typ: Boolesch

Required: No

Standardwert: `False`

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat keine CloudFormation Entsprechung.

`ProvisionedConcurrencyConfig`

Die bereitgestellte Parallelitätskonfiguration des Alias einer Funktion.

> **Note**
>
> `ProvisionedConcurrencyConfig`kann nur angegeben werden, wenn der gesetzt `AutoPublishAlias` ist. Andernfalls tritt ein Fehler auf.

Typ: ProvisionedConcurrencyConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `ProvisionedConcurrencyConfig` Eigenschaft einer `AWS::Lambda::Alias` Ressource übergeben.

`RecursiveLoop`

Der Status der Konfiguration für die Erkennung rekursiver Schleifen Ihrer Funktion.

Wenn dieser Wert auf gesetzt ist `Allow` und Lambda erkennt, dass Ihre Funktion als Teil einer rekursiven Schleife aufgerufen wird, ergreift es keine Aktion.

Wenn dieser Wert auf gesetzt ist `Terminate` und Lambda erkennt, dass Ihre Funktion als Teil einer rekursiven Schleife aufgerufen wird, stoppt es den Aufruf Ihrer Funktion und benachrichtigt Sie.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die Eigenschaft der `RecursiveLoop` Ressource übergeben. `AWS::Lambda::Function`

`ReservedConcurrentExecutions`

Die maximale Anzahl gleichzeitiger Ausführungen, die Sie für die Funktion reservieren möchten.

Weitere Informationen zu dieser Eigenschaft finden Sie unter Lambda Function Scaling im AWS Lambda Entwicklerhandbuch.

Typ: Ganzzahl

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `ReservedConcurrentExecutions` Eigenschaft einer `AWS::Lambda::Function` Ressource übergeben.

`Role`

Der ARN einer IAM-Rolle, die als Ausführungsrolle dieser Funktion verwendet werden soll.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `Role` Eigenschaft einer `AWS::Lambda::Function` Ressource. Dies ist in erforderlich CloudFormation , aber nicht in AWS SAM. Wenn keine Rolle angegeben ist, wird eine für Sie mit der logischen ID von erstellt*<function-logical-id>*`Role`.

`RolePath`

Der Pfad zur IAM-Ausführungsrolle der Funktion.

Verwenden Sie diese Eigenschaft, wenn die Rolle für Sie generiert wird. Verwenden Sie sie nicht, wenn die Rolle mit der `Role` Eigenschaft angegeben ist.

Typ: Zeichenfolge

Required: Conditional

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Path` Eigenschaft einer `AWS::IAM::Role` Ressource übergeben.

`Runtime`

Die ID der Laufzeit der Funktion. Diese Eigenschaft ist nur erforderlich, wenn die `PackageType` Eigenschaft auf gesetzt ist`Zip`.

> **Note**
>
> Wenn Sie den `provided` Bezeichner für diese Eigenschaft angeben, können Sie das `Metadata` Ressourcenattribut verwenden, um anzuweisen AWS SAM , die benutzerdefinierte Laufzeit zu erstellen, die diese Funktion benötigt. Weitere Hinweise zum Erstellen benutzerdefinierter Laufzeiten finden Sie unter. Erstellen von Lambda-Funktionen mit benutzerdefinierten Laufzeiten in AWS SAM

Typ: Zeichenfolge

Required: Conditional

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Runtime` Eigenschaft einer `AWS::Lambda::Function` Ressource übergeben.

`RuntimeManagementConfig`

Konfigurieren Sie Laufzeitverwaltungsoptionen für Ihre Lambda-Funktionen wie Aktualisierungen der Laufzeitumgebung, Rollback-Verhalten und Auswahl einer bestimmten Laufzeitversion. Weitere Informationen finden Sie unter Lambda-Runtime-Updates im Entwicklerhandbuch. AWS Lambda

Typ: RuntimeManagementConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `RuntimeManagementConfig` Eigenschaft einer `AWS::Lambda::Function` Ressource übergeben.

`SnapStart`

Erstellen Sie einen Snapshot jeder neuen Lambda-Funktionsversion. Ein Snapshot ist ein zwischengespeicherter Zustand Ihrer initialisierten Funktion, einschließlich all ihrer Abhängigkeiten. Die Funktion wird nur einmal initialisiert und der zwischengespeicherte Status wird für alle zukünftigen Aufrufe wiederverwendet. Dadurch wird die Anwendungsleistung verbessert, da die Häufigkeit, mit der Ihre Funktion initialisiert werden muss, reduziert wird. Weitere Informationen finden Sie unter Verbessern der Startleistung mit Lambda SnapStart im AWS Lambda Entwicklerhandbuch.

Typ: SnapStart

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `SnapStart` Eigenschaft einer `AWS::Lambda::Function` Ressource übergeben.

`SourceKMSKeyArn`

Stellt einen KMS-Schlüssel-ARN dar, der zum Verschlüsseln des ZIP-Funktionscodes des Kunden verwendet wird.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `SourceKMSKeyArn` Eigenschaft eines `AWS::Lambda::Function` `Code` Datentyps übergeben.

`Tags`

Eine Map (von Zeichenfolge zu Zeichenfolge), die die zu dieser Funktion hinzugefügten Tags angibt. Einzelheiten zu gültigen Schlüsseln und Werten für Tags finden Sie unter Anforderungen an Tag-Schlüssel und -Werte im AWS Lambda Entwicklerhandbuch.

Wenn der Stack erstellt wird, fügt dieser Lambda-Funktion und den Standardrollen, die für diese Funktion generiert werden, AWS SAM automatisch ein `lambda:createdBy:SAM` Tag hinzu.

Typ: Karte

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `Tags` Eigenschaft einer `AWS::Lambda::Function` Ressource. Die `Tags` Eigenschaft in AWS SAM besteht aus Schlüssel-Wert-Paaren (wohingegen CloudFormation diese Eigenschaft aus einer Liste von `Tag` Objekten besteht). Fügt dieser Lambda-Funktion und den Standardrollen, die für diese Funktion generiert werden, außerdem AWS SAM automatisch ein `lambda:createdBy:SAM` Tag hinzu.

`TenancyConfig`

Konfiguration für den Lambda-Mandanten-Isolationsmodus. Stellt sicher, dass Ausführungsumgebungen niemals von verschiedenen Mandanten-IDs gemeinsam genutzt

werden, und ermöglicht so eine Isolierung auf Rechenebene für Anwendungen mit mehreren Mandanten.

Typ: TenancyConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `TenancyConfig` Eigenschaft einer Ressource übergeben. `AWS::Lambda::Function`

`Timeout`

Die maximale Zeit in Sekunden, die die Funktion ausführen kann, bevor sie gestoppt wird.

Typ: Ganzzahl

Required: No

Standard: 3

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Timeout` Eigenschaft einer `AWS::Lambda::Function` Ressource übergeben.

`Tracing`

Die Zeichenfolge, die den X-Ray Ablaufverfolgungsmodus der Funktion angibt.

- `Active`— Aktiviert das X-Ray Tracing für die Funktion.
- `Disabled`— Deaktiviert X-Ray die Funktion.
- `PassThrough`— Aktiviert das X-Ray Tracing für die Funktion. Die Entscheidung über die Probenahme wird an die nachgelagerten Dienste delegiert.

Wenn als `Active` oder angegeben `PassThrough` und die `Role` Eigenschaft nicht festgelegt ist, wird die `arn:aws:iam::aws:policy/AWSXrayWriteOnlyAccess` Richtlinie der Lambda-Ausführungsrolle AWS SAM hinzugefügt, die sie für Sie erstellt.

Weitere Informationen dazu finden Sie X-Ray unter Using AWS Lambda with AWS X-Ray im AWS Lambda Entwicklerhandbuch.

Gültige Werte: [`Active`| `Disabled` |`PassThrough`]

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `TracingConfig` Eigenschaft einer `AWS::Lambda::Function` Ressource.

`VersionDescription`

Gibt das `Description` Feld an, das der neuen Lambda-Versionsressource hinzugefügt wird.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Description` Eigenschaft einer `AWS::Lambda::Version` Ressource übergeben.

`VersionDeletionPolicy`

Gibt die Löschrichtlinie für die Lambda-Versionsressource an, die erstellt wird, wenn sie festgelegt `AutoPublishAlias` ist. Dies steuert, ob die Versionsressource beibehalten oder gelöscht wird, wenn der Stack gelöscht wird.

Gültige Werte: `Delete`, `Retain`, oder `Snapshot`

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent. Sie legt das `DeletionPolicy` Attribut für die generierte `AWS::Lambda::Version` Ressource fest.

`VpcConfig`

Die Konfiguration, die dieser Funktion den Zugriff auf private Ressourcen in Ihrer Virtual Private Cloud (VPC) ermöglicht.

Typ: VpcConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `VpcConfig` Eigenschaft einer `AWS::Lambda::Function` Ressource übergeben.

Rückgabewerte

Punkt

Wenn die logische ID dieser Ressource für die `Ref` intrinsische Funktion bereitgestellt wird, gibt sie den Ressourcennamen der zugrunde liegenden Lambda-Funktion zurück.

Weitere Informationen zur Verwendung der `Ref` Funktion finden Sie `Ref` im AWS CloudFormation Benutzerhandbuch.

Fan:: GetAtt

`Fn::GetAtt` gibt einen Wert für ein angegebenes Attribut dieses Typs zurück. Im Folgenden sehen Sie die verfügbaren Attribute und Beispielrückgabewerte.

Weitere Informationen zur Verwendung `Fn::GetAtt` finden Sie `Fn::GetAtt` im AWS CloudFormation Benutzerhandbuch.

`Arn`

Der ARN der zugrunde liegenden Lambda-Funktion.

Beispiele

Einfache Funktion

Das Folgende ist ein einfaches Beispiel für eine AWS::Serverless: :Funktion Ressource mit Pakettyp `Zip` (Standard) und Funktionscode in einem Amazon S3-Bucket.

YAML

```
Type: AWS::Serverless::Function
Properties:
  Handler: index.handler
  Runtime: python3.9
  CodeUri: s3://bucket-name/key-name
```

Beispiel für Funktionseigenschaften

Das Folgende ist ein Beispiel für einen Pakettyp `Zip` (Standard)`InlineCode`, AWS::Serverless: :Funktion der das `Amazon EFS` Dateisystem `Layers` `TracingPolicies`,,, und eine `Api` Ereignisquelle verwendet.

YAML

```
Type: AWS::Serverless::Function
DependsOn: MyMountTarget        # This is needed if an AWS::EFS::MountTarget resource
 is declared for EFS
Properties:
  Handler: index.handler
  Runtime: python3.9
  InlineCode: |
    def handler(event, context):
      print("Hello, world!")
  ReservedConcurrentExecutions: 30
  Layers:
    - Ref: MyLayer
  Tracing: Active
  Timeout: 120
  FileSystemConfigs:
    - Arn: !Ref MyEfsFileSystem
      LocalMountPath: /mnt/EFS
  Policies:
    - AWSLambdaExecute
    - Version: '2012-10-17       '
      Statement:
        - Effect: Allow
          Action:
            - s3:GetObject
            - s3:GetObjectACL
          Resource: 'arn:aws:s3:::sam-s3-demo-bucket/*'
  Events:
    ApiEvent:
      Type: Api
      Properties:
        Path: /path
        Method: get
```

ImageConfig Beispiel

Das Folgende ist ein Beispiel `ImageConfig` für eine Lambda-Funktion des Pakettyps. `Image`

YAML

```
HelloWorldFunction:
  Type: AWS::Serverless::Function
```

```
  Properties:
    PackageType: Image
    ImageUri: account-id.dkr.ecr.region.amazonaws.com/ecr-repo-name:image-name
    ImageConfig:
      Command:
        - "app.lambda_handler"
      EntryPoint:
        - "entrypoint1"
      WorkingDirectory: "workDir"
```

RuntimeManagementConfig Beispiele

Eine Lambda-Funktion, die so konfiguriert ist, dass sie ihre Laufzeitumgebung entsprechend dem aktuellen Verhalten aktualisiert:

```
TestFunction
  Type: AWS::Serverless::Function
  Properties:
    ...
    Runtime: python3.9
    RuntimeManagementConfig:
      UpdateRuntimeOn: Auto
```

Eine Lambda-Funktion, die so konfiguriert ist, dass sie ihre Laufzeitumgebung aktualisiert, wenn die Funktion aktualisiert wird:

```
TestFunction
  Type: AWS::Serverless::Function
  Properties:
    ...
    Runtime: python3.9
    RuntimeManagementConfig:
      UpdateRuntimeOn: FunctionUpdate
```

Eine Lambda-Funktion, die so konfiguriert ist, dass sie ihre Laufzeitumgebung manuell aktualisiert:

```
TestFunction
  Type: AWS::Serverless::Function
  Properties:
    ...
    Runtime: python3.9
```

```
    RuntimeManagementConfig:
      RuntimeVersionArn: arn:aws:lambda:us-
east-1::runtime:4c459dd0104ee29ec65dcad056c0b3ddbe20d6db76b265ade7eda9a066859b1e
      UpdateRuntimeOn: Manual
```

SnapStart Beispiele

Beispiel für eine Lambda-Funktion, die für zukünftige Versionen SnapStart aktiviert ist:

```
TestFunc
  Type: AWS::Serverless::Function
  Properties:
    ...
    SnapStart:
      ApplyOn: PublishedVersions
```

TenancyConfig Beispiele

Beispiel für eine Lambda-Funktion mit aktiviertem Tenant-Isolationsmodus:

```
TestFunction
  Type: AWS::Serverless::Function
  Properties:
    ...
    TenancyConfig:
      TenantIsolationMode: PER_TENANT
```

Beispiel für ein S3-Dateisystem

Das folgende Beispiel erstellt eine Lambda-Funktion, die ein Amazon S3 Files-Dateisystem mountet. Die Vorlage erstellt einen S3-Bucket, ein von diesem Bucket unterstütztes S3 Files-Dateisystem, ein Mount-Ziel in einem VPC-Subnetz und einen Access Point. Die Funktion mountet den Access Point am S3-Bucket `/mnt/s3files` und kann Dateien lesen und schreiben, die mit dem S3-Bucket synchronisiert werden.

> **Note**
>
> Sie können entweder ein Amazon EFS-Dateisystem oder ein S3 Files-Dateisystem an eine Lambda-Funktion anhängen, aber nicht beide gleichzeitig.

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: Lambda function with S3 Files file system

Resources:
  # VPC and networking
  MyVpc:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: true
      EnableDnsHostnames: true

  MySubnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref MyVpc
      CidrBlock: 10.0.1.0/24
      AvailabilityZone: !Select [0, !GetAZs '']

  MySecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Security group for Lambda and S3 Files
      VpcId: !Ref MyVpc
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 2049
          ToPort: 2049
          CidrIp: 10.0.0.0/16

  # S3 bucket for file storage
  MyS3Bucket:
    Type: AWS::S3::Bucket

  # IAM role for S3 Files to access the bucket
  S3FilesRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
```

```
        - Effect: Allow
          Principal:
            Service: elasticfilesystem.amazonaws.com
          Action: sts:AssumeRole
    Policies:
      - PolicyName: S3FilesBucketAccess
        PolicyDocument:
          Version: '2012-10-17'
          Statement:
            - Effect: Allow
              Action:
                - s3:GetObject
                - s3:PutObject
                - s3:DeleteObject
                - s3:ListBucket
              Resource:
                - !GetAtt MyS3Bucket.Arn
                - !Sub '${MyS3Bucket.Arn}/*'

# S3 Files resources
MyS3FilesFileSystem:
  Type: AWS::S3Files::FileSystem
  Properties:
    Bucket: !GetAtt MyS3Bucket.Arn
    RoleArn: !GetAtt S3FilesRole.Arn

MyS3FilesMountTarget:
  Type: AWS::S3Files::MountTarget
  Properties:
    FileSystemId: !Ref MyS3FilesFileSystem
    SubnetId: !Ref MySubnet
    SecurityGroups:
      - !Ref MySecurityGroup

MyS3FilesAccessPoint:
  Type: AWS::S3Files::AccessPoint
  Properties:
    FileSystemId: !Ref MyS3FilesFileSystem
    PosixUser:
      Uid: '1000'
      Gid: '1000'
    RootDirectory:
      Path: /lambda
      CreationInfo:
```

```
        OwnerUid: '1000'
        OwnerGid: '1000'
        Permissions: '750'

# Lambda function with S3 Files mount
MyFunction:
  Type: AWS::Serverless::Function
  DependsOn: MyS3FilesMountTarget
  Properties:
    Handler: index.handler
    Runtime: python3.12
    Timeout: 120
    VpcConfig:
      SecurityGroupIds:
        - !Ref MySecurityGroup
      SubnetIds:
        - !Ref MySubnet
    FileSystemConfigs:
      - Arn: !GetAtt MyS3FilesAccessPoint.AccessPointArn
        LocalMountPath: /mnt/s3files
    Policies:
      - Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Action:
              - s3files:ClientMount
              - s3files:ClientWrite
            Resource: !GetAtt MyS3FilesAccessPoint.AccessPointArn
    InlineCode: |
      import os

      def handler(event, context):
          # Write a file to the S3 Files mount
          with open('/mnt/s3files/hello.txt', 'w') as f:
              f.write('Hello from Lambda!')

          # List files at the mount path
          files = os.listdir('/mnt/s3files')
          return {'files': files}
```

DeadLetterQueue

Gibt eine SQS-Warteschlange oder ein SNS-Thema an, an das AWS Lambda (Lambda) Ereignisse sendet, wenn sie nicht verarbeitet werden können. Weitere Informationen zur Funktionalität von Warteschlangen für unzustellbare Briefe finden Sie unter Warteschlangen für unzustellbare Briefe im Entwicklerhandbuch.AWS Lambda

SAM fügt Ihrer Lambda-Funktionsausführungsrolle automatisch die entsprechende Berechtigung hinzu, um dem Lambda-Dienst Zugriff auf die Ressource zu gewähren. sqs: SendMessage wird für SQS-Warteschlangen und sns:Publish für SNS-Themen hinzugefügt.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage () zu deklarieren. AWS Serverless Application Model AWS SAM

YAML

```
TargetArn: String
Type: String
```

Eigenschaften

`TargetArn`

Der Amazon-Ressourcenname (ARN) einer Amazon SQS-Warteschlange oder eines Amazon SNS-Themas.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `TargetArn` Eigenschaft des `AWS::Lambda::Function` `DeadLetterConfig` Datentyps übergeben.

`Type`

Der Typ der Warteschlange für unzustellbare Nachrichten.

Zulässige Werte: `SNS`, `SQS`

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

DeadLetterQueue

Beispiel für eine Dead-Letter-Warteschlange für ein SNS-Thema.

YAML

```
DeadLetterQueue:
  Type: SNS
  TargetArn: arn:aws:sns:us-east-2:123456789012:my-topic
```

DeploymentPreference

Gibt die Konfigurationen an, um schrittweise Lambda-Bereitstellungen zu ermöglichen. Weitere Informationen zur Konfiguration schrittweiser Lambda-Bereitstellungen finden Sie unter. Schrittweise Bereitstellung serverloser Anwendungen mit AWS SAM

> **Note**
>
> Sie müssen ein `AutoPublishAlias` in Ihrem angebenAWS::Serverless: :Funktion, um ein `DeploymentPreference` Objekt verwenden zu können, andernfalls tritt ein Fehler auf.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
  Alarms: List
  Enabled: Boolean
  Hooks: Hooks
  PassthroughCondition: Boolean
```

```
  Role: String
  TriggerConfigurations: List
  Type: String
```

Eigenschaften

`Alarms`

Eine Liste von CloudWatch Alarmen, die durch alle durch die Bereitstellung ausgelösten Fehler ausgelöst werden sollen.

Diese Eigenschaft akzeptiert die `Fn::If` intrinsische Funktion. Im Abschnitt Beispiele am Ende dieses Themas finden Sie eine Beispielvorlage, die verwendet. `Fn::If`

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Enabled`

Ob diese Bereitstellungspräferenz aktiviert ist.

Typ: Boolesch

Required: No

Standard: Wahr

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Hooks`

Validierung Lambda-Funktionen, die vor und nach der Verkehrsverlagerung ausgeführt werden.

Typ: Hooks

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig AWS SAM und hat kein CloudFormation Äquivalent.

`PassthroughCondition`

Bei True und wenn diese Bereitstellungseinstellung aktiviert ist, wird die Bedingung der Funktion an die generierte CodeDeploy Ressource weitergegeben. Im Allgemeinen sollten Sie dies auf True setzen. Andernfalls würde die CodeDeploy Ressource auch dann erstellt, wenn die Bedingung der Funktion auf False aufgelöst wird.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Role`

Ein ARN mit einer IAM-Rolle, der für die Verkehrsverlagerung verwendet CodeDeploy wird. Eine IAM-Rolle wird nicht erstellt, wenn diese bereitgestellt wird.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`TriggerConfigurations`

Eine Liste der Triggerkonfigurationen, die Sie der Bereitstellungsgruppe zuordnen möchten. Wird verwendet, um ein SNS-Thema über Lebenszyklusereignisse zu informieren.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `TriggerConfigurations` Eigenschaft einer `AWS::CodeDeploy::DeploymentGroup` Ressource übergeben.

`Type`

Derzeit gibt es zwei Kategorien von Bereitstellungstypen: Linear und Canary. Weitere Informationen zu den verfügbaren Bereitstellungstypen finden Sie unterSchrittweise Bereitstellung serverloser Anwendungen mit AWS SAM.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

DeploymentPreference mit Haken vor und nach dem Verkehr.

Beispiel für eine Bereitstellungspräferenz, die Hooks vor und nach dem Traffic enthält.

YAML

```
DeploymentPreference:
  Enabled: true
  Type: Canary10Percent10Minutes
  Alarms:
    - !Ref: AliasErrorMetricGreaterThanZeroAlarm
    - !Ref: LatestVersionErrorMetricGreaterThanZeroAlarm
  Hooks:
    PreTraffic:
      !Ref: PreTrafficLambdaFunction
    PostTraffic:
      !Ref: PostTrafficLambdaFunction
```

DeploymentPreference mit der intrinsischen Funktion Fn: :If

Beispiel für eine Bereitstellungspräferenz, die `Fn::If` für die Konfiguration von Alarmen verwendet wird. In diesem Beispiel `Alarm1` wird konfiguriert, ob dies der Fall `MyCondition` ist`true`, `Alarm2` und `Alarm5` wird konfiguriert, wenn dies der Fall `MyCondition` ist`false`.

YAML

```
DeploymentPreference:
  Enabled: true
  Type: Canary10Percent10Minutes
  Alarms:
    Fn::If:
      - MyCondition
      - - Alarm1
      - - Alarm2
        - Alarm5
```

Hooks

Validierung Lambda-Funktionen, die vor und nach der Verkehrsverlagerung ausgeführt werden.

> **Note**
>
> Die Lambda-Funktionen, auf die in dieser Eigenschaft verwiesen wird, konfigurieren das `CodeDeployLambdaAliasUpdate` Objekt der resultierenden AWS::Lambda::AliasRessource. Weitere Informationen finden Sie unter CodeDeployLambdaAliasUpdate Richtlinie im AWS CloudFormation Benutzerhandbuch.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
PostTraffic: String
PreTraffic: String
```

Eigenschaften

`PostTraffic`

Lambda-Funktion, die nach einer Verkehrsverlagerung ausgeführt wird.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`PreTraffic`

Lambda-Funktion, die vor der Verkehrsverlagerung ausgeführt wird.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

Hook

Beispiele für Hook-Funktionen

YAML

```
Hooks:
  PreTraffic:
    Ref: PreTrafficLambdaFunction
  PostTraffic:
    Ref: PostTrafficLambdaFunction
```

DurableConfig

Konfiguriert dauerhafte Ausführungseinstellungen für AWS Lambda Funktionen. Dauerhafte Funktionen können bis zu einem Jahr lang ausgeführt werden und überprüfen den Fortschritt automatisch, sodass Workflows mit langer Laufzeit und fehlertolerante Anwendungen ermöglicht werden. Weitere Informationen zu dauerhaften Funktionen finden Sie unter Dauerhafte Lambda-Funktionen im AWS Lambda Entwicklerhandbuch.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer AWS Serverless Application Model (AWS SAM) -Vorlage zu deklarieren.

YAML

```
  ExecutionTimeout: Integer
  RetentionPeriodInDays: Integer
```

Eigenschaften

`ExecutionTimeout`

Die Zeitspanne (in Sekunden), die Lambda einer dauerhaften Funktion ermöglicht, bevor sie gestoppt wird. Das Maximum ist ein Jahr mit 366 Tagen oder 31.622.400 Sekunden.

Typ: Ganzzahl

Erforderlich: Ja

Minimum: 1

Maximum: 31622400

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `ExecutionTimeout` Eigenschaft des `AWS::Lambda::Function` `DurableConfig` Datentyps übergeben.

`RetentionPeriodInDays`

Die Anzahl der Tage, an denen Lambda nach Abschluss einer dauerhaften Ausführung seinen Verlauf beibehält, von einem bis 90 Tagen. Der Standardwert ist 14 Tage.

Typ: Ganzzahl

Required: No

Standard: 14

Minimum: 1

Maximal: 90

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `RetentionPeriodInDays` Eigenschaft des `AWS::Lambda::Function` `DurableConfig` Datentyps übergeben.

Beispiele

DurableConfig

Beispiel für eine dauerhafte Konfiguration für eine Funktion mit einem Ausführungstimeout von 1 Stunde und einer Aufbewahrungsfrist von 7 Tagen.

YAML

```
DurableConfig:
  ExecutionTimeout: 3600
  RetentionPeriodInDays: 7
```

EventInvokeConfiguration

Konfigurationsoptionen für asynchrone Lambda-Alias- oder Versionsaufrufe.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
DestinationConfig: EventInvokeDestinationConfiguration
MaximumEventAgeInSeconds: Integer
MaximumRetryAttempts: Integer
```

Eigenschaften

`DestinationConfig`

Ein Konfigurationsobjekt, das das Ziel eines Ereignisses angibt, nachdem Lambda es verarbeitet hat.

Typ: EventInvokeDestinationConfiguration

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `DestinationConfig` Eigenschaft einer `AWS::Lambda::EventInvokeConfig` Ressource. SAM benötigt einen zusätzlichen Parameter, „Type“, der in nicht existiert CloudFormation.

`MaximumEventAgeInSeconds`

Das maximale Alter einer Anforderung, die Lambda an eine Funktion zur Verarbeitung sendet.

Typ: Ganzzahl

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `MaximumEventAgeInSeconds` Eigenschaft einer `AWS::Lambda::EventInvokeConfig` Ressource übergeben.

`MaximumRetryAttempts`

Die maximale Anzahl von Wiederholungen, bis die Funktion einen Fehler zurückgibt.

Typ: Ganzzahl

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `MaximumRetryAttempts` Eigenschaft einer `AWS::Lambda::EventInvokeConfig` Ressource übergeben.

Beispiele

MaximumEventAgeInSeconds

MaximumEventAgeInSeconds Beispiel

YAML

```
EventInvokeConfig:
  MaximumEventAgeInSeconds: 60
  MaximumRetryAttempts: 2
  DestinationConfig:
    OnSuccess:
      Type: SQS
      Destination: arn:aws:sqs:us-west-2:012345678901:my-queue
    OnFailure:
      Type: Lambda
      Destination: !GetAtt DestinationLambda.Arn
```

EventInvokeDestinationConfiguration

Ein Konfigurationsobjekt, das das Ziel eines Ereignisses angibt, nachdem Lambda es verarbeitet hat.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
  OnFailure: OnFailure
```

```
  OnSuccess: OnSuccess
```

Eigenschaften

`OnFailure`

Ein Ziel für Ereignisse, bei denen die Verarbeitung fehlgeschlagen ist.

Typ: OnFailure

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `OnFailure` Eigenschaft einer `AWS::Lambda::EventInvokeConfig` Ressource. Erfordert eine zusätzliche Eigenschaft`Type`, die nur für SAM bestimmt ist.

`OnSuccess`

Ein Ziel für Ereignisse, die erfolgreich verarbeitet wurden.

Typ: OnSuccess

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `OnSuccess` Eigenschaft einer `AWS::Lambda::EventInvokeConfig` Ressource. Erfordert eine zusätzliche Eigenschaft`Type`, die nur für SAM bestimmt ist.

Beispiele

OnSuccess

OnSuccess Beispiel

YAML

```
EventInvokeConfig:
  DestinationConfig:
    OnSuccess:
      Type: SQS
      Destination: arn:aws:sqs:us-west-2:012345678901:my-queue
    OnFailure:
      Type: Lambda
```

```
      Destination: !GetAtt DestinationLambda.Arn
```

OnFailure

Ein Ziel für Ereignisse, bei denen die Verarbeitung fehlgeschlagen ist.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
  Destination: String
  Type: String
```

Eigenschaften

`Destination`

Der Amazon-Ressourcenname (ARN) der Zielressource.

Typ: Zeichenfolge

Required: Conditional

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `OnFailure` Eigenschaft einer `AWS::Lambda::EventInvokeConfig` Ressource. SAM fügt der automatisch generierten IAM-Rolle, die mit dieser Funktion verknüpft ist, alle erforderlichen Berechtigungen hinzu, um auf die in dieser Eigenschaft referenzierte Ressource zuzugreifen.

Zusätzliche Hinweise: Wenn der Typ Lambda/ istEventBridge, ist Destination erforderlich.

`Type`

Typ der Ressource, auf die im Ziel verwiesen wird. Unterstützte Typen sind `SQS`SNS,`S3`,`Lambda`, und`EventBridge`.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Zusätzliche Hinweise: Wenn der Typ ist SQS/SNS und die `Destination` Eigenschaft leer gelassen wird, wird die SQS/SNS Ressource auto von SAM generiert. Um auf die Ressource zu verweisen, verwenden Sie *<function-logical-id>*`.DestinationQueue` für SQS oder *<function-logical-id>*`.DestinationTopic` für SNS. Wenn der Typ Lambda/ istEventBridge, `Destination` ist dies erforderlich.

Beispiele

EventInvoke Konfigurationsbeispiel mit SQS- und Lambda-Zielen

In diesem Beispiel wurde kein Ziel für die OnSuccess SQS-Konfiguration angegeben, daher erstellt SAM implizit eine SQS-Warteschlange und fügt alle erforderlichen Berechtigungen hinzu. Auch für dieses Beispiel wird ein Ziel für eine in der Vorlagendatei deklarierte Lambda-Ressource in der OnFailure Konfiguration angegeben, sodass SAM dieser Lambda-Funktion die erforderlichen Berechtigungen zum Aufrufen der Lambda-Zielfunktion hinzufügt.

YAML

```
EventInvokeConfig:
  DestinationConfig:
    OnSuccess:
      Type: SQS
    OnFailure:
      Type: Lambda
      Destination: !GetAtt DestinationLambda.Arn  # Arn of a Lambda function declared
 in the template file.
```

EventInvoke Konfigurationsbeispiel mit SNS-Ziel

In diesem Beispiel wird ein Ziel für ein SNS-Thema angegeben, das in der Vorlagendatei für die OnSuccess Konfiguration deklariert ist.

YAML

```
EventInvokeConfig:
  DestinationConfig:
    OnSuccess:
      Type: SNS
      Destination:
        Ref: DestinationSNS       # Arn of an SNS topic declared in the tempate file
```

OnSuccess

Ein Ziel für Ereignisse, die erfolgreich verarbeitet wurden.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
Destination: String
Type: String
```

Eigenschaften

`Destination`

Der Amazon-Ressourcenname (ARN) der Zielressource.

Typ: Zeichenfolge

Required: Conditional

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `OnSuccess` Eigenschaft einer `AWS::Lambda::EventInvokeConfig` Ressource. SAM fügt der automatisch generierten IAM-Rolle, die mit dieser Funktion verknüpft ist, alle erforderlichen Berechtigungen hinzu, um auf die in dieser Eigenschaft referenzierte Ressource zuzugreifen.

Zusätzliche Hinweise: Wenn der Typ Lambda/ istEventBridge, ist Destination erforderlich.

`Type`

Typ der Ressource, auf die im Ziel verwiesen wird. Unterstützte Typen sind `SQSSNS`,`S3`,`Lambda`, und`EventBridge`.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Zusätzliche Hinweise: Wenn der Typ ist SQS/SNS und die `Destination` Eigenschaft leer gelassen wird, wird die SQS/SNS Ressource auto von SAM generiert. Um auf die Ressource zu verweisen, verwenden Sie *<function-logical-id>*`.DestinationQueue` für SQS oder *<function-logical-id>*`.DestinationTopic` für SNS. Wenn der Typ Lambda/ istEventBridge, `Destination` ist dies erforderlich.

Beispiele

EventInvoke Konfigurationsbeispiel mit SQS- und Lambda-Destinationen

In diesem Beispiel wurde kein Ziel für die OnSuccess SQS-Konfiguration angegeben, daher erstellt SAM implizit eine SQS-Warteschlange und fügt alle erforderlichen Berechtigungen hinzu. Auch für dieses Beispiel wird ein Ziel für eine in der Vorlagendatei deklarierte Lambda-Ressource in der OnFailure Konfiguration angegeben, sodass SAM dieser Lambda-Funktion die erforderlichen Berechtigungen zum Aufrufen der Lambda-Zielfunktion hinzufügt.

YAML

```
EventInvokeConfig:
  DestinationConfig:
    OnSuccess:
      Type: SQS
    OnFailure:
      Type: Lambda
      Destination: !GetAtt DestinationLambda.Arn  # Arn of a Lambda function declared
 in the template file.
```

EventInvoke Konfigurationsbeispiel mit SNS-Ziel

In diesem Beispiel wird ein Ziel für ein SNS-Thema angegeben, das in der Vorlagendatei für die OnSuccess Konfiguration deklariert ist.

YAML

```
EventInvokeConfig:
  DestinationConfig:
    OnSuccess:
      Type: SNS
      Destination:
        Ref: DestinationSNS       # Arn of an SNS topic declared in the tempate file
```

EventSource

Das Objekt, das die Quelle der Ereignisse beschreibt, die die Funktion auslösen. Jedes Ereignis besteht aus einem Typ und einer Reihe von Eigenschaften, die von diesem Typ abhängen. Weitere Informationen zu den Eigenschaften der einzelnen Ereignisquellen finden Sie in dem entsprechenden Thema.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer AWS Serverless Application Model (AWS SAM) -Vorlage zu deklarieren.

YAML

```
  Properties: AlexaSkill | Api | CloudWatchEvent | CloudWatchLogs | Cognito
 | DocumentDB | DynamoDB | EventBridgeRule | HttpApi | IoTRule | Kinesis | MQ | MSK
 | S3 | Schedule | ScheduleV2 | SelfManagedKafka | SNS | SQS
  Type: String
```

Eigenschaften

`Properties`

Objekt, das die Eigenschaften dieser Event-Mapping beschreibt. Der Satz von Eigenschaften muss dem definierten Typ entsprechen.

Typ : AlexaSkill| Api | | CloudWatchEvent| Cognito CloudWatchLogs| DocumentDB | DynamoDB | | | Io EventBridgeRule| Kinesis HttpApi| MQ TRule | MSK | S3 | Zeitplan | ScheduleV2 | | SNS | SQS SelfManagedKafka

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für und hat kein Äquivalent. AWS SAM CloudFormation

`Type`

Der Ereignistyp.

Gültige Werte: `AlexaSkill` `ApiCloudWatchEvent`,`CloudWatchLogs`,`Cognito`,`DocumentDB`,`DynamoDB`,`EventBridgeRule`,`Htt` `MQMSK`,`S3`,`Schedule`,`ScheduleV2`,`SelfManagedKafka`,`SNS`, `SQS`

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

APIEvent

Beispiel für die Verwendung eines API-Ereignisses

YAML

```
ApiEvent:
  Type: Api
  Properties:
    Method: get
    Path: /group/{user}
    RestApiId:
      Ref: MyApi
```

AlexaSkill

Das Objekt, das einen `AlexaSkill` Ereignisquellentyp beschreibt.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
  SkillId: String
```

Eigenschaften

`SkillId`

Die Alexa Skill ID für Ihren Alexa Skill. Weitere Informationen zur Skill-ID finden Sie unter Konfigurieren des Triggers für eine Lambda-Funktion in der Alexa Skills Kit-Dokumentation.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

AlexaSkillTrigger

Beispiel für ein Alexa Skill-Event

YAML

```
AlexaSkillEvent:
  Type: AlexaSkill
```

Api

Das Objekt, das einen `Api` Ereignisquellentyp beschreibt. Wenn eine AWS::Serverless::Api Ressource definiert ist, müssen der Pfad und die Methodenwerte einer Operation in der OpenAPI-Definition der API entsprechen.

Wenn nein definiert AWS::Serverless::Api ist, stellen die Eingabe und Ausgabe der Funktion die HTTP-Anfrage und die HTTP-Antwort dar.

Mithilfe der JavaScript API können beispielsweise der statusCode und der Hauptteil der Antwort gesteuert werden, indem ein Objekt mit den Schlüsseln StatusCode und body zurückgegeben wird.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
  Auth: ApiFunctionAuth
  Method: String
  Path: String
  RequestModel: RequestModel
  RequestParameters: List of [ String | RequestParameter ]
```

```
  RestApiId: String
  ResponseTransferMode: String
  TimeoutInMillis: Integer
```

Eigenschaften

`Auth`

Authentifizierungskonfiguration für diese spezielle API+Path+Methode.

Nützlich, um die Authentifizierungskonfiguration der `DefaultAuthorizer` API-Einstellung für einen einzelnen Pfad zu überschreiben, wenn kein Pfad angegeben `DefaultAuthorizer` ist, oder um die Standardeinstellung zu überschreiben. `ApiKeyRequired`

Typ: ApiFunctionAuth

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein Äquivalent. CloudFormation

`Method`

HTTP-Methode, für die diese Funktion aufgerufen wird. Zu den Optionen gehören `DELETEGET`,`HEAD`,`OPTIONS`,`PATCH`, `POSTPUT`, und`ANY`. Einzelheiten finden Sie unter HTTP-Methode einrichten im API Gateway Developer Guide.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Path`

URI-Pfad, für den diese Funktion aufgerufen wird. Muss mit `/` beginnen.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig AWS SAM und hat kein CloudFormation Äquivalent.

`RequestModel`

Fordern Sie das Modell an, das für diese spezielle API+Path+Methode verwendet werden soll. Dies sollte auf den Namen eines Modells verweisen, das im `Models` Abschnitt einer Ressource angegeben ist. AWS::Serverless::Api

Typ: RequestModel

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`RequestParameters`

Fordern Sie die Parameterkonfiguration für diese spezielle API+Path+Methode an. Alle Parameternamen müssen mit, oder beginnen `method.request` und darauf beschränkt `method.request.header` sein. `method.request.querystring` `method.request.path`

Eine Liste kann sowohl Zeichenketten als auch RequestParameterObjekte mit Parameternamen enthalten. Bei Zeichenketten sind die `Caching` Eigenschaften `Required` und standardmäßig auf eingestellt`false`.

Typ: Liste von [Zeichenfolge | RequestParameter]

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`RestApiId`

Bezeichner einer RestApi Ressource, die eine Operation mit dem angegebenen Pfad und der angegebenen Methode enthalten muss. In der Regel ist dies so eingestellt, dass es auf eine in dieser Vorlage definierte AWS::Serverless::Api Ressource verweist.

Wenn Sie diese Eigenschaft nicht definieren, AWS SAM wird mithilfe eines generierten `OpenApi` Dokuments eine AWS::Serverless::Api Standardressource erstellt. Diese Ressource enthält eine Vereinigung aller Pfade und Methoden, die durch `Api` Ereignisse in derselben Vorlage definiert wurden, ohne dass a angegeben ist`RestApiId`.

Dies kann nicht auf eine AWS::Serverless::Api Ressource verweisen, die in einer anderen Vorlage definiert ist.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`ResponseTransferMode`

Der Antwortübertragungsmodus für die Lambda-Funktionsintegration. Wird auf gesetzt`RESPONSE_STREAM`, um Lambda-Antwort-Streaming über API Gateway zu aktivieren, sodass die Funktion Antworten zurück an Clients streamen kann. Wenn auf gesetzt`RESPONSE_STREAM`, verwendet API Gateway die InvokeWithResponseStreaming Lambda-API.

Typ: Zeichenfolge

Required: No

Gültige Werte: `BUFFERED` | `RESPONSE_STREAM`

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `ResponseTransferMode`Eigenschaft eines `AWS::ApiGateway::Method Integration` übergeben.

`TimeoutInMillis`

Benutzerdefinierte Zeitüberschreitung zwischen 50 und 29.000 Millisekunden.

> **Note**
>
> Wenn Sie diese Eigenschaft angeben, AWS SAM wird Ihre OpenAPI-Definition geändert. Die OpenAPI-Definition muss mithilfe der `DefinitionBody` Eigenschaft inline angegeben werden.

Typ: Ganzzahl

Required: No

Standard: 29.000 Millisekunden oder 29 Sekunden

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein Äquivalent. CloudFormation

Beispiele

Einfaches Beispiel

YAML

```
Events:
  ApiEvent:
    Type: Api
    Properties:
      Path: /path
      Method: get
      RequestParameters:
        - method.request.header.Authorization
        - method.request.querystring.keyword:
            Required: true
            Caching: false
```

ApiFunctionAuth

Konfiguriert die Autorisierung auf Ereignisebene für eine bestimmte API, einen bestimmten Pfad und eine Methode.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
  ApiKeyRequired: Boolean
  AuthorizationScopes: List
  Authorizer: String
  InvokeRole: String
  OverrideApiAuth: Boolean
  ResourcePolicy: ResourcePolicyStatement
```

Eigenschaften

`ApiKeyRequired`

Erfordert einen API-Schlüssel für diese API, diesen Pfad und diese Methode.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`AuthorizationScopes`

Die Autorisierungsbereiche, die für diese API, diesen Pfad und diese Methode gelten sollen.

Die von Ihnen angegebenen Bereiche haben Vorrang vor allen Bereichen, die von der `DefaultAuthorizer` Eigenschaft angewendet werden, sofern Sie sie angegeben haben.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Authorizer`

Die `Authorizer` für eine bestimmte Funktion.

Wenn Sie für Ihre `AWS::Serverless::Api` Ressource einen globalen Autorisierer angegeben haben, können Sie den Autorisierer überschreiben, indem Sie auf setzen`Authorizer`. `NONE` Ein Beispiel finden Sie unter Überschreiben Sie einen globalen Autorisierer für Ihre Amazon API Gateway Gateway-REST-API.

> **Note**
>
> Wenn Sie die `DefinitionBody` Eigenschaft einer `AWS::Serverless::Api` Ressource verwenden, um Ihre API zu beschreiben, müssen Sie `OverrideApiAuth` with verwenden, um Ihren globalen `Authorizer` Autorisierer zu überschreiben. Weitere Informationen finden Sie unter `OverrideApiAuth`.

Gültige Werte: `AWS_IAM`, `NONE`, oder die logische ID für jeden in Ihrer AWS SAM Vorlage definierten Authorizer.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`InvokeRole`

Gibt an, welche für `InvokeRole` die `AWS_IAM` Autorisierung verwendet werden soll.

Typ: Zeichenfolge

Required: No

Standardwert: `CALLER_CREDENTIALS`

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Zusätzliche Hinweise: `CALLER_CREDENTIALS` Mappings to`arn:aws:iam::`*`:<user>/`*, das die Anmeldeinformationen des Anrufers verwendet, um den Endpunkt aufzurufen.

`OverrideApiAuth`

Geben Sie `true` an, ob die globale Autorisierungskonfiguration Ihrer Ressource überschrieben werden soll. `AWS::Serverless::Api` Diese Eigenschaft ist nur erforderlich, wenn Sie einen globalen Autorisierer angeben und die `DefinitionBody` Eigenschaft einer `AWS::Serverless::Api` Ressource verwenden, um Ihre API zu beschreiben.

> **Note**
>
> Wenn Sie `OverrideApiAuth` als angeben`true`, AWS SAM wird Ihr globaler Autorisierer mit allen für `ApiKeyRequiredAuthorizer`, oder angegebenen Werten überschrieben. `ResourcePolicy` Daher muss bei der Verwendung `OverrideApiAuth` mindestens eine dieser Eigenschaften ebenfalls angegeben werden. Ein Beispiel finden Sie unter Überschreiben Sie einen globalen Autorisierer, wenn DefinitionBody für AWS::Serverless::Api angegeben ist.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`ResourcePolicy`

Konfigurieren Sie die Ressourcenrichtlinie für diesen Pfad auf einer API.

Typ: ResourcePolicyStatement

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

Funktions-Auth

Das folgende Beispiel spezifiziert die Autorisierung auf Funktionsebene.

YAML

```
Auth:
  ApiKeyRequired: true
  Authorizer: NONE
```

Überschreiben Sie einen globalen Autorisierer für Ihre Amazon API Gateway Gateway-REST-API

Sie können einen globalen Autorisierer für Ihre `AWS::Serverless::Api` Ressource angeben. Im Folgenden finden Sie ein Beispiel für die Konfiguration eines globalen Standardautorisierers:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyApiWithLambdaRequestAuth:
    Type: AWS::Serverless::Api
    Properties:
      ...
      Auth:
        Authorizers:
          MyLambdaRequestAuth:
            FunctionArn: !GetAtt MyAuthFn.Arn
```

```
        DefaultAuthorizer: MyLambdaRequestAuth
```

Um den Standardautorisierer für Ihre AWS Lambda Funktion zu überschreiben, können Sie als angeben. `Authorizer` `NONE` Im Folgenden wird ein Beispiel gezeigt:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  ...
  MyFn:
    Type: AWS::Serverless::Function
    Properties:
      ...
      Events:
        LambdaRequest:
          Type: Api
          Properties:
            RestApiId: !Ref MyApiWithLambdaRequestAuth
            Method: GET
            Auth:
              Authorizer: NONE
```

Überschreiben Sie einen globalen Autorisierer, wenn DefinitionBody für AWS::Serverless::Api angegeben ist

Wenn Sie die `DefinitionBody` Eigenschaft zur Beschreibung Ihrer `AWS::Serverless::Api` Ressource verwenden, funktioniert die vorherige Override-Methode nicht. Im Folgenden finden Sie ein Beispiel für die Verwendung der `DefinitionBody` Eigenschaft für eine `AWS::Serverless::Api` Ressource:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyApiWithLambdaRequestAuth:
    Type: AWS::Serverless::Api
    Properties:
      ...
      DefinitionBody:
        swagger: 2.0
        ...
```

```
        paths:
          /lambda-request:
            ...
      Auth:
        Authorizers:
          MyLambdaRequestAuth:
            FunctionArn: !GetAtt MyAuthFn.Arn
        DefaultAuthorizer: MyLambdaRequestAuth
```

Verwenden Sie die `OverrideApiAuth` Eigenschaft, um den globalen Autorisierer zu überschreiben. Im Folgenden finden Sie ein Beispiel, das verwendet wird, `OverrideApiAuth` um den globalen Autorisierer mit dem für angegebenen Wert zu überschreiben: `Authorizer`

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyApiWithLambdaRequestAuth:
    Type: AWS::Serverless::Api
    Properties:
      ...
      DefinitionBody:
        swagger: 2-0
        ...
        paths:
          /lambda-request:
            ...
      Auth:
        Authorizers:
          MyLambdaRequestAuth:
            FunctionArn: !GetAtt MyAuthFn.Arn
        DefaultAuthorizer: MyLambdaRequestAuth

    MyAuthFn:
      Type: AWS::Serverless::Function
      ...

    MyFn:
      Type: AWS::Serverless::Function
        Properties:
          ...
          Events:
            LambdaRequest:
```

```
            Type: Api
            Properties:
              RestApiId: !Ref MyApiWithLambdaRequestAuth
              Method: GET
              Auth:
                Authorizer: NONE
                OverrideApiAuth: true
              Path: /lambda-token
```

ResourcePolicyStatement

Konfiguriert eine Ressourcenrichtlinie für alle Methoden und Pfade einer API. Weitere Informationen zu Ressourcenrichtlinien finden Sie unter Steuern des Zugriffs auf eine API mit API-Gateway-Ressourcenrichtlinien im API Gateway Developer Guide.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
AwsAccountBlacklist: List
AwsAccountWhitelist: List
CustomStatements: List
IntrinsicVpcBlacklist: List
IntrinsicVpcWhitelist: List
IntrinsicVpceBlacklist: List
IntrinsicVpceWhitelist: List
IpRangeBlacklist: List
IpRangeWhitelist: List
SourceVpcBlacklist: List
SourceVpcWhitelist: List
```

Eigenschaften

`AwsAccountBlacklist`

Die zu blockierenden AWS Konten.

Typ: Liste von Zeichenfolgen

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig AWS SAM und hat kein CloudFormation Äquivalent.

`AwsAccountWhitelist`

Die AWS Konten, die zugelassen werden sollen. Ein Beispiel für die Verwendung dieser Eigenschaft finden Sie im Abschnitt Beispiele unten auf dieser Seite.

Typ: Liste von Zeichenfolgen

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`CustomStatements`

Eine Liste von benutzerdefinierten Ressourcenrichtlinien-Anweisungen, die auf diese API angewendet werden sollen. Ein Beispiel für die Verwendung dieser Eigenschaft finden Sie im Abschnitt Beispiele unten auf dieser Seite.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`IntrinsicVpcBlacklist`

Die Liste der zu blockierenden virtuellen privaten Clouds (VPCs), wobei jede VPC als Referenz angegeben ist, z. B. als dynamische Referenz oder als `Ref` intrinsische Funktion. Ein Beispiel für die Verwendung dieser Eigenschaft finden Sie im Abschnitt Beispiele unten auf dieser Seite.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`IntrinsicVpcWhitelist`

Die Liste der VPCs Zulässigen, wobei jede VPC als Referenz angegeben wird, z. B. als dynamische Referenz oder als `Ref` intrinsische Funktion.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig AWS SAM und hat kein Äquivalent. CloudFormation

`IntrinsicVpceBlacklist`

Die Liste der zu blockierenden VPC-Endpunkte, wobei jeder VPC-Endpunkt als Referenz angegeben ist, z. B. als dynamische Referenz oder als intrinsische Funktion. `Ref`

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein Äquivalent. CloudFormation

`IntrinsicVpceWhitelist`

Die Liste der VPC-Endpunkte, die zugelassen werden sollen, wobei jeder VPC-Endpunkt als Referenz angegeben ist, z. B. als dynamische Referenz oder als systemeigene Funktion. `Ref` Ein Beispiel für die Verwendung dieser Eigenschaft finden Sie im Abschnitt Beispiele unten auf dieser Seite.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`IpRangeBlacklist`

Die IP-Adressen oder Adressbereiche, die blockiert werden sollen. Ein Beispiel für die Verwendung dieser Eigenschaft finden Sie im Abschnitt Beispiele unten auf dieser Seite.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`IpRangeWhitelist`

Die IP-Adressen oder Adressbereiche, die zugelassen werden sollen.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`SourceVpcBlacklist`

Die Quell-VPC oder die VPC-Endpoints, die blockiert werden sollen. Quell-VPC-Namen müssen mit beginnen `"vpc-"` und Quell-VPC-Endpunktnamen müssen mit beginnen. `"vpce-"` Ein Beispiel für die Verwendung dieser Eigenschaft finden Sie im Abschnitt Beispiele unten auf dieser Seite.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`SourceVpcWhitelist`

Die Quell-VPC oder VPC-Endpoints, die zugelassen werden sollen. Quell-VPC-Namen müssen mit beginnen `"vpc-"` und Quell-VPC-Endpunktnamen müssen mit beginnen. `"vpce-"`

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

Beispiel für eine Ressourcenrichtlinie

Das folgende Beispiel blockiert zwei IP-Adressen und eine Quell-VPC und lässt ein AWS Konto zu.

YAML

```
Auth:
  ResourcePolicy:
    CustomStatements: [{
                         "Effect": "Allow",
                         "Principal": "*",
                         "Action": "execute-api:Invoke",
                         "Resource": "execute-api:/Prod/GET/pets",
                         "Condition": {
                           "IpAddress": {
                             "aws:SourceIp": "1.2.3.4"
                           }
                         }
                       }]
    IpRangeBlacklist:
      - "10.20.30.40"
      - "1.2.3.4"
    SourceVpcBlacklist:
      - "vpce-1a2b3c4d"
    AwsAccountWhitelist:
      - "111122223333"
    IntrinsicVpcBlacklist:
      - "{{resolve:ssm:SomeVPCReference:1}}"
      - !Ref MyVPC
    IntrinsicVpceWhitelist:
      - "{{resolve:ssm:SomeVPCEReference:1}}"
      - !Ref MyVPCE
```

RequestModel

Konfiguriert ein Anforderungsmodell für eine bestimmte API+Path+Methode.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer AWS Serverless Application Model (AWS SAM) -Vorlage zu deklarieren.

YAML

```
  Model: String
  Required: Boolean
  ValidateBody: Boolean
```

```
  ValidateParameters: Boolean
```

Eigenschaften

`Model`

Name eines Modells, das in der Models-Eigenschaft von definiert istAWS::Serverless::Api.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Required`

Fügt dem Parameterbereich der OpenApi Definition eine `required` Eigenschaft für den angegebenen API-Endpunkt hinzu.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`ValidateBody`

Gibt an, ob API Gateway den verwendet`Model`, um den Anfragetext zu validieren. Weitere Informationen finden Sie unter Aktivieren der Anforderungsvalidierung in API Gateway im API Gateway Developer Guide.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`ValidateParameters`

Gibt an, ob API Gateway die `Model` zur Überprüfung von Anforderungspfadparametern, Abfragezeichenfolgen und Headern verwendet. Weitere Informationen finden Sie unter Aktivieren der Anforderungsvalidierung in API Gateway im API Gateway Developer Guide.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

Anforderungsmodell

Beispiel für ein Modell anfordern

YAML

```
RequestModel:
  Model: User
  Required: true
  ValidateBody: true
  ValidateParameters: true
```

RequestParameter

Konfigurieren Sie den Anforderungsparameter für eine bestimmte API+Path+Methode.

Für den `Required` `Caching` Anforderungsparameter muss entweder die Eigenschaft oder angegeben werden

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
  Caching: Boolean
  Required: Boolean
```

Eigenschaften

`Caching`

Fügt `cacheKeyParameters` einen Abschnitt zur OpenApi API-Gateway-Definition hinzu

Typ: Boolesch

Required: Conditional

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Required`

Dieses Feld gibt an, ob ein Parameter erforderlich ist

Typ: Boolesch

Required: Conditional

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

Anfrageparameter

Beispiel für die Einstellung von Anforderungsparametern

YAML

```
RequestParameters:
  - method.request.header.Authorization:
      Required: true
      Caching: true
```

CloudWatchEvent

Das Objekt, das einen `CloudWatchEvent` Ereignisquellentyp beschreibt.

AWS Serverless Application Model (AWS SAM) generiert eine AWS::Events::RuleRessource, wenn dieser Ereignistyp gesetzt ist.

Wichtiger Hinweis: EventBridgeRule ist der bevorzugte Typ der Ereignisquelle, der anstelle von verwendet wird`CloudWatchEvent`. `EventBridgeRule`und `CloudWatchEvent` verwenden Sie denselben zugrunde liegenden Dienst, dieselbe API und dieselben CloudFormation Ressourcen. AWS SAM Wird jedoch nur Unterstützung für neue Funktionen hinzufügen`EventBridgeRule`.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
Enabled: Boolean
EventBusName: String
Input: String
InputPath: String
Pattern: EventPattern
State: String
```

Eigenschaften

`Enabled`

Gibt an, ob die Regel aktiviert ist.

Um die Regel zu deaktivieren, setzen Sie diese Eigenschaft auf`false`.

> **Note**
>
> Geben Sie entweder die `State` Eigenschaft `Enabled` oder an, aber nicht beide.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `State` Eigenschaft einer `AWS::Events::Rule` Ressource. Wenn diese Eigenschaft auf `true` gesetzt ist, gilt sie AWS SAM als erfolgreich`ENABLED`, andernfalls gilt sie als erfolgreich`DISABLED`.

`EventBusName`

Der Ereignisbus, der dieser Regel zugeordnet werden soll. Wenn Sie diese Eigenschaft weglassen, AWS SAM wird der Standard-Event-Bus verwendet.

Typ: Zeichenfolge

Required: No

Standard: Standard-Event-Bus

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `EventBusName` Eigenschaft einer `AWS::Events::Rule` Ressource übergeben.

`Input`

Gültiger JSON-Text wurde an das Ziel übergeben. Wenn Sie diese Eigenschaft verwenden, wird nichts aus dem Ereignistext selbst an das Ziel weitergeleitet.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Input` Eigenschaft einer `AWS::Events::Rule Target` Ressource übergeben.

`InputPath`

Wenn Sie nicht das gesamte übereinstimmende Ereignis an das Ziel übergeben möchten, verwenden Sie die `InputPath` Eigenschaft, um zu beschreiben, welcher Teil des Ereignisses übergeben werden soll.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `InputPath` Eigenschaft einer `AWS::Events::Rule Target` Ressource übergeben.

`Pattern`

Beschreibt, welche Ereignisse an das angegebene Ziel weitergeleitet werden. Weitere Informationen finden Sie unter Ereignisse und Ereignismuster EventBridge im EventBridge Amazon-Benutzerhandbuch.

Typ: EventPattern

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `EventPattern` Eigenschaft einer `AWS::Events::Rule` Ressource übergeben.

`State`

Der Status der Regel.

Zulässige Werte: `DISABLED | ENABLED`

> **Note**
>
> Geben Sie entweder die `State` Eigenschaft `Enabled` oder an, aber nicht beide.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `State` Eigenschaft einer `AWS::Events::Rule` Ressource übergeben.

Beispiele

CloudWatchEvent

Das Folgende ist ein Beispiel für einen `CloudWatchEvent` Ereignisquellentyp.

YAML

```
CWEvent:
  Type: CloudWatchEvent
  Properties:
    Enabled: false
    Input: '{"Key": "Value"}'
    Pattern:
      detail:
        state:
          - running
```

CloudWatchLogs

Das Objekt, das einen `CloudWatchLogs` Ereignisquellentyp beschreibt.

Dieses Ereignis generiert eine AWS::Logs::SubscriptionFilterRessource, spezifiziert einen Abonnementfilter und ordnet ihn der angegebenen Protokollgruppe zu.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
FilterPattern: String
LogGroupName: String
```

Eigenschaften

`FilterPattern`

Die Filterausdrücke, die einschränken, was an die AWS Zielressource übermittelt wird. Weitere Informationen zur Syntax von Filtermustern finden Sie unter Filters und Mustersyntax.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `FilterPattern` Eigenschaft einer `AWS::Logs::SubscriptionFilter` Ressource übergeben.

`LogGroupName`

Die Protokollgruppe, die mit dem Abonnementfilter verknüpft werden soll. Alle Protokollereignisse, die in diese Protokollgruppe hochgeladen werden, werden gefiltert und an die angegebene AWS Ressource übermittelt, wenn das Filtermuster mit den Protokollereignissen übereinstimmt.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `LogGroupName` Eigenschaft einer `AWS::Logs::SubscriptionFilter` Ressource übergeben.

Beispiele

Cloudwatchlogs-Abonnementfilter

Beispiel für einen Cloudwatchlogs-Abonnementfilter

YAML

```
CWLog:
  Type: CloudWatchLogs
  Properties:
    LogGroupName:
      Ref: CloudWatchLambdaLogsGroup
    FilterPattern: My pattern
```

Cognito

Das Objekt, das einen `Cognito` Ereignisquellentyp beschreibt.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
  Trigger: List
  UserPool: String
```

Eigenschaften

`Trigger`

Die Lambda-Auslösekonfiguration für den neuen Benutzerpool.

Typ: Liste

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `LambdaConfig` Eigenschaft einer `AWS::Cognito::UserPool` Ressource übergeben.

`UserPool`

Verweis auf, der in derselben Vorlage UserPool definiert ist

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

Cognito-Ereignis

Beispiel für ein Cognito-Event

YAML

```
CognitoUserPoolPreSignup:
  Type: Cognito
  Properties:
    UserPool:
      Ref: MyCognitoUserPool
    Trigger: PreSignUp
```

DocumentDB

Das Objekt, das einen `DocumentDB` Ereignisquellentyp beschreibt. Weitere Informationen finden Sie unter Using AWS Lambda with Amazon DocumentDB im AWS Lambda Developer Guide.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer AWS SAM Vorlage zu deklarieren.

YAML

```
BatchSize: Integer
Cluster: String
CollectionName: String
DatabaseName: String
Enabled: Boolean
FilterCriteria: FilterCriteria
FullDocument: String
KmsKeyArn: String
MaximumBatchingWindowInSeconds: Integer
SecretsManagerKmsKeyId: String
SourceAccessConfigurations: List
StartingPosition: String
```

```
StartingPositionTimestamp: Double
```

Eigenschaften

`BatchSize`

Gibt die maximale Anzahl der Elemente an, die in einem einzigen Stapel zurückzugeben werden.

Typ: Ganzzahl

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `BatchSize` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`Cluster`

Der Amazon-Ressourcenname (ARN) des Amazon DocumentDB-Clusters.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `EventSourceArn` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`CollectionName`

Der Name der Sammlung, die in der Datenbank verwendet werden soll. Wenn Sie keine Sammlung angeben, verbraucht Lambda alle Sammlungen.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `CollectionName` Eigenschaft eines `AWS::Lambda::EventSourceMapping` `DocumentDBEventSourceConfig` Datentyps übergeben.

`DatabaseName`

Der Name der Datenbank, die innerhalb des Amazon DocumentDB-Clusters verwendet werden soll.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `DatabaseName` Eigenschaft eines `AWS::Lambda::EventSourceMapping` `DocumentDBEventSourceConfig` Datentyps übergeben.

`Enabled`

Fall`true`, ist die Zuordnung der Ereignisquelle aktiv. Um die Abfrage und den Aufruf zu unterbrechen, legen Sie den Wert auf fest. `false`

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Enabled` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`FilterCriteria`

Ein Objekt, das die Kriterien definiert, die bestimmen, ob Lambda ein Ereignis verarbeiten soll. Weitere Informationen finden Sie unter Lambda-Ereignisfilterung im AWS Lambda Developer Guide.

Typ: FilterCriteria

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `FilterCriteria` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`FullDocument`

Legt fest, was Amazon DocumentDB bei Dokumentaktualisierungen an Ihren Event-Stream sendet. Wenn auf gesetzt`UpdateLookup`, sendet Amazon DocumentDB ein Delta, das die Änderungen beschreibt, zusammen mit einer Kopie des gesamten Dokuments. Andernfalls sendet Amazon DocumentDB nur einen Teil des Dokuments, das die Änderungen enthält.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `FullDocument` Eigenschaft eines `AWS::Lambda::EventSourceMapping` `DocumentDBEventSourceConfig` Datentyps übergeben.

`KmsKeyArn`

Der Amazon-Ressourcenname (ARN) des Schlüssels zur Verschlüsselung von Informationen im Zusammenhang mit diesem Ereignis.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `KmsKeyArn` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`MaximumBatchingWindowInSeconds`

Die maximale Zeitspanne zur Erfassung von Datensätzen vor dem Aufruf der Funktion in Sekunden.

Typ: Ganzzahl

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `MaximumBatchingWindowInSeconds` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`SecretsManagerKmsKeyId`

Die AWS Key Management Service (AWS KMS) Schlüssel-ID eines vom Kunden verwalteten Schlüssels von AWS Secrets Manager. Erforderlich, wenn Sie einen vom Kunden verwalteten Schlüssel von Secrets Manager mit einer Lambda-Ausführungsrolle verwenden, die die `kms:Decrypt` Berechtigung nicht enthält.

Der Wert dieser Eigenschaft ist eine UUID. Beispiel: `1abc23d4-567f-8ab9-cde0-1fab234c5d67`.

Typ: Zeichenfolge

Required: Conditional

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`SourceAccessConfigurations`

Ein Array des Authentifizierungsprotokolls oder des virtuellen Hosts. Geben Sie dies mithilfe des SourceAccessConfigurationsDatentyps an.

Für den Typ der `DocumentDB` Ereignisquelle ist der einzig gültige Konfigurationstyp`BASIC_AUTH`.

- `BASIC_AUTH`— Das Secrets Manager Manager-Geheimnis, in dem Ihre Broker-Anmeldeinformationen gespeichert werden. Für diesen Typ müssen die Anmeldeinformationen das folgende Format haben:`{"username": "your-username", "password": "your-password"}`. Nur ein Objekt des Typs `BASIC_AUTH` ist zulässig.

Typ: Liste

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `SourceAccessConfigurations` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`StartingPosition`

Die Position im Stream, an der mit dem Lesen begonnen wird.

- `AT_TIMESTAMP`— Geben Sie einen Zeitpunkt an, ab dem mit dem Lesen von Datensätzen begonnen werden soll.
- `LATEST`— Nur neue Datensätze lesen.
- `TRIM_HORIZON`— Verarbeitet alle verfügbaren Datensätze.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `StartingPosition` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`StartingPositionTimestamp`

Die Zeit, ab der mit dem Lesen begonnen werden soll, in Unix-Zeitsekunden. Definiert`StartingPositionTimestamp`, wann als angegeben `StartingPosition` ist`AT_TIMESTAMP`.

Type: Double

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `StartingPositionTimestamp` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

Beispiele

Amazon DocumentDB DocumentDB-Ereignisquelle

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
    ...
      Events:
        MyDDBEvent:
          Type: DocumentDB
          Properties:
            Cluster: "arn:aws:rds:us-west-2:123456789012:cluster:docdb-2023-01-01"
            BatchSize: 10
            MaximumBatchingWindowInSeconds: 5
            DatabaseName: "db1"
            CollectionName: "collection1"
            FullDocument: "UpdateLookup"
            SourceAccessConfigurations:
              - Type: BASIC_AUTH
                URI: "arn:aws:secretsmanager:us-west-2:123456789012:secret:doc-db"
```

DynamoDB

Das Objekt, das einen `DynamoDB` Ereignisquellentyp beschreibt. Weitere Informationen finden Sie unter Using AWS Lambda with Amazon DynamoDB im AWS Lambda Developer Guide.

AWS SAM generiert eine AWS::Lambda::EventSourceMappingRessource, wenn dieser Ereignistyp festgelegt ist.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```yaml
BatchSize: Integer
BisectBatchOnFunctionError: Boolean
DestinationConfig: DestinationConfig
Enabled: Boolean
FilterCriteria: FilterCriteria
FunctionResponseTypes: List
KmsKeyArn: String
MaximumBatchingWindowInSeconds: Integer
MaximumRecordAgeInSeconds: Integer
MaximumRetryAttempts: Integer
MetricsConfig: MetricsConfig
ParallelizationFactor: Integer
StartingPosition: String
StartingPositionTimestamp: Double
Stream: String
TumblingWindowInSeconds: Integer
```

Eigenschaften

`BatchSize`

Gibt die maximale Anzahl der Elemente an, die in einem einzigen Stapel zurückzugeben werden.

Typ: Ganzzahl

Required: No

Standard: 100

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `BatchSize` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

Minimum: 1

Maximum: `1000`

`BisectBatchOnFunctionError`

Wenn die Funktion einen Fehler zurückgibt, teilen Sie den Stapel in zwei Teile auf und versuchen Sie es erneut.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `BisectBatchOnFunctionError` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`DestinationConfig`

Eine Amazon Simple Queue Service (Amazon SQS) -Warteschlange oder ein Amazon Simple Notification Service (Amazon SNS) -Themenziel für verworfene Datensätze.

Typ: DestinationConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `DestinationConfig` Eigenschaft einer Ressource übergeben. `AWS::Lambda::EventSourceMapping`

`Enabled`

Deaktiviert den Ereignis-Quellzuweisung zum Anhalten und Aufrufen von Abfragen.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Enabled` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`FilterCriteria`

Ein Objekt, das die Kriterien definiert, anhand derer bestimmt wird, ob Lambda ein Ereignis verarbeiten soll. Weitere Informationen finden Sie unter AWS Lambda Ereignisfilterung im AWS Lambda Entwicklerhandbuch.

Typ: FilterCriteria

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `FilterCriteria` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`FunctionResponseTypes`

Eine Liste der Antworttypen, die derzeit auf die Ereignisquellenzuordnung angewendet werden. Weitere Informationen finden Sie unter Melden von Batch-Elementen im AWS Lambda -Leitfaden für Entwickler.

Gültige Werte: `ReportBatchItemFailures`

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `FunctionResponseTypes` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`KmsKeyArn`

Der Amazon-Ressourcenname (ARN) des Schlüssels zur Verschlüsselung von Informationen im Zusammenhang mit diesem Ereignis.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `KmsKeyArn` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`MaximumBatchingWindowInSeconds`

Die maximale Zeitspanne zur Erfassung von Datensätzen vor dem Aufruf der Funktion in Sekunden.

Typ: Ganzzahl

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `MaximumBatchingWindowInSeconds` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`MaximumRecordAgeInSeconds`

Das maximale Alter eines Datensatzes, den Lambda zur Verarbeitung an eine Funktion sendet.

Typ: Ganzzahl

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `MaximumRecordAgeInSeconds` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`MaximumRetryAttempts`

Die maximale Anzahl der Wiederholungen, wenn die Funktion einen Fehler zurückgibt.

Typ: Ganzzahl

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `MaximumRetryAttempts` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`MetricsConfig`

Eine optionale Konfiguration, um erweiterte Metriken für Zuordnungen von Ereignisquellen zu erhalten, die jede Phase der Verarbeitung erfassen. Ein Beispiel finden Sie unter MetricsConfig Ereignis.

Typ: MetricsConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `MetricsConfig` Eigenschaft einer Ressource übergeben. `AWS::Lambda::EventSourceMapping`

`ParallelizationFactor`

Die Anzahl der Batches, die von jedem Shard gleichzeitig verarbeitet werden sollen.

Typ: Ganzzahl

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `ParallelizationFactor` Eigenschaft einer Ressource übergeben. `AWS::Lambda::EventSourceMapping`

`StartingPosition`

Die Position im Stream, an der mit dem Lesen begonnen wird.

- `AT_TIMESTAMP`— Geben Sie einen Zeitpunkt an, ab dem mit dem Lesen von Datensätzen begonnen werden soll.
- `LATEST`— Nur neue Datensätze lesen.
- `TRIM_HORIZON`— Verarbeitet alle verfügbaren Datensätze.

Zulässige Werte: `AT_TIMESTAMP` | `LATEST` | `TRIM_HORIZON`

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `StartingPosition` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`StartingPositionTimestamp`

Die Zeit, ab der mit dem Lesen begonnen werden soll, in Unix-Zeitsekunden. Definiert`StartingPositionTimestamp`, wann als angegeben `StartingPosition` ist`AT_TIMESTAMP`.

Type: Double

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `StartingPositionTimestamp` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`Stream`

Der Amazon-Ressourcenname (ARN) des DynamoDB-Streams.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `EventSourceArn` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`TumblingWindowInSeconds`

Die Dauer eines Verarbeitungsfensters in Sekunden. Der gültige Bereich liegt zwischen 1 und 900 (15 Minuten).

Weitere Informationen finden Sie unter Tumbling windows im AWS Lambda Entwicklerhandbuch.

Typ: Ganzzahl

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `TumblingWindowInSeconds` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

Beispiele

MetricsConfig Ereignis

Im Folgenden finden Sie ein Beispiel für eine Ressource, die die `MetricsConfig` Eigenschaft verwendet, um jede Phase der Verarbeitung ihrer Ereignisquellenzuordnungen zu erfassen.

```
Resources:
  FilteredEventsFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: s3://sam-demo-bucket/metricsConfig.zip
      Handler: index.handler
      Runtime: nodejs16.x
      Events:
        KinesisStream:
          Type: Kinesis
          Properties:
            Stream: !GetAtt KinesisStream.Arn
            StartingPosition: LATEST
            MetricsConfig:
              Metrics:
              - EventCount
```

DynamoDB-Ereignisquelle für bestehende DynamoDB-Tabelle

DynamoDB-Ereignisquelle für eine DynamoDB-Tabelle, die bereits in einem Konto vorhanden ist. AWS

YAML

```
Events:
```

```
  DDBEvent:
    Type: DynamoDB
    Properties:
      Stream: arn:aws:dynamodb:us-east-1:123456789012:table/TestTable/
stream/2016-08-11T21:21:33.291
      StartingPosition: TRIM_HORIZON
      BatchSize: 10
      Enabled: false
```

DynamoDB-Ereignis für in der Vorlage deklarierte DynamoDB-Tabelle

DynamoDB-Ereignis für eine DynamoDB-Tabelle, die in derselben Vorlagendatei deklariert ist.

YAML

```
Events:
  DDBEvent:
    Type: DynamoDB
    Properties:
      Stream:
        !GetAtt MyDynamoDBTable.StreamArn   # This must be the name of a DynamoDB table
 declared in the same template file
      StartingPosition: TRIM_HORIZON
      BatchSize: 10
      Enabled: false
```

EventBridgeRule

Das Objekt, das einen `EventBridgeRule` Ereignisquellentyp beschreibt, der Ihre serverlose Funktion als Ziel einer EventBridge Amazon-Regel festlegt. Weitere Informationen finden Sie unter Was ist Amazon EventBridge? im EventBridge Amazon-Benutzerhandbuch.

AWS SAM generiert eine AWS::Events::RuleRessource, wenn dieser Ereignistyp festgelegt ist. AWS SAM erstellt auch eine `AWS::Lambda::Permission` Ressource, die benötigt wird, damit Lambda aufgerufen werden `EventBridgeRule` kann.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
DeadLetterConfig: DeadLetterConfig
EventBusName: String
Input: String
InputPath: String
InputTransformer: InputTransformer
Pattern: EventPattern
RetryPolicy: RetryPolicy
RuleName: String
State: String
Target: Target
```

Eigenschaften

`DeadLetterConfig`

Konfigurieren Sie die Amazon Simple Queue Service (Amazon SQS) -Warteschlange, über die Ereignisse nach einem fehlgeschlagenen Zielaufruf EventBridge gesendet werden. Der Aufruf kann beispielsweise fehlschlagen, wenn ein Ereignis an eine Lambda-Funktion gesendet wird, die nicht existiert, oder wenn EventBridge die Berechtigungen zum Aufrufen der Lambda-Funktion nicht ausreichen. Weitere Informationen finden Sie unter Richtlinien zur Wiederholung von Ereignissen und Verwenden von Warteschlangen mit unerlaubten Briefen im Amazon-Benutzerhandbuch. EventBridge

> **Note**
>
> Der AWS::Serverless: :Funktion Ressourcentyp hat einen ähnlichen Datentyp`DeadLetterQueue`, der Fehler behandelt, die nach einem erfolgreichen Aufruf der Lambda-Zielfunktion auftreten. Beispiele für diese Arten von Fehlern sind Lambda-Drosselung oder Fehler, die von der Lambda-Zielfunktion zurückgegeben werden. Weitere Informationen zur *`DeadLetterQueue`* Funktionseigenschaft finden Sie im Developer Guide unter Warteschlangen mit uneingeschränktem Inhalt.AWS Lambda

Typ: DeadLetterConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `DeadLetterConfig` Eigenschaft des Datentyps. `AWS::Events::Rule Target` Die AWS SAM Version dieser Eigenschaft

enthält zusätzliche Untereigenschaften für den Fall, dass Sie die Warteschlange AWS SAM für unzustellbare Briefe erstellen möchten.

`EventBusName`

Der Ereignisbus, der dieser Regel zugeordnet werden soll. Wenn Sie diese Eigenschaft weglassen, wird der AWS SAM Standardereignisbus verwendet.

Typ: Zeichenfolge

Required: No

Standard: Standard-Event-Bus

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `EventBusName` Eigenschaft einer `AWS::Events::Rule` Ressource übergeben.

`Input`

Gültiger JSON-Text wurde an das Ziel übergeben. Wenn Sie diese Eigenschaft verwenden, wird nichts aus dem Ereignistext selbst an das Ziel weitergeleitet.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Input` Eigenschaft einer `AWS::Events::Rule Target` Ressource übergeben.

`InputPath`

Wenn Sie nicht das gesamte übereinstimmende Ereignis an das Ziel übergeben möchten, verwenden Sie die `InputPath` Eigenschaft, um zu beschreiben, welcher Teil des Ereignisses übergeben werden soll.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `InputPath` Eigenschaft einer `AWS::Events::Rule Target` Ressource übergeben.

`InputTransformer`

Einstellungen, mit denen Sie benutzerdefinierte Eingaben für ein Ziel basierend auf bestimmten Ereignisdaten bereitstellen können. Sie können einzelne oder mehrere Schlüssel-Wert-Paare

aus dem Ereignis extrahieren und diese Daten dann verwenden, um benutzerdefinierte Eingaben an das Ziel zu senden. Weitere Informationen finden Sie unter Amazon EventBridge Input Transformation im EventBridge Amazon-Benutzerhandbuch.

Typ: InputTransformer

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `InputTransformer` Eigenschaft eines `AWS::Events::Rule Target` Datentyps übergeben.

`Pattern`

Beschreibt, welche Ereignisse an das angegebene Ziel weitergeleitet werden. Weitere Informationen finden Sie unter EventBridgeAmazon-Ereignisse und EventBridge -Ereignismuster im EventBridge Amazon-Benutzerhandbuch.

Typ: EventPattern

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `EventPattern` Eigenschaft einer `AWS::Events::Rule` Ressource übergeben.

`RetryPolicy`

Ein `RetryPolicy`-Objekt, das Informationen zu den Richtlinieneinstellungen für Wiederholungsversuche enthält. Weitere Informationen finden Sie unter Richtlinien zur Wiederholung von Ereignissen und Verwenden von Warteschlangen mit unerlaubten Briefen im Amazon-Benutzerhandbuch. EventBridge

Typ: RetryPolicy

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `RetryPolicy` Eigenschaft des Datentyps übergeben. `AWS::Events::Rule Target`

`RuleName`

Der Name der Regel.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Name` Eigenschaft einer `AWS::Events::Rule` Ressource übergeben.

`State`

Der Status der Regel.

Zulässige Werte: `DISABLED` | `ENABLED` | `ENABLED_WITH_ALL_CLOUDTRAIL_MANAGEMENT_EVENTS`

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `State` Eigenschaft einer `AWS::Events::Rule` Ressource übergeben.

`Target`

Die AWS Ressource, die EventBridge aufgerufen wird, wenn eine Regel ausgelöst wird. Sie können diese Eigenschaft verwenden, um die logische ID des Ziels anzugeben. Wenn diese Eigenschaft nicht angegeben ist, wird die logische ID des Ziels AWS SAM generiert.

Typ: Ziel

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `Targets` Eigenschaft einer `AWS::Events::Rule` Ressource. `Amazon EC2 RebootInstances API call`ist ein Beispiel für eine Zieleigenschaft. In der AWS SAM Version dieser Eigenschaft können Sie nur die logische ID eines einzelnen Ziels angeben.

Beispiele

EventBridgeRule

Im Folgenden finden Sie ein Beispiel für einen `EventBridgeRule` Ereignisquellentyp.

YAML

```
EBRule:
```

```
  Type: EventBridgeRule
  Properties:
    Input: '{"Key": "Value"}'
    Pattern:
      detail:
        state:
          - terminated
    RetryPolicy:
      MaximumRetryAttempts: 5
      MaximumEventAgeInSeconds: 900
    DeadLetterConfig:
      Type: SQS
      QueueLogicalId: EBRuleDLQ
    Target:
      Id: MyTarget
```

DeadLetterConfig

Das Objekt, das zur Angabe der Amazon Simple Queue Service (Amazon SQS) -Warteschlange verwendet wird, über die Ereignisse nach einem fehlgeschlagenen Zielaufruf EventBridge gesendet werden. Der Aufruf kann beispielsweise fehlschlagen, wenn ein Ereignis an eine Lambda-Funktion gesendet wird, die nicht existiert, oder wenn die Berechtigungen zum Aufrufen der Lambda-Funktion nicht ausreichen. Weitere Informationen finden Sie unter Richtlinien zur Wiederholung von Ereignissen und Verwenden von Warteschlangen mit unerlaubten Briefen im Amazon-Benutzerhandbuch. EventBridge

> **Note**
>
> Der AWS::Serverless: :Funktion Ressourcentyp hat einen ähnlichen Datentyp, `DeadLetterQueue` der Fehler behandelt, die nach einem erfolgreichen Aufruf der Lambda-Zielfunktion auftreten. Beispiele für diese Art von Fehler sind Lambda-Drosselung oder Fehler, die von der Lambda-Zielfunktion zurückgegeben werden. Weitere Informationen zur *`DeadLetterQueue`* Funktionseigenschaft finden Sie im Entwicklerhandbuch unter Warteschlangen mit uneingeschränktem Inhalt.AWS Lambda

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer AWS Serverless Application Model (AWS SAM) -Vorlage zu deklarieren.

YAML

```
Arn: String
QueueLogicalId: String
Type: String
```

Eigenschaften

`Arn`

Der Amazon-Ressourcenname (ARN) der Amazon SQS-Warteschlange, der als Ziel für die Warteschlange mit unzustellbaren Briefen angegeben wurde.

> **Note**
>
> Geben Sie entweder die `Type` Eigenschaft oder `Arn` die Eigenschaft an, aber nicht beide.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Arn` Eigenschaft des `AWS::Events::Rule` `DeadLetterConfig` Datentyps übergeben.

`QueueLogicalId`

Der benutzerdefinierte Name der Warteschlange mit unerlaubten Briefen, die sie AWS SAM erstellt, `Type` ist angegeben.

> **Note**
>
> Wenn die `Type` Eigenschaft nicht festgelegt ist, wird diese Eigenschaft ignoriert.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Type

Der Typ der Warteschlange. Wenn diese Eigenschaft festgelegt ist, wird AWS SAM automatisch eine Warteschlange für unzustellbare Nachrichten erstellt und die erforderliche ressourcenbasierte Richtlinie angehängt, um der Regelressource die Erlaubnis zu erteilen, Ereignisse an die Warteschlange zu senden.

> **Note**
>
> Geben Sie entweder die `Type` Eigenschaft oder die Eigenschaft an, aber nicht beide`Arn`.

Gültige Werte: `SQS`

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

DeadLetterConfig

DeadLetterConfig

YAML

```
DeadLetterConfig:
  Type: SQS
  QueueLogicalId: MyDLQ
```

Target

Konfiguriert die AWS Ressource, die EventBridge aufgerufen wird, wenn eine Regel ausgelöst wird.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
  Id: String
```

Eigenschaften

`Id`

Die logische ID des Ziels.

Der Wert von `Id` kann alphanumerische Zeichen, Punkte (`.`), Bindestriche () und Unterstriche (`-`) enthalten. _

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Id` Eigenschaft des Datentyps übergeben. `AWS::Events::Rule Target`

Beispiele

Target

YAML

```
EBRule:
  Type: EventBridgeRule
  Properties:
    Target:
      Id: MyTarget
```

HttpApi

Das Objekt, das eine Ereignisquelle mit Typ beschreibt HttpApi.

Wenn eine OpenApi Definition für den angegebenen Pfad und die angegebene Methode in der API vorhanden ist, fügt SAM den Abschnitt Lambda-Integration und -Sicherheit (falls zutreffend) für Sie hinzu.

Wenn in der API keine OpenApi Definition für den angegebenen Pfad und die angegebene Methode vorhanden ist, erstellt SAM diese Definition für Sie.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
ApiId: String
Auth: HttpApiFunctionAuth
Method: String
Path: String
PayloadFormatVersion: String
RouteSettings: RouteSettings
TimeoutInMillis: Integer
```

Eigenschaften

`ApiId`

Bezeichner einer in dieser Vorlage definierten AWS::Serverless::HttpApi Ressource.

Falls nicht definiert, wird eine AWS::Serverless::HttpApi Standardressource erstellt, die `ServerlessHttpApi` mithilfe eines generierten OpenApi Dokuments aufgerufen wird, das eine Vereinigung aller Pfade und Methoden enthält, die durch API-Ereignisse definiert sind, die in dieser Vorlage definiert sind und keine angeben`ApiId`.

Dies kann nicht auf eine in einer anderen Vorlage definierte AWS::Serverless::HttpApi Ressource verweisen.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Auth`

Authentifizierungskonfiguration für diese spezielle API+Path+Methode.

Nützlich, um die APIs zu überschreiben `DefaultAuthorizer` oder die Authentifizierungskonfiguration für einen einzelnen Pfad festzulegen, wenn keine angegeben ist.
`DefaultAuthorizer`

Typ: HttpApiFunctionAuth

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Method`

HTTP-Methode, für die diese Funktion aufgerufen wird.

Wenn nein `Path` und angegeben `Method` sind, erstellt SAM einen Standard-API-Pfad, der alle Anfragen weiterleitet, die nicht einem anderen Endpunkt dieser Lambda-Funktion zugeordnet sind. Pro API kann nur einer dieser Standardpfade existieren.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Path`

URI-Pfad, für den diese Funktion aufgerufen wird. Muss mit `/` beginnen.

Wenn nein `Path` und angegeben `Method` sind, erstellt SAM einen Standard-API-Pfad, der alle Anfragen weiterleitet, die nicht einem anderen Endpunkt dieser Lambda-Funktion zugeordnet sind. Pro API kann nur einer dieser Standardpfade existieren.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`PayloadFormatVersion`

Gibt das Format der an eine Integration gesendeten Nutzlast an.

HINWEIS: PayloadFormatVersion erfordert, dass SAM Ihre OpenAPI-Definition ändert, sodass es nur mit Inline funktioniert, die in der `DefinitionBody` Eigenschaft OpenApi definiert ist.

Typ: Zeichenfolge

Required: No

Standard: 2.0

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`RouteSettings`

Die Routeneinstellungen pro Route für diese HTTP-API. Weitere Informationen zu Routeneinstellungen finden Sie AWS::ApiGatewayV2::Stage RouteSettingsim API Gateway Developer Guide.

Hinweis: Wenn sie sowohl in der HttpApi Ressource als auch in der Ereignisquelle angegeben RouteSettings sind, werden sie mit den Eigenschaften der Ereignisquelle AWS SAM zusammengeführt, die Vorrang haben.

Typ: RouteSettings

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `RouteSettings` Eigenschaft einer `AWS::ApiGatewayV2::Stage` Ressource übergeben.

`TimeoutInMillis`

Benutzerdefinierte Zeitüberschreitung zwischen 50 und 29.000 Millisekunden.

HINWEIS: TimeoutInMillis erfordert, dass SAM Ihre OpenAPI-Definition ändert, sodass es nur mit Inline funktioniert, die in der `DefinitionBody` Eigenschaft OpenApi definiert ist.

Typ: Ganzzahl

Required: No

Standard: 5000

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

HttpApi Standardereignis

HttpApi Ereignis, das den Standardpfad verwendet. Alle nicht zugewiesenen Pfade und Methoden auf dieser API werden zu diesem Endpunkt weitergeleitet.

YAML

```
Events:
  HttpApiEvent:
    Type: HttpApi
```

HttpApi

HttpApi Ereignis, das einen bestimmten Pfad und eine bestimmte Methode verwendet.

YAML

```
Events:
  HttpApiEvent:
    Type: HttpApi
    Properties:
      Path: /
      Method: GET
```

HttpApi Autorisierung

HttpApi Ereignis, das einen Authorizer verwendet.

YAML

```
Events:
  HttpApiEvent:
    Type: HttpApi
    Properties:
      Path: /authenticated
      Method: GET
      Auth:
        Authorizer: OpenIdAuth
        AuthorizationScopes:
          - scope1
```

```
          - scope2
```

HttpApiFunctionAuth

Konfiguriert die Autorisierung auf Veranstaltungsebene.

Konfigurieren Sie Auth für eine bestimmte API + Path + Methode

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer AWS Serverless Application Model (AWS SAM) -Vorlage zu deklarieren.

YAML

```
AuthorizationScopes: List
Authorizer: String
```

Eigenschaften

`AuthorizationScopes`

Die Autorisierungsbereiche, die für diese API, diesen Pfad und diese Methode gelten sollen.

Die hier aufgelisteten Bereiche setzen alle Bereiche außer Kraft, die von angewendet werden, `DefaultAuthorizer` falls vorhanden.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Authorizer`

Die `Authorizer` für eine bestimmte Funktion. Um die IAM-Autorisierung zu verwenden, geben Sie `EnableIamAuthorizer` im `Globals` Abschnitt Ihrer Vorlage die Angabe `AWS_IAM` und Spezifizierung `true` für an.

Wenn Sie in der API einen Global Authorizer angegeben haben und eine bestimmte Funktion öffentlich machen möchten, überschreiben Sie dies, indem Sie auf setzen`Authorizer`. `NONE`

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

Funktions-Auth

Autorisierung auf Funktionsebene angeben

YAML

```
Auth:
  Authorizer: OpenIdAuth
  AuthorizationScopes:
    - scope1
    - scope2
```

IAM-Autorisierung

Gibt die IAM-Autorisierung auf Ereignisebene an. Um die `AWS_IAM` Autorisierung auf Veranstaltungsebene zu verwenden, müssen Sie `EnableIamAuthorizer` im `Globals` Abschnitt Ihrer Vorlage auch `true` for angeben. Weitere Informationen finden Sie unter Sektion Globales der AWS SAM Vorlage.

YAML

```
Globals:
  HttpApi:
    Auth:
      EnableIamAuthorizer: true

Resources:
  HttpApiFunctionWithIamAuth:
    Type: AWS::Serverless::Function
    Properties:
      Events:
        ApiEvent:
```

```
          Type: HttpApi
          Properties:
            Path: /iam-auth
            Method: GET
            Auth:
              Authorizer: AWS_IAM
      Handler: index.handler
      InlineCode: |
        def handler(event, context):
          return {'body': 'HttpApiFunctionWithIamAuth', 'statusCode': 200}
      Runtime: python3.9
```

IoTRule

Das Objekt, das einen `IoTRule` Ereignisquellentyp beschreibt.

Erstellt eine AWS::IoT::TopicRuleRessource zur Deklaration einer AWS IoT Regel. Weitere Informationen finden Sie in der CloudFormation Dokumentation

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer AWS Serverless Application Model (AWS SAM) -Vorlage zu deklarieren.

YAML

```
AwsIotSqlVersion: String
Sql: String
```

Eigenschaften

`AwsIotSqlVersion`

Die Version der SQL-Regel-Engine, die beim Auswerten der Regel verwendet wird.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `AwsIotSqlVersion` Eigenschaft einer `AWS::IoT::TopicRule TopicRulePayload` Ressource übergeben.

`Sql`

Die SQL-Anweisung für die Abfrage des Themas. Weitere Informationen finden Sie in der AWS IoT SQL-Referenz im AWS IoT Entwicklerhandbuch.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Sql` Eigenschaft einer `AWS::IoT::TopicRule TopicRulePayload` Ressource übergeben.

Beispiele

IOT-Regel

Beispiel für eine IOT-Regel

YAML

```
IoTRule:
  Type: IoTRule
  Properties:
    Sql: SELECT * FROM 'topic/test'
```

Kinesis

Das Objekt, das einen `Kinesis` Ereignisquellentyp beschreibt. Weitere Informationen finden Sie unter Using AWS Lambda with Amazon Kinesis im AWS Lambda Developer Guide.

AWS SAM generiert eine AWS::Lambda::EventSourceMappingRessource, wenn dieser Ereignistyp festgelegt ist.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
  BatchSize: Integer
```

```
  BisectBatchOnFunctionError: Boolean
  DestinationConfig: DestinationConfig
  Enabled: Boolean
  FilterCriteria: FilterCriteria
  FunctionResponseTypes: List
  KmsKeyArn: String
  MaximumBatchingWindowInSeconds: Integer
  MaximumRecordAgeInSeconds: Integer
  MaximumRetryAttempts: Integer
  MetricsConfig: MetricsConfig
  ParallelizationFactor: Integer
  StartingPosition: String
  StartingPositionTimestamp: Double
  Stream: String
  TumblingWindowInSeconds: Integer
```

Eigenschaften

`BatchSize`

Gibt die maximale Anzahl der Elemente an, die in einem einzigen Stapel zurückzugeben werden.

Typ: Ganzzahl

Required: No

Standard: 100

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `BatchSize` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

Minimum: 1

Maximum: `10000`

`BisectBatchOnFunctionError`

Wenn die Funktion einen Fehler zurückgibt, teilen Sie den Stapel in zwei Teile auf und versuchen Sie es erneut.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `BisectBatchOnFunctionError` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`DestinationConfig`

Eine Amazon Simple Queue Service (Amazon SQS) -Warteschlange oder ein Amazon Simple Notification Service (Amazon SNS) -Themenziel für verworfene Datensätze.

Typ: DestinationConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `DestinationConfig` Eigenschaft einer Ressource übergeben. `AWS::Lambda::EventSourceMapping`

`Enabled`

Deaktiviert den Ereignis-Quellzuweisung zum Anhalten und Aufrufen von Abfragen.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Enabled` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`FilterCriteria`

Ein Objekt, das die Kriterien definiert, anhand derer bestimmt wird, ob Lambda ein Ereignis verarbeiten soll. Weitere Informationen finden Sie unter AWS Lambda Ereignisfilterung im AWS Lambda Entwicklerhandbuch.

Typ: FilterCriteria

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `FilterCriteria` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`FunctionResponseTypes`

Eine Liste der Antworttypen, die derzeit auf die Ereignisquellenzuordnung angewendet werden. Weitere Informationen finden Sie unter Melden von Batch-Elementen im AWS Lambda -Leitfaden für Entwickler.

Gültige Werte: `ReportBatchItemFailures`

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `FunctionResponseTypes` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`KmsKeyArn`

Der Amazon-Ressourcenname (ARN) des Schlüssels zur Verschlüsselung von Informationen im Zusammenhang mit diesem Ereignis.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `KmsKeyArn` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`MaximumBatchingWindowInSeconds`

Die maximale Zeitspanne zur Erfassung von Datensätzen vor dem Aufruf der Funktion in Sekunden.

Typ: Ganzzahl

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `MaximumBatchingWindowInSeconds` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`MaximumRecordAgeInSeconds`

Das maximale Alter eines Datensatzes, den Lambda zur Verarbeitung an eine Funktion sendet.

Typ: Ganzzahl

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `MaximumRecordAgeInSeconds` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`MaximumRetryAttempts`

Die maximale Anzahl der Wiederholungen, wenn die Funktion einen Fehler zurückgibt.

Typ: Ganzzahl

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `MaximumRetryAttempts` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`MetricsConfig`

Eine optionale Konfiguration, um erweiterte Metriken für Zuordnungen von Ereignisquellen zu erhalten, die jede Phase der Verarbeitung erfassen. Ein Beispiel finden Sie unter MetricsConfig Ereignis.

Typ: MetricsConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `MetricsConfig` Eigenschaft einer Ressource übergeben. `AWS::Lambda::EventSourceMapping`

`ParallelizationFactor`

Die Anzahl der Batches, die von jedem Shard gleichzeitig verarbeitet werden sollen.

Typ: Ganzzahl

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `ParallelizationFactor` Eigenschaft einer Ressource übergeben. `AWS::Lambda::EventSourceMapping`

`StartingPosition`

Die Position im Stream, an der mit dem Lesen begonnen wird.

- `AT_TIMESTAMP`— Geben Sie einen Zeitpunkt an, ab dem mit dem Lesen von Datensätzen begonnen werden soll.
- `LATEST`— Nur neue Datensätze lesen.
- `TRIM_HORIZON`— Verarbeitet alle verfügbaren Datensätze.

Zulässige Werte: `AT_TIMESTAMP` | `LATEST` | `TRIM_HORIZON`

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `StartingPosition` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`StartingPositionTimestamp`

Die Zeit, ab der mit dem Lesen begonnen werden soll, in Unix-Zeitsekunden. Definiert`StartingPositionTimestamp`, wann als angegeben `StartingPosition` ist`AT_TIMESTAMP`.

Type: Double

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `StartingPositionTimestamp` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`Stream`

Der Amazon-Ressourcenname (ARN) des Datenstroms oder eines Stream-Verbrauchers.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `EventSourceArn` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`TumblingWindowInSeconds`

Die Dauer eines Verarbeitungsfensters in Sekunden. Der gültige Bereich liegt zwischen 1 und 900 (15 Minuten).

Weitere Informationen finden Sie unter Tumbling windows im AWS Lambda Entwicklerhandbuch.

Typ: Ganzzahl

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `TumblingWindowInSeconds` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

Beispiele

MetricsConfig Ereignis

Im Folgenden finden Sie ein Beispiel für eine Ressource, die die `MetricsConfig` Eigenschaft verwendet, um jede Phase der Verarbeitung ihrer Ereignisquellenzuordnungen zu erfassen.

```
Resources:
  FilteredEventsFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: s3://sam-demo-bucket/metricsConfig.zip
      Handler: index.handler
      Runtime: nodejs16.x
      Events:
        KinesisStream:
          Type: Kinesis
          Properties:
            Stream: !GetAtt KinesisStream.Arn
            StartingPosition: LATEST
            MetricsConfig:
              Metrics:
              - EventCount
```

Kinesis-Ereignisquelle

Im Folgenden finden Sie ein Beispiel für eine Kinesis-Ereignisquelle.

YAML

```
Events:
  KinesisEvent:
    Type: Kinesis
    Properties:
      Stream: arn:aws:kinesis:us-east-1:123456789012:stream/my-stream
      StartingPosition: TRIM_HORIZON
      BatchSize: 10
      Enabled: false
```

```
      FilterCriteria:
        Filters:
          - Pattern: '{"key": ["val1", "val2"]}'
```

MQ

Das Objekt, das einen `MQ` Ereignisquellentyp beschreibt. Weitere Informationen finden Sie unter Using Lambda with Amazon MQ im AWS Lambda Developer Guide.

AWS Serverless Application Model (AWS SAM) generiert eine AWS::Lambda::EventSourceMappingRessource, wenn dieser Ereignistyp festgelegt ist.

> **Note**
>
> Um eine Amazon MQ MQ-Warteschlange in einer Virtual Private Cloud (VPC) zu haben, die eine Verbindung zu einer Lambda-Funktion in einem öffentlichen Netzwerk herstellt, muss die Ausführungsrolle Ihrer Funktion die folgenden Berechtigungen beinhalten:
>
> - `ec2:CreateNetworkInterface`
> - `ec2:DeleteNetworkInterface`
> - `ec2:DescribeNetworkInterfaces`
> - `ec2:DescribeSecurityGroups`
> - `ec2:DescribeSubnets`
> - `ec2:DescribeVpcs`
>
> Weitere Informationen finden Sie unter Berechtigungen für Ausführungsrollen im Entwicklerhandbuch.AWS Lambda

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer AWS SAM Vorlage zu deklarieren.

YAML

```
BatchSize: Integer
Broker: String
DynamicPolicyName: Boolean
Enabled: Boolean
```

```
  FilterCriteria: FilterCriteria
  KmsKeyArn: String
  MaximumBatchingWindowInSeconds: Integer
  Queues: List
  SecretsManagerKmsKeyId: String
  SourceAccessConfigurations: List
```

Eigenschaften

`BatchSize`

Gibt die maximale Anzahl der Elemente an, die in einem einzigen Stapel zurückzugeben werden.

Typ: Ganzzahl

Required: No

Standard: 100

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `BatchSize` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

Minimum: 1

Maximum: `10000`

`Broker`

Der Amazon-Ressourcenname (ARN) des Amazon MQ-Brokers.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `EventSourceArn` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`DynamicPolicyName`

Standardmäßig dient der Richtlinienname AWS Identity and Access Management (IAM) der `SamAutoGeneratedAMQPolicy` Abwärtskompatibilität. Geben Sie `true` an, dass ein automatisch generierter Name für Ihre IAM-Richtlinie verwendet werden soll. Dieser Name beinhaltet die logische ID der Amazon MQ MQ-Ereignisquelle.

> **Note**
>
> Wenn Sie mehr als eine Amazon MQ MQ-Ereignisquelle verwenden, geben Sie dies an, `true` um doppelte IAM-Richtliniennamen zu vermeiden.

Typ: Boolesch

Required: No

Standardwert: `false`

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Enabled`

Fall`true`, ist die Zuordnung der Ereignisquelle aktiv. Um die Abfrage und den Aufruf zu unterbrechen, legen Sie den Wert auf fest. `false`

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Enabled` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`FilterCriteria`

Ein Objekt, das die Kriterien definiert, die bestimmen, ob Lambda ein Ereignis verarbeiten soll. Weitere Informationen finden Sie unter AWS Lambda Ereignisfilterung im AWS Lambda Entwicklerhandbuch.

Typ: FilterCriteria

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `FilterCriteria` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`KmsKeyArn`

Der Amazon-Ressourcenname (ARN) des Schlüssels zur Verschlüsselung von Informationen zu diesem Ereignis.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `KmsKeyArn` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`MaximumBatchingWindowInSeconds`

Die maximale Zeitspanne zur Erfassung von Datensätzen vor dem Aufruf der Funktion in Sekunden.

Typ: Ganzzahl

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `MaximumBatchingWindowInSeconds` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`Queues`

Der Name der zu verwendenden Zielwarteschlange des Amazon-MQ-Brokers.

Typ: Liste

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Queues` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`SecretsManagerKmsKeyId`

Die AWS Key Management Service (AWS KMS) Schlüssel-ID eines vom Kunden verwalteten Schlüssels von AWS Secrets Manager. Erforderlich, wenn Sie einen vom Kunden verwalteten Schlüssel von Secrets Manager mit einer Lambda-Ausführungsrolle verwenden, die die `kms:Decrypt` Berechtigung nicht enthält.

Der Wert dieser Eigenschaft ist eine UUID. Beispiel: `1abc23d4-567f-8ab9-cde0-1fab234c5d67`.

Typ: Zeichenfolge

Required: Conditional

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`SourceAccessConfigurations`

Ein Array des Authentifizierungsprotokolls oder des virtuellen Hosts. Geben Sie dies mithilfe des SourceAccessConfigurationsDatentyps an.

Für den Typ der `MQ` Ereignisquelle sind die einzig gültigen Konfigurationstypen `BASIC_AUTH` und`VIRTUAL_HOST`.

- **`BASIC_AUTH`**— Das Secrets Manager Manager-Geheimnis, das Ihre Broker-Anmeldeinformationen speichert. Für diesen Typ müssen die Anmeldeinformationen das folgende Format haben:`{"username": "your-username", "password": "your-password"}`. Nur ein Objekt des Typs `BASIC_AUTH` ist zulässig.
- **`VIRTUAL_HOST`**— Der Name des virtuellen Hosts in Ihrem RabbitMQ-Broker. Lambda verwendet den Host dieses Rabbit MQ als Ereignisquelle. Nur ein Objekt des Typs `VIRTUAL_HOST` ist zulässig.

Typ: Liste

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `SourceAccessConfigurations` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

Beispiele

Amazon MQ MQ-Ereignisquelle

Im Folgenden finden Sie ein Beispiel für einen `MQ` Ereignisquellentyp für einen Amazon MQ-Broker.

YAML

```
Events:
  MQEvent:
    Type: MQ
    Properties:
      Broker: arn:aws:mq:us-
east-2:123456789012:broker:MyBroker:b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9
      Queues: List of queues
```

```
      SourceAccessConfigurations:
        - Type: BASIC_AUTH
          URI: arn:aws:secretsmanager:us-east-1:01234567890:secret:MyBrokerSecretName
      BatchSize: 200
      Enabled: true
```

MSK

Das Objekt, das einen `MSK` Ereignisquellentyp beschreibt. Weitere Informationen finden Sie unter Using AWS Lambda with Amazon MSK im AWS Lambda Developer Guide.

AWS Serverless Application Model (AWS SAM) generiert eine AWS::Lambda::EventSourceMappingRessource, wenn dieser Ereignistyp festgelegt ist.

Um Schema Registry verwenden zu können, müssen Sie spezifische IAM-Rollenberechtigungen für Ihre Funktion definieren. Ein Beispiel für die erforderliche Konfiguration finden Sie unter Vollständige Einrichtung mit IAM-Rollen.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer AWS SAM Vorlage zu deklarieren.

YAML

```
  BatchSize: Integer
  BisectBatchOnFunctionError: Boolean
  ConsumerGroupId: String
  DestinationConfig: DestinationConfig
  Enabled: Boolean
  FilterCriteria: FilterCriteria
  FunctionResponseTypes: List
  KmsKeyArn: String
  MaximumBatchingWindowInSeconds: Integer
  MaximumRecordAgeInSeconds: Integer
  MaximumRetryAttempts: Integer
  LoggingConfig: LoggingConfig
  MetricsConfig: MetricsConfig
  ProvisionedPollerConfig: ProvisionedPollerConfig
  SchemaRegistryConfig: SchemaRegistryConfig
  SourceAccessConfigurations: SourceAccessConfigurations
  StartingPosition: String
  StartingPositionTimestamp: Double
  Stream: String
```

```
  Topics: List
```

Eigenschaften

`BatchSize`

Die maximale Anzahl von Datensätzen in jedem Batch, die Lambda aus Ihrem Stream oder der Warteschlange abruft und an Ihre Funktion sendet. Lambda übergibt alle Datensätze im Batch in einem einzigen Aufruf an die Funktion, bis zur Nutzlastgrenze für den synchronen Aufruf (6 MB).

Standard: 100

Gültiger Bereich: Mindestwert von 1. Höchstwert von 10.000.

Typ: Ganzzahl

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `BatchSize` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`BisectBatchOnFunctionError`

Wenn die Funktion einen Fehler zurückgibt, teilen Sie den Stapel in zwei Teile auf und versuchen Sie es erneut.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `BisectBatchOnFunctionError` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`ConsumerGroupId`

Eine Zeichenfolge, die konfiguriert, wie Ereignisse aus Kafka-Themen gelesen werden.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `AmazonManagedKafkaConfiguration` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`DestinationConfig`

Ein Konfigurationsobjekt, das das Ziel eines Ereignisses angibt, nachdem Lambda es verarbeitet hat.

Verwenden Sie diese Eigenschaft, um das Ziel fehlgeschlagener Aufrufe aus der Amazon MSK-Ereignisquelle anzugeben.

Typ: DestinationConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `DestinationConfig` Eigenschaft einer Ressource übergeben. `AWS::Lambda::EventSourceMapping`

`Enabled`

Deaktiviert den Ereignis-Quellzuweisung zum Anhalten und Aufrufen von Abfragen.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Enabled` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`FilterCriteria`

Ein Objekt, das die Kriterien definiert, die bestimmen, ob Lambda ein Ereignis verarbeiten soll. Weitere Informationen finden Sie unter AWS Lambda Ereignisfilterung im AWS Lambda Entwicklerhandbuch.

Typ: FilterCriteria

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `FilterCriteria` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`FunctionResponseTypes`

Eine Liste der Antworttypen, die derzeit auf die Ereignisquellenzuordnung angewendet werden. Weitere Informationen finden Sie unter Melden von Batch-Elementen im AWS Lambda -Leitfaden für Entwickler.

Gültige Werte: `ReportBatchItemFailures`

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `FunctionResponseTypes` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`KmsKeyArn`

Der Amazon-Ressourcenname (ARN) des Schlüssels zur Verschlüsselung von Informationen im Zusammenhang mit diesem Ereignis.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `KmsKeyArn` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`MaximumBatchingWindowInSeconds`

Die maximale Zeitspanne zur Erfassung von Datensätzen vor dem Aufruf der Funktion in Sekunden.

Typ: Ganzzahl

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `MaximumBatchingWindowInSeconds` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`MaximumRecordAgeInSeconds`

Das maximale Alter eines Datensatzes, den Lambda zur Verarbeitung an eine Funktion sendet.

Typ: Ganzzahl

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `MaximumRecordAgeInSeconds` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`MaximumRetryAttempts`

Die maximale Anzahl der Wiederholungen, wenn die Funktion einen Fehler zurückgibt.

Typ: Ganzzahl

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `MaximumRetryAttempts` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`LoggingConfig`

Ein Konfigurationsobjekt, das die Protokollierungskonfiguration für die Zuordnung der Ereignisquellen angibt.

Typ: LoggingConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `LoggingConfig` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`MetricsConfig`

Ein Konfigurationsobjekt, das die Metrikkonfiguration für die Zuordnung der Ereignisquellen angibt.

Typ: MetricsConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `MetricsConfig` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`ProvisionedPollerConfig`

Konfiguration zur Erhöhung der Anzahl von Pollern, die zur Berechnung von Ereignisquellenzuordnungen verwendet werden. Diese Konfiguration ermöglicht mindestens einen Poller und ein Maximum von 2000 Pollern. Ein Beispiel finden Sie unter. ProvisionedPollerConfig Beispiel

Typ: ProvisionedPollerConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `ProvisionedPollerConfig` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`SchemaRegistryConfig`

Konfiguration für die Verwendung einer Schemaregistrierung mit der Kafka-Ereignisquelle.

> **Note**
>
> Diese Funktion `ProvisionedPollerConfig` muss konfiguriert werden.

Typ: SchemaRegistryConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `AmazonManagedKafkaEventSourceConfig` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`SourceAccessConfigurations`

Ein Array des Authentifizierungsprotokolls, der VPC-Komponenten oder des virtuellen Hosts zum Sichern und Definieren Ihrer Ereignisquelle.

Gültige Werte: `CLIENT_CERTIFICATE_TLS_AUTH`

Typ: Liste von SourceAccessConfiguration

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist Teil der AmazonManagedKafkaEventSourceConfigEigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource.

`StartingPosition`

Die Position im Stream, an der mit dem Lesen begonnen wird.

- `AT_TIMESTAMP`— Geben Sie einen Zeitpunkt an, ab dem mit dem Lesen von Datensätzen begonnen werden soll.
- `LATEST`— Nur neue Datensätze lesen.

- `TRIM_HORIZON`— Verarbeitet alle verfügbaren Datensätze.

Zulässige Werte: `AT_TIMESTAMP` | `LATEST` | `TRIM_HORIZON`

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `StartingPosition` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`StartingPositionTimestamp`

Die Zeit, ab der mit dem Lesen begonnen werden soll, in Unix-Zeitsekunden. Definiert`StartingPositionTimestamp`, wann als angegeben `StartingPosition` ist`AT_TIMESTAMP`.

Type: Double

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `StartingPositionTimestamp` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`Stream`

Der Amazon-Ressourcenname (ARN) des Datenstroms oder eines Stream-Verbrauchers.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `EventSourceArn` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`Topics`

Der Name des Kafka-Themas.

Typ: Liste

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Topics` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

Beispiele

Vollständige Einrichtung mit IAM-Rollen

Das folgende Beispiel zeigt ein vollständiges Setup einschließlich der erforderlichen IAM-Rollenkonfiguration für die Verwendung von Schema Registry:

```
Parameters:
  PreCreatedSubnetOne:
    Type: String
  PreCreatedSubnetTwo:
    Type: String
  MskClusterName4:
    Type: String

Resources:
  MyLambdaExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Action: [sts:AssumeRole]
          Effect: Allow
          Principal:
            Service: [lambda.amazonaws.com]
      Policies:
      - PolicyName: KafkaClusterPermissions
        PolicyDocument:
          Statement:
          - Action: [kafka:DescribeClusterV2, kafka:GetBootstrapBrokers]
            Effect: Allow
            Resource: 'arn:aws:kafka:us-east-1:123456789012:cluster/*'
      - PolicyName: KafkaAuthPolicy
        PolicyDocument:
          Statement:
          - Action: [secretsmanager:GetSecretValue, kms:Decrypt]
            Effect: "Allow"
            Resource: ['arn:aws:secretsmanager:us-
west-2:123456789012:secret:kafkaSecret-******',
                       'arn:aws:kms:us-west-2:123456789012:key/keyId']
      - PolicyName: ENIPolicy
        PolicyDocument:
          Statement:
```

```
          - Action: [ec2:CreateNetworkInterface,
              ec2:DescribeNetworkInterfaces, ec2:DescribeVpcs,
 ec2:DeleteNetworkInterface,
              ec2:DescribeSubnets, ec2:DescribeSecurityGroups]
            Effect: Allow
            Resource: '*'
      - PolicyName: SchemaRegistryPolicy
        PolicyDocument:
          Statement:
          - Action: [glue:GetRegistry]
            Effect: Allow
            Resource: 'arn:aws:glue:{region}:{account-id}:registry/{registry-name}'
      - PolicyName: SchemaVersionsPolicy
        PolicyDocument:
          Statement:
          - Action: [glue:GetSchemaVersions]
            Effect: Allow
            Resource: '*'
      ManagedPolicyArns:
      - !Sub arn:${AWS::Partition}:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole
      Tags:
      - {Value: SAM, Key: lambda:createdBy}

  MyMskCluster:
    Type: AWS::MSK::Cluster
    Properties:
      BrokerNodeGroupInfo:
        ClientSubnets:
        - Ref: PreCreatedSubnetOne
        - Ref: PreCreatedSubnetTwo
        InstanceType: kafka.t3.small
        StorageInfo:
          EBSStorageInfo:
            VolumeSize: 1
      ClusterName:
        Ref: MskClusterName4
      KafkaVersion: 3.8.x
      NumberOfBrokerNodes: 2

  MyMskStreamProcessor:
    Type: AWS::Serverless::Function
    Properties:
      Runtime: nodejs18.x
```

```
      Handler: index.handler
      CodeUri: ${codeuri}
      Role:
        Fn::GetAtt: [MyLambdaExecutionRole, Arn]
      Events:
        MyMskEvent:
          Type: MSK
          Properties:
            StartingPosition: LATEST
            Stream:
              Ref: MyMskCluster
            SourceAccessConfigurations:
            - Type: SASL_SCRAM_512_AUTH
              URI: !Sub arn:${AWS::Partition}:secretsmanager:us-
west-2:123456789012:secret:my-path/my-secret-name-1a2b3c
            Topics:
            - SchemaRegistryTestTopic
            ProvisionedPollerConfig:
              MinimumPollers: 1
            SchemaRegistryConfig:
              AccessConfigs:
              - Type: BASIC_AUTH
                URI: !Sub arn:${AWS::Partition}:secretsmanager:us-
west-2:123456789012:secret:my-path/my-secret-name-1a2b3c
              SchemaValidationConfigs:
              - Attribute: KEY
              EventRecordFormat: JSON
              SchemaRegistryURI: !Sub arn:${AWS::Partition}:glue:us-
west-2:123456789012:registry/myregistry
```

ProvisionedPollerConfig Beispiel

```
ProvisionedPollerConfig:
  MinimumPollers: 1
  MaximumPollers: 200
```

Amazon MSK-Beispiel für einen vorhandenen Cluster

Im Folgenden finden Sie ein Beispiel für einen `MSK` Ereignisquellentyp für einen Amazon MSK-Cluster, der bereits in einem AWS-Konto vorhanden ist.

YAML

```
Events:
  MSKEvent:
    Type: MSK
    Properties:
      StartingPosition: LATEST
      Stream: arn:aws:kafka:us-east-1:012345678012:cluster/exampleClusterName/
abcdefab-1234-abcd-5678-cdef0123ab01-2
      Topics:
        - MyTopic
```

Amazon MSK-Beispiel für Cluster, der in derselben Vorlage deklariert wurde

Im Folgenden finden Sie ein Beispiel für einen `MSK` Ereignisquellentyp für einen Amazon MSK-Cluster, der in derselben Vorlagendatei deklariert ist.

YAML

```
Events:
  MSKEvent:
    Type: MSK
    Properties:
      StartingPosition: LATEST
      Stream:
        Ref: MyMskCluster   # This must be the name of an MSK cluster declared in the
 same template file
      Topics:
        - MyTopic
```

MSK-Ereignisquelle mit Schemaregistrierung

Im Folgenden finden Sie ein Beispiel für einen `MSK` Ereignisquellentyp, der mit einer Schemaregistrierung konfiguriert wurde.

```
Events:
  MSKEvent:
    Type: MSK
    Properties:
      StartingPosition: LATEST
      Stream:
```

```
        Ref: MyMskCluster
      Topics:
        - SchemaRegistryTestTopic
      ProvisionedPollerConfig:
        MinimumPollers: 1
      SchemaRegistryConfig:
        SchemaRegistryURI: !Sub arn:${AWS::Partition}:glue:us-
west-2:123456789012:registry/myregistry
        EventRecordFormat: JSON
        SchemaValidationConfigs:
          - Attribute: KEY
          - Attribute: VALUE
```

MSK-Ereignisquelle mit Confluent Schema Registry

Im Folgenden finden Sie ein Beispiel für einen MSK Ereignisquellentyp, der mit einer Confluent Schema Registry konfiguriert wurde.

```
Events:
  MSKEvent:
    Type: MSK
    Properties:
      StartingPosition: LATEST
      Stream:
        Ref: MyMskCluster
      Topics:
        - SchemaRegistryTestTopic
      ProvisionedPollerConfig:
        MinimumPollers: 1
      SchemaRegistryConfig:
        SchemaRegistryURI: https://my-schema-registry.confluent.cloud
        AccessConfigs:
          - Type: BASIC_AUTH
            URI: !Sub arn:${AWS::Partition}:secretsmanager:us-
west-2:123456789012:secret:my-secret
        EventRecordFormat: JSON
        SchemaValidationConfigs:
          - Attribute: KEY
          - Attribute: VALUE
```

S3

Das Objekt, das einen S3 Ereignisquellentyp beschreibt.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
Bucket: String
Events: String | List
Filter: NotificationFilter
```

Eigenschaften

`Bucket`

Name des S3 Buckets. Dieser Bucket muss in derselben Vorlage vorhanden sein.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `BucketName` Eigenschaft einer `AWS::S3::Bucket` Ressource. Dies ist ein Pflichtfeld in SAM. Dieses Feld akzeptiert nur einen Verweis auf den S3-Bucket, der in dieser Vorlage erstellt wurde

`Events`

Das Amazon S3 S3-Bucket-Ereignis, für das die Lambda-Funktion aufgerufen werden soll. Eine Liste der gültigen Werte finden Sie unter Von Amazon S3 unterstützte Ereignistypen.

Typ: Zeichenfolge | Liste

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Event` Eigenschaft des `AWS::S3::Bucket` `LambdaConfiguration` Datentyps übergeben.

`Filter`

Die Filterregeln, die bestimmen, welche Amazon S3 S3-Objekte die Lambda-Funktion aufrufen. Informationen zur Amazon S3 S3-Schlüsselnamenfilterung finden Sie unter Konfiguration

von Amazon S3 S3-Ereignisbenachrichtigungen im Amazon Simple Storage Service-Benutzerhandbuch.

Typ: NotificationFilter

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Filter` Eigenschaft des `AWS::S3::Bucket` `LambdaConfiguration` Datentyps übergeben.

Beispiele

S3-Ereignis

Beispiel für ein S3-Event.

YAML

```
Events:
  S3Event:
    Type: S3
    Properties:
      Bucket:
        Ref: ImagesBucket     # This must be the name of an S3 bucket declared in the
 same template file
      Events: s3:ObjectCreated:*
      Filter:
        S3Key:
          Rules:
          - Name: prefix      # or "suffix"
            Value: value      # The value to search for in the S3 object key names
```

Schedule

Das Objekt, das einen `Schedule` Ereignisquellentyp beschreibt, der Ihre serverlose Funktion als Ziel einer EventBridge Amazon-Regel festlegt, die nach einem Zeitplan ausgelöst wird. Weitere Informationen finden Sie unter Was ist Amazon EventBridge? im EventBridge Amazon-Benutzerhandbuch.

AWS Serverless Application Model (AWS SAM) generiert eine AWS::Events::RuleRessource, wenn dieser Ereignistyp festgelegt ist.

> **Note**
>
> EventBridge bietet jetzt eine neue Planungsfunktion, Amazon EventBridge Scheduler. Amazon EventBridge Scheduler ist ein serverloser Scheduler, mit dem Sie Aufgaben von einem zentralen, verwalteten Service aus erstellen, ausführen und verwalten können. EventBridge Schedulerist hochgradig anpassbar und bietet eine verbesserte Skalierbarkeit im EventBridge Vergleich zu geplanten Regeln mit einem breiteren Spektrum an API-Zieloperationen und. AWS-Services
> Wir empfehlen, dass Sie EventBridge Scheduler verwenden, um Ziele nach einem Zeitplan aufzurufen. Informationen zur Definition dieses Ereignisquellentyps in Ihren AWS SAM Vorlagen finden Sie unterScheduleV2.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
DeadLetterConfig: DeadLetterConfig
Description: String
Enabled: Boolean
Input: String
Name: String
RetryPolicy: RetryPolicy
Schedule: String
State: String
```

Eigenschaften

`DeadLetterConfig`

Konfigurieren Sie die Amazon Simple Queue Service (Amazon SQS) -Warteschlange, über die Ereignisse nach einem fehlgeschlagenen Zielaufruf EventBridge gesendet werden. Der Aufruf kann beispielsweise fehlschlagen, wenn ein Ereignis an eine Lambda-Funktion gesendet wird, die nicht existiert, oder wenn EventBridge nicht genügend Berechtigungen zum Aufrufen der Lambda-Funktion vorhanden sind. Weitere Informationen finden Sie unter Richtlinien zur Wiederholung von Ereignissen und Verwenden von Warteschlangen mit unerlaubten Briefen im Amazon-Benutzerhandbuch. EventBridge

> **Note**
>
> Der AWS::Serverless: :Funktion Ressourcentyp hat einen ähnlichen DatentypDeadLetterQueue, der Fehler behandelt, die nach einem erfolgreichen Aufruf der Lambda-Zielfunktion auftreten. Beispiele für diese Arten von Fehlern sind Lambda-Drosselung oder Fehler, die von der Lambda-Zielfunktion zurückgegeben werden. Weitere Informationen zur *DeadLetterQueue* Funktionseigenschaft finden Sie im Developer Guide unter Warteschlangen mit uneingeschränktem Inhalt.AWS Lambda

Typ: DeadLetterConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `DeadLetterConfig` Eigenschaft des Datentyps. `AWS::Events::Rule Target` Die AWS SAM Version dieser Eigenschaft enthält zusätzliche Untereigenschaften für den Fall, dass Sie die Warteschlange AWS SAM für unzustellbare Briefe erstellen möchten.

`Description`

Eine Beschreibung der Regel.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Description` Eigenschaft einer Ressource übergeben. `AWS::Events::Rule`

`Enabled`

Gibt an, ob die Regel aktiviert ist.

Um die Regel zu deaktivieren, setzen Sie diese Eigenschaft auf`false`.

> **Note**
>
> Geben Sie entweder die `State` Eigenschaft `Enabled` oder an, aber nicht beide.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `State` Eigenschaft einer `AWS::Events::Rule` Ressource. Wenn diese Eigenschaft auf `true` gesetzt ist, gilt sie AWS SAM als erfolgreich`ENABLED`, andernfalls gilt sie als erfolgreich`DISABLED`.

`Input`

Gültiger JSON-Text wurde an das Ziel übergeben. Wenn Sie diese Eigenschaft verwenden, wird nichts aus dem Ereignistext selbst an das Ziel weitergeleitet.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Input` Eigenschaft einer `AWS::Events::Rule Target` Ressource übergeben.

`Name`

Der Name der Regel. Wenn Sie keinen Namen angeben, CloudFormation generiert es eine eindeutige physische ID und verwendet diese ID als Regelnamen.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Name` Eigenschaft einer `AWS::Events::Rule` Ressource übergeben.

`RetryPolicy`

Ein `RetryPolicy`-Objekt, das Informationen zu den Richtlinieneinstellungen für Wiederholungsversuche enthält. Weitere Informationen finden Sie unter Richtlinien zur Wiederholung von Ereignissen und Verwenden von Warteschlangen mit unerlaubten Briefen im Amazon-Benutzerhandbuch. EventBridge

Typ: RetryPolicy

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `RetryPolicy` Eigenschaft des Datentyps übergeben. `AWS::Events::Rule Target`

`Schedule`

Der Planungsausdruck, der bestimmt, wann und wie oft die Regel ausgeführt wird. Weitere Informationen finden Sie unter Planen von Ausdrücken für Regeln.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `ScheduleExpression` Eigenschaft einer `AWS::Events::Rule` Ressource übergeben.

`State`

Der Status der Regel.

Zulässige Werte: `DISABLED | ENABLED`

> **Note**
>
> Geben Sie entweder die `State` Eigenschaft `Enabled` oder an, aber nicht beide.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `State` Eigenschaft einer `AWS::Events::Rule` Ressource übergeben.

Beispiele

CloudWatch Veranstaltung planen

CloudWatch Beispiel für einen Termin planen

YAML

```
CWSchedule:
  Type: Schedule
  Properties:
    Schedule: 'rate(1 minute)'
    Name: TestSchedule
```

```
    Description: test schedule
    Enabled: false
```

DeadLetterConfig

Das Objekt, das zur Angabe der Amazon Simple Queue Service (Amazon SQS) -Warteschlange verwendet wird, über die Ereignisse nach einem fehlgeschlagenen Zielaufruf EventBridge gesendet werden. Der Aufruf kann beispielsweise fehlschlagen, wenn ein Ereignis an eine Lambda-Funktion gesendet wird, die nicht existiert, oder wenn die Berechtigungen zum Aufrufen der Lambda-Funktion nicht ausreichen. Weitere Informationen finden Sie unter Richtlinien zur Wiederholung von Ereignissen und Verwenden von Warteschlangen mit unerlaubten Briefen im Amazon-Benutzerhandbuch. EventBridge

> **Note**
>
> Der AWS::Serverless: :Funktion Ressourcentyp hat einen ähnlichen Datentyp, `DeadLetterQueue` der Fehler behandelt, die nach einem erfolgreichen Aufruf der Lambda-Zielfunktion auftreten. Beispiele für diese Art von Fehler sind Lambda-Drosselung oder Fehler, die von der Lambda-Zielfunktion zurückgegeben werden. Weitere Informationen zur *`DeadLetterQueue`* Funktionseigenschaft finden Sie im Entwicklerhandbuch unter Warteschlangen mit uneingeschränktem Inhalt.AWS Lambda

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer AWS Serverless Application Model (AWS SAM) -Vorlage zu deklarieren.

YAML

```
Arn: String
QueueLogicalId: String
Type: String
```

Eigenschaften

`Arn`

Der Amazon-Ressourcenname (ARN) der Amazon SQS-Warteschlange, der als Ziel für die Warteschlange mit unzustellbaren Briefen angegeben wurde.

> **Note**
>
> Geben Sie entweder die `Type` Eigenschaft oder `Arn` die Eigenschaft an, aber nicht beide.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Arn` Eigenschaft des `AWS::Events::Rule` `DeadLetterConfig` Datentyps übergeben.

`QueueLogicalId`

Der benutzerdefinierte Name der Warteschlange mit unerlaubten Briefen, die sie AWS SAM erstellt, `Type` ist angegeben.

> **Note**
>
> Wenn die `Type` Eigenschaft nicht festgelegt ist, wird diese Eigenschaft ignoriert.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Type`

Der Typ der Warteschlange. Wenn diese Eigenschaft festgelegt ist, wird AWS SAM automatisch eine Warteschlange für unzustellbare Nachrichten erstellt und die erforderliche ressourcenbasierte Richtlinie angehängt, um der Regelressource die Erlaubnis zu erteilen, Ereignisse an die Warteschlange zu senden.

> **Note**
>
> Geben Sie entweder die `Type` Eigenschaft oder die Eigenschaft an, aber nicht beide`Arn`.

Gültige Werte: SQS

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

DeadLetterConfig

DeadLetterConfig

YAML

```
DeadLetterConfig:
  Type: SQS
  QueueLogicalId: MyDLQ
```

ScheduleV2

Das Objekt, das einen `ScheduleV2` Ereignisquellentyp beschreibt, der Ihre serverlose Funktion als Ziel eines Amazon EventBridge Scheduler-Ereignisses festlegt, das nach einem Zeitplan ausgelöst wird. Weitere Informationen finden Sie unter Was ist Amazon EventBridge Scheduler? im EventBridge Scheduler-Benutzerhandbuch.

AWS Serverless Application Model (AWS SAM) generiert eine AWS::Scheduler::ScheduleRessource, wenn dieser Ereignistyp festgelegt ist.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
DeadLetterConfig: DeadLetterConfig
Description: String
```

```
EndDate: String
FlexibleTimeWindow: FlexibleTimeWindow
GroupName: String
Input: String
KmsKeyArn: String
Name: String
OmitName: Boolean
PermissionsBoundary: String
RetryPolicy: RetryPolicy
RoleArn: String
ScheduleExpression: String
ScheduleExpressionTimezone: String
StartDate: String
State: String
```

Eigenschaften

`DeadLetterConfig`

Konfigurieren Sie die Amazon Simple Queue Service (Amazon SQS) -Warteschlange, über die Ereignisse nach einem fehlgeschlagenen Zielaufruf EventBridge gesendet werden. Der Aufruf kann beispielsweise fehlschlagen, wenn ein Ereignis an eine Lambda-Funktion gesendet wird, die nicht existiert, oder wenn EventBridge nicht genügend Berechtigungen zum Aufrufen der Lambda-Funktion vorhanden sind. Weitere Informationen finden Sie im Scheduler-Benutzerhandbuch unter Konfiguration einer Warteschlange mit unerlaubten Briefen für den EventBridge Scheduler. EventBridge

> **Note**
>
> Der AWS::Serverless: :Funktion Ressourcentyp hat einen ähnlichen DatentypDeadLetterQueue, der Fehler behandelt, die nach einem erfolgreichen Aufruf der Lambda-Zielfunktion auftreten. Beispiele für diese Arten von Fehlern sind Lambda-Drosselung oder Fehler, die von der Lambda-Zielfunktion zurückgegeben werden. Weitere Informationen zur *DeadLetterQueue* Funktionseigenschaft finden Sie im Developer Guide unter Warteschlangen mit uneingeschränktem Inhalt.AWS Lambda

Typ: DeadLetterConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `DeadLetterConfig` Eigenschaft des Datentyps. `AWS::Scheduler::Schedule Target` Die AWS SAM Version dieser Eigenschaft enthält zusätzliche Untereigenschaften für den Fall, dass Sie die Warteschlange AWS SAM für unzustellbare Briefe erstellen möchten.

`Description`

Eine Beschreibung des Zeitplans.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Description` Eigenschaft einer `AWS::Scheduler::Schedule` Ressource übergeben.

`EndDate`

Das Datum in UTC, bevor der Zeitplan sein Ziel aufrufen kann. Abhängig vom Wiederholungsausdruck des Zeitplans können Aufrufe an oder vor dem von Ihnen angegebenen EndDate anhalten.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `EndDate` Eigenschaft einer `AWS::Scheduler::Schedule` Ressource übergeben.

`FlexibleTimeWindow`

Ermöglicht die Konfiguration eines Fensters, in dem ein Zeitplan aufgerufen werden kann.

Typ: FlexibleTimeWindow

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `FlexibleTimeWindow` Eigenschaft einer `AWS::Scheduler::Schedule` Ressource übergeben.

`GroupName`

Der Name der Zeitplangruppe, die diesem Zeitplan zugeordnet werden soll. Wenn nicht definiert, wird die Standardgruppe verwendet.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `GroupName` Eigenschaft einer `AWS::Scheduler::Schedule` Ressource übergeben.

`Input`

Gültiger JSON-Text wurde an das Ziel übergeben. Wenn Sie diese Eigenschaft verwenden, wird nichts aus dem Ereignistext selbst an das Ziel weitergeleitet.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Input` Eigenschaft einer `AWS::Scheduler::Schedule Target` Ressource übergeben.

`KmsKeyArn`

Der ARN für einen KMS-Schlüssel, der zur Verschlüsselung von Kundendaten verwendet wird.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `KmsKeyArn` Eigenschaft einer `AWS::Scheduler::Schedule` Ressource übergeben.

`Name`

Der Name des Plans. Wenn Sie keinen Namen angeben, AWS SAM generiert es einen Namen im Format *`Function-Logical-IDEvent-Source-Name`* und verwendet diese ID als Namen des Zeitplans.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Name` Eigenschaft einer `AWS::Scheduler::Schedule` Ressource übergeben.

`OmitName`

AWS SAM Generiert und verwendet standardmäßig einen Zeitplannamen im Format von*<Function-logical-ID><event-source-name>*. Stellen Sie diese Eigenschaft so ein`true`, dass eine eindeutige physische ID CloudFormation generiert und diese stattdessen für den Zeitplannamen verwendet wird.

Typ: Boolesch

Required: No

Standardwert: `false`

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`PermissionsBoundary`

Der ARN der Richtlinie, mit der die Berechtigungsgrenze für die Rolle festgelegt wurde.

> **Note**
>
> Wenn sie definiert `PermissionsBoundary` ist, AWS SAM werden dieselben Grenzen auf die IAM-Zielrolle des Scheduler-Zeitplans angewendet.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `PermissionsBoundary` Eigenschaft einer `AWS::IAM::Role` Ressource übergeben.

`RetryPolicy`

Ein RetryPolicy-Objekt, das Informationen zu den Richtlinieneinstellungen für Wiederholungsversuche enthält.

Typ: RetryPolicy

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `RetryPolicy` Eigenschaft des `AWS::Scheduler::Schedule` `Target` Datentyps übergeben.

`RoleArn`

Der ARN der IAM-Rolle, die der EventBridge Scheduler für das Ziel verwendet, wenn der Zeitplan aufgerufen wird.

Typ: RoleArn

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `RoleArn` Eigenschaft des Datentyps übergeben. `AWS::Scheduler::Schedule Target`

`ScheduleExpression`

Der Scheduling-Ausdruck, der bestimmt, wann und wie oft das Scheduler-Zeitplanereignis ausgeführt wird.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `ScheduleExpression` Eigenschaft einer `AWS::Scheduler::Schedule` Ressource übergeben.

`ScheduleExpressionTimezone`

Die Zeitzone, in der der Planungsausdruck ausgewertet wird.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `ScheduleExpressionTimezone` Eigenschaft einer `AWS::Scheduler::Schedule` Ressource übergeben.

`StartDate`

Das Datum in UTC, nach dem der Zeitplan mit dem Aufrufen eines Ziels beginnen kann. Abhängig vom Wiederholungsausdruck des Zeitplans können Aufrufe an oder nach dem von Ihnen angegebenen StartDate erfolgen.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `StartDate` Eigenschaft einer `AWS::Scheduler::Schedule` Ressource übergeben.

`State`

Der Status des Scheduler-Zeitplans.

Zulässige Werte: `DISABLED | ENABLED`

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `State` Eigenschaft einer `AWS::Scheduler::Schedule` Ressource übergeben.

Beispiele

Einfaches Beispiel für die Definition einer ScheduleV2-Ressource

```
Resources:
  Function:
    Properties:
      ...
      Events:
        ScheduleEvent:
          Type: ScheduleV2
          Properties:
            ScheduleExpression: "rate(1 minute)"
        ComplexScheduleEvent:
          Type: ScheduleV2
          Properties:
            ScheduleExpression: rate(1 minute)
            FlexibleTimeWindow:
              Mode: FLEXIBLE
              MaximumWindowInMinutes: 5
            StartDate: '2022-12-28T12:00:00.000Z'
            EndDate: '2023-01-28T12:00:00.000Z'
            ScheduleExpressionTimezone: UTC
            RetryPolicy:
              MaximumRetryAttempts: 5
```

```
        MaximumEventAgeInSeconds: 300
      DeadLetterConfig:
        Type: SQS
```

> **Note**
>
> Die generierte physische ID von ScheduleV2 enthält keinen Stacknamen.

SelfManagedKafka

Das Objekt, das einen `SelfManagedKafka` Ereignisquellentyp beschreibt. Weitere Informationen finden Sie unter Verwendung AWS Lambda mit selbstverwaltetem Apache Kafka im AWS Lambda Entwicklerhandbuch.

AWS Serverless Application Model (AWS SAM) generiert eine AWS::Lambda::EventSourceMappingRessource, wenn dieser Ereignistyp gesetzt ist.

Um Schema Registry verwenden zu können, müssen Sie spezifische IAM-Rollenberechtigungen für Ihre Funktion definieren. Ein Beispiel für die erforderliche Konfiguration finden Sie unter Vollständige Einrichtung mit IAM-Rollen.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer AWS SAM Vorlage zu deklarieren.

YAML

```
BatchSize: Integer
BisectBatchOnFunctionError: Boolean
ConsumerGroupId: String
DestinationConfig: DestinationConfig
Enabled: Boolean
FilterCriteria: FilterCriteria
KafkaBootstrapServers: List
FunctionResponseTypes: List
KmsKeyArn: String
LoggingConfig: LoggingConfig
MaximumRecordAgeInSeconds: Integer
MaximumRetryAttempts: Integer
MetricsConfig: MetricsConfig
ProvisionedPollerConfig: ProvisionedPollerConfig
```

```
  SchemaRegistryConfig: SchemaRegistryConfig
  SourceAccessConfigurations: SourceAccessConfigurations
  StartingPosition: String
  StartingPositionTimestamp: Double
  Topics: List
```

Eigenschaften

`BatchSize`

Die maximale Anzahl von Datensätzen in jedem Batch, die Lambda aus Ihrem Stream abruft und an Ihre Funktion sendet.

Typ: Ganzzahl

Required: No

Standard: 100

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `BatchSize` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

Minimum: `1`

Maximum: `10000`

`BisectBatchOnFunctionError`

Wenn die Funktion einen Fehler zurückgibt, teilen Sie den Stapel in zwei Teile auf und versuchen Sie es erneut.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `BisectBatchOnFunctionError` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`ConsumerGroupId`

Eine Zeichenfolge, die konfiguriert, wie Ereignisse aus Kafka-Themen gelesen werden.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `SelfManagedKafkaEventSourceConfig` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`DestinationConfig`

Ein Konfigurationsobjekt, das das Ziel eines Ereignisses angibt, nachdem Lambda es verarbeitet hat.

Verwenden Sie diese Eigenschaft, um das Ziel fehlgeschlagener Aufrufe aus der selbstverwalteten Kafka-Ereignisquelle anzugeben.

Typ: DestinationConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `DestinationConfig` Eigenschaft einer Ressource übergeben. `AWS::Lambda::EventSourceMapping`

`Enabled`

Deaktiviert den Ereignis-Quellzuweisung zum Anhalten und Aufrufen von Abfragen.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Enabled` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`FilterCriteria`

Ein Objekt, das die Kriterien definiert, anhand derer bestimmt wird, ob Lambda ein Ereignis verarbeiten soll. Weitere Informationen finden Sie unter AWS Lambda Ereignisfilterung im AWS Lambda Entwicklerhandbuch.

Typ: FilterCriteria

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `FilterCriteria` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`KafkaBootstrapServers`

Die Liste der Bootstrap-Server für Ihre Kafka-Broker. Geben Sie zum Beispiel den Port an `broker.example.com:xxxx`

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`FunctionResponseTypes`

Eine Liste der Antworttypen, die derzeit auf die Ereignisquellenzuordnung angewendet werden. Weitere Informationen finden Sie unter Melden von Batch-Elementen im AWS Lambda -Leitfaden für Entwickler.

Gültige Werte: `ReportBatchItemFailures`

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `FunctionResponseTypes` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`KmsKeyArn`

Der Amazon-Ressourcenname (ARN) des Schlüssels zur Verschlüsselung von Informationen im Zusammenhang mit diesem Ereignis.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `KmsKeyArn` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`LoggingConfig`

Die Protokollierungskonfiguration für Ihre Ereignisquelle.

Typ: LoggingConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `LoggingConfig` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`MaximumRecordAgeInSeconds`

Das maximale Alter eines Datensatzes, den Lambda zur Verarbeitung an eine Funktion sendet.

Typ: Ganzzahl

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `MaximumRecordAgeInSeconds` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`MetricsConfig`

Die Metrikkonfiguration für Ihre Ereignisquelle.

Typ: MetricsConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `MetricsConfig` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`MaximumRetryAttempts`

Die maximale Anzahl der Wiederholungen, wenn die Funktion einen Fehler zurückgibt.

Typ: Ganzzahl

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `MaximumRetryAttempts` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`ProvisionedPollerConfig`

Konfiguration zur Erhöhung der Anzahl von Pollern, die zur Berechnung von Ereignisquellenzuordnungen verwendet werden. Diese Konfiguration ermöglicht mindestens einen Poller und ein Maximum von 2000 Pollern. Ein Beispiel finden Sie unter ProvisionedPollerConfig Beispiel

Typ: ProvisionedPollerConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `ProvisionedPollerConfig` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`SchemaRegistryConfig`

Konfiguration für die Verwendung einer Schemaregistrierung mit der selbstverwalteten Kafka-Ereignisquelle.

> **Note**
>
> Diese Funktion muss konfiguriert `ProvisionedPollerConfig` werden.

Typ: SchemaRegistryConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `SelfManagedKafkaEventSourceConfig` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`SourceAccessConfigurations`

Ein Array des Authentifizierungsprotokolls, der VPC-Komponenten oder des virtuellen Hosts zum Sichern und Definieren Ihrer Ereignisquelle.

Gültige Werte: `BASIC_AUTH | CLIENT_CERTIFICATE_TLS_AUTH | SASL_SCRAM_256_AUTH | SASL_SCRAM_512_AUTH | SERVER_ROOT_CA_CERTIFICATE`

Typ: Liste von SourceAccessConfiguration

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist Teil der `SelfManagedKafkaEventSourceConfig` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource.

`StartingPosition`

Die Position im Stream, an der mit dem Lesen begonnen wird.

- `AT_TIMESTAMP`— Geben Sie einen Zeitpunkt an, ab dem mit dem Lesen von Datensätzen begonnen werden soll.
- `LATEST`— Nur neue Datensätze lesen.
- `TRIM_HORIZON`— Verarbeitet alle verfügbaren Datensätze.

Zulässige Werte: `AT_TIMESTAMP` | `LATEST` | `TRIM_HORIZON`

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `StartingPosition` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`StartingPositionTimestamp`

Die Zeit, ab der mit dem Lesen begonnen werden soll, in Unix-Zeitsekunden. Definiert`StartingPositionTimestamp`, wann als angegeben `StartingPosition` ist`AT_TIMESTAMP`.

Type: Double

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `StartingPositionTimestamp` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`Topics`

Der Name des Kafka-Themas.

Typ: Liste

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Topics` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

Beispiele

Vollständige Einrichtung mit IAM-Rollen

Das folgende Beispiel zeigt ein vollständiges Setup einschließlich der erforderlichen IAM-Rollenkonfiguration für die Verwendung von Schema Registry:

```
Parameters:
  PreCreatedSubnetOne:
    Type: String
  PreCreatedSubnetTwo:
    Type: String

Resources:
  MyLambdaExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Action: [sts:AssumeRole]
          Effect: Allow
          Principal:
            Service: [lambda.amazonaws.com]
      Policies:
      - PolicyName: KafkaAuthPolicy
        PolicyDocument:
          Statement:
          - Action: [secretsmanager:GetSecretValue, kms:Decrypt]
            Effect: "Allow"
            Resource: ['arn:aws:secretsmanager:us-
west-2:123456789012:secret:kafkaSecret-******',
                       'arn:aws:kms:us-west-2:123456789012:key/keyId']
      - PolicyName: ENIPolicy
        PolicyDocument:
          Statement:
          - Action: [ec2:CreateNetworkInterface,
              ec2:DescribeNetworkInterfaces, ec2:DescribeVpcs,
 ec2:DeleteNetworkInterface,
              ec2:DescribeSubnets, ec2:DescribeSecurityGroups]
            Effect: Allow
            Resource: '*'
      - PolicyName: SchemaRegistryPolicy
        PolicyDocument:
```

```
          Statement:
          - Action: [glue:GetRegistry]
            Effect: Allow
            Resource: 'arn:aws:glue:{region}:{account-id}:registry/{registry-name}'
      - PolicyName: SchemaVersionsPolicy
        PolicyDocument:
          Statement:
          - Action: [glue:GetSchemaVersions]
            Effect: Allow
            Resource: '*'
      ManagedPolicyArns:
      - !Sub arn:${AWS::Partition}:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole
      Tags:
      - {Value: SAM, Key: lambda:createdBy}

  MyKafkaProcessor:
    Type: AWS::Serverless::Function
    Properties:
      Runtime: nodejs18.x
      Handler: index.handler
      CodeUri: ${codeuri}
      Role:
        Fn::GetAtt: [MyLambdaExecutionRole, Arn]
      Events:
        SelfManagedKafkaEvent:
          Type: SelfManagedKafka
          Properties:
            KafkaBootstrapServers:
              - my-kafka-broker-1:9092
              - my-kafka-broker-2:9092
            Topics:
              - SchemaRegistryTestTopic
            StartingPosition: LATEST
            SourceAccessConfigurations:
              - Type: VPC_SUBNET
                URI: subnet:subnet-12345678
              - Type: VPC_SECURITY_GROUP
                URI: security_group:sg-12345678
              - Type: BASIC_AUTH
                URI: !Sub arn:${AWS::Partition}:secretsmanager:us-
west-2:123456789012:secret:my-path/my-secret-name-1a2b3c
            ProvisionedPollerConfig:
              MinimumPollers: 1
```

```
            SchemaRegistryConfig:
              AccessConfigs:
              - Type: BASIC_AUTH
                URI: !Sub arn:${AWS::Partition}:secretsmanager:us-
west-2:123456789012:secret:my-path/my-secret-name-1a2b3c
              SchemaValidationConfigs:
              - Attribute: KEY
              EventRecordFormat: JSON
              SchemaRegistryURI: !Sub arn:${AWS::Partition}:glue:us-
west-2:123456789012:registry/myregistry
```

ProvisionedPollerConfig Beispiel

```
ProvisionedPollerConfig:
  MinimumPollers: 1
  MaximumPollers: 200
```

Selbstverwaltete Kafka-Eventquelle

Im Folgenden finden Sie ein Beispiel für einen `SelfManagedKafka` Ereignisquellentyp.

YAML

```
Events:
  SelfManagedKafkaEvent:
    Type: SelfManagedKafka
    Properties:
      BatchSize: 1000
      Enabled: true
      KafkaBootstrapServers:
        - abc.xyz.com:xxxx
      SourceAccessConfigurations:
        -  Type: BASIC_AUTH
           URI: arn:aws:secretsmanager:us-west-2:123456789012:secret:my-path/my-secret-
name-1a2b3c
      Topics:
        - MyKafkaTopic
```

Selbstverwaltete Kafka-Ereignisquelle mit AWS Glue Schemaregistrierung

Im Folgenden finden Sie ein Beispiel für einen mit AWS Glue Schema Registry konfigurierten `SelfManagedKafka` Ereignisquellentyp.

```
Events:
  SelfManagedKafkaEvent:
    Type: SelfManagedKafka
    Properties:
      KafkaBootstrapServers:
        - abc.xyz.com:9092
      Topics:
        - SchemaRegistryTestTopic
      StartingPosition: LATEST
      ProvisionedPollerConfig:
        MinimumPollers: 1
      SchemaRegistryConfig:
        SchemaRegistryURI: !Sub arn:${AWS::Partition}:glue:us-
west-2:123456789012:registry/myregistry
        EventRecordFormat: JSON
        SchemaValidationConfigs:
          - Attribute: KEY
          - Attribute: VALUE
      SourceAccessConfigurations:
        - Type: VPC_SUBNET
          URI: subnet:subnet-12345678
        - Type: VPC_SECURITY_GROUP
          URI: security_group:sg-12345678
```

Selbstverwaltete Kafka-Ereignisquelle mit Confluent Schema Registry

Im Folgenden finden Sie ein Beispiel für einen `SelfManagedKafka` Ereignisquellentyp, der mit Confluent Schema Registry konfiguriert wurde.

```
Events:
  SelfManagedKafkaEvent:
    Type: SelfManagedKafka
    Properties:
      KafkaBootstrapServers:
        - abc.xyz.com:9092
      Topics:
        - SchemaRegistryTestTopic
      StartingPosition: LATEST
      ProvisionedPollerConfig:
        MinimumPollers: 1
      SchemaRegistryConfig:
        SchemaRegistryURI: https://my-schema-registry.confluent.cloud
        AccessConfigs:
```

```
          - Type: BASIC_AUTH
            URI: !Sub arn:${AWS::Partition}:secretsmanager:us-
west-2:123456789012:secret:my-secret
        EventRecordFormat: JSON
        SchemaValidationConfigs:
          - Attribute: KEY
          - Attribute: VALUE
      SourceAccessConfigurations:
        - Type: VPC_SUBNET
          URI: subnet:subnet-12345678
        - Type: VPC_SECURITY_GROUP
          URI: security_group:sg-12345678
        - Type: BASIC_AUTH
          URI: !Sub arn:${AWS::Partition}:secretsmanager:us-
west-2:123456789012:secret:kafka-secret
```

SNS

Das Objekt, das einen SNS Ereignisquellentyp beschreibt.

SAM generiert eine AWS::SNS::SubscriptionRessource, wenn dieser Ereignistyp festgelegt ist

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
  FilterPolicy: SnsFilterPolicy
  FilterPolicyScope: String
  RedrivePolicy: Json
  Region: String
  SqsSubscription: Boolean | SqsSubscriptionObject
  Topic: String
```

Eigenschaften

`FilterPolicy`

Das Filterrichtlinien-JSON, das dem Abonnement zugeordnet ist. Weitere Informationen finden Sie GetSubscriptionAttributesin der Amazon Simple Notification Service API-Referenz.

Typ: SnsFilterPolicy

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `FilterPolicy` Eigenschaft einer `AWS::SNS::Subscription` Ressource übergeben.

`FilterPolicyScope`

Mit diesem Attribut können Sie den Filterbereich mithilfe eines der folgenden Zeichenfolgenwerttypen auswählen:

- `MessageAttributes`— Der Filter wird auf die Nachrichtenattribute angewendet.
- `MessageBody`— Der Filter wird auf den Nachrichtentext angewendet.

Typ: Zeichenfolge

Required: No

Standardwert: `MessageAttributes`

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `FilterPolicyScope` Eigenschaft einer `AWS::SNS::Subscription` Ressource übergeben.

`RedrivePolicy`

Wenn angegeben, werden unzustellbare Nachrichten an die angegebene Amazon SQS-Warteschlange für unzustellbare Nachrichten gesendet. Nachrichten, die aufgrund von Clientfehlern (z. B. wenn der abonnierte Endpunkt nicht erreichbar ist) oder Serverfehlern (z. B. wenn der Service, der den abonnierten Endpunkt betreibt, nicht verfügbar ist) nicht zugestellt werden können, werden in der Warteschlange für unzustellbare Nachrichten zur weiteren Analyse oder erneuten Verarbeitung gespeichert.

Weitere Informationen zur Redrive-Richtlinie und zu Warteschlangen für unzustellbare Briefe finden Sie unter Amazon SQS Dead-Letter-Warteschlangen im Amazon Simple Queue Service Developer Guide.

Type: Json

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die Eigenschaft einer Ressource übergeben. `RedrivePolicy` `AWS::SNS::Subscription`

`Region`

Für regionsübergreifende Abonnements, die Region, in der das Thema gespeichert ist.

Wenn keine Region angegeben ist, wird standardmäßig die Region des Anrufers CloudFormation verwendet.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Region` Eigenschaft einer `AWS::SNS::Subscription` Ressource übergeben.

`SqsSubscription`

Setzen Sie diese Eigenschaft auf true oder geben Sie `SqsSubscriptionObject` an, dass SNS-Themenbenachrichtigungen in einer SQS-Warteschlange gebündelt werden sollen. Wenn Sie diese Eigenschaft auf festlegen, `true` wird eine neue SQS-Warteschlange erstellt, wohingegen die Angabe von a eine vorhandene SQS-Warteschlange `SqsSubscriptionObject` verwendet.

Typ: Boolean | SqsSubscriptionObject

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Topic`

Der ARN des zu abonnierenden Themas.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `TopicArn` Eigenschaft einer `AWS::SNS::Subscription` Ressource übergeben.

Beispiele

Beispiel für eine SNS-Ereignisquelle

Beispiel für eine SNS-Ereignisquelle

YAML

```
Events:
  SNSEvent:
    Type: SNS
    Properties:
      Topic: arn:aws:sns:us-east-1:123456789012:my_topic
      SqsSubscription: true
      FilterPolicy:
        store:
          - example_corp
        price_usd:
          - numeric:
              - ">="
              - 100
```

SqsSubscriptionObject

Geben Sie eine bestehende SQS-Warteschlangenoption für das SNS-Ereignis an

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
  BatchSize: String
  Enabled: Boolean
  QueueArn: String
  QueuePolicyLogicalId: String
  QueueUrl: String
```

Eigenschaften

`BatchSize`

Die maximale Anzahl von Elementen, die in einem einzigen Batch für die SQS-Warteschlange abgerufen werden können.

Typ: Zeichenfolge

Required: No

Standard: 10

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Enabled`

Deaktiviert die Zuordnung der SQS-Ereignisquellen, um die Abfrage und den Aufruf zu unterbrechen.

Typ: Boolesch

Required: No

Standard: Wahr

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`QueueArn`

Geben Sie einen vorhandenen SQS-Warteschlangen-ARN an.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`QueuePolicyLogicalId`

Geben Sie einen benutzerdefinierten LogicalID-Namen für die Ressource ein. AWS::SQS::QueuePolicy

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein Äquivalent. CloudFormation

`QueueUrl`

Geben Sie die Warteschlangen-URL an, die der `QueueArn` Eigenschaft zugeordnet ist.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

Existierendes Ereignis SQS für SNS

Beispiel für das Hinzufügen einer vorhandenen SQS-Warteschlange zum Abonnieren eines SNS-Themas.

YAML

```
QueuePolicyLogicalId: CustomQueuePolicyLogicalId
QueueArn:
  Fn::GetAtt: MyCustomQueue.Arn
QueueUrl:
  Ref: MyCustomQueue
BatchSize: 5
```

SQS

Das Objekt, das einen SQS Ereignisquellentyp beschreibt. Weitere Informationen finden Sie unter Using AWS Lambda with Amazon SQS im AWS Lambda Developer Guide.

SAM generiert AWS::Lambda::EventSourceMappingRessourcen, wenn dieser Ereignistyp festgelegt ist

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
  BatchSize: Integer
  Enabled: Boolean
  FilterCriteria: FilterCriteria
  FunctionResponseTypes: List
```

```
  KmsKeyArn: String
  MaximumBatchingWindowInSeconds: Integer
  MetricsConfig: MetricsConfig
  ProvisionedPollerConfig: ProvisionedPollerConfig
  Queue: String
  ScalingConfig: ScalingConfig
```

Eigenschaften

`BatchSize`

Gibt die maximale Anzahl der Elemente an, die in einem einzigen Stapel zurückzugeben werden.

Typ: Ganzzahl

Required: No

Standard: 10

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `BatchSize` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

Minimum: 1

Maximum: `10000`

`Enabled`

Deaktiviert den Ereignis-Quellzuweisung zum Anhalten und Aufrufen von Abfragen.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Enabled` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`FilterCriteria`

Ein Objekt, das die Kriterien definiert, anhand derer bestimmt wird, ob Lambda ein Ereignis verarbeiten soll. Weitere Informationen finden Sie unter AWS Lambda Ereignisfilterung im AWS Lambda Entwicklerhandbuch.

Typ: FilterCriteria

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `FilterCriteria` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`FunctionResponseTypes`

Eine Liste der Antworttypen, die derzeit auf die Ereignisquellenzuordnung angewendet werden. Weitere Informationen finden Sie im AWS Lambda Entwicklerhandbuch unter Melden von Fehlern bei Batch-Elementen.

Gültige Werte: `ReportBatchItemFailures`

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `FunctionResponseTypes` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`KmsKeyArn`

Der Amazon-Ressourcenname (ARN) des Schlüssels zur Verschlüsselung von Informationen zu diesem Ereignis.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `KmsKeyArn` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`MaximumBatchingWindowInSeconds`

Die maximale Zeit in Sekunden für das Sammeln von Datensätzen vor dem Aufrufen der Funktion.

Typ: Ganzzahl

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `MaximumBatchingWindowInSeconds` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`MetricsConfig`

Eine optionale Konfiguration, um erweiterte Metriken für Zuordnungen von Ereignisquellen zu erhalten, die jede Phase der Verarbeitung erfassen. Ein Beispiel finden Sie unter MetricsConfig Ereignis.

Typ: MetricsConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `MetricsConfig` Eigenschaft einer Ressource übergeben. `AWS::Lambda::EventSourceMapping`

`ProvisionedPollerConfig`

Konfiguration zur Erhöhung der Anzahl von Pollern, die zur Berechnung von Ereignisquellenzuordnungen verwendet werden. Diese Konfiguration ermöglicht ein Minimum von 2 Pollern und ein Maximum von 2000 Pollern. Ein Beispiel finden Sie unter. ProvisionedPollerConfig Beispiel

Typ: ProvisionedPollerConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `ProvisionedPollerConfig` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`Queue`

Der ARN der Warteschlange.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `EventSourceArn` Eigenschaft einer `AWS::Lambda::EventSourceMapping` Ressource übergeben.

`ScalingConfig`

Skalierung der Konfiguration von SQS-Pollern zur Steuerung der Aufrufrate und zur Festlegung der maximalen Anzahl gleichzeitiger Aufrufe.

Typ: ScalingConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die ScalingConfig Eigenschaft einer Ressource übergeben. AWS::Lambda::EventSourceMapping

Beispiele

MetricsConfig Ereignis

Im Folgenden finden Sie ein Beispiel für eine Ressource, die die MetricsConfig Eigenschaft verwendet, um jede Phase der Verarbeitung ihrer Ereignisquellenzuordnungen zu erfassen.

```
Resources:
  FilteredEventsFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: s3://sam-demo-bucket/metricsConfig.zip
      Handler: index.handler
      Runtime: nodejs16.x
      Events:
        KinesisStream:
          Type: Kinesis
          Properties:
            Stream: !GetAtt KinesisStream.Arn
            StartingPosition: LATEST
            MetricsConfig:
              Metrics:
              - EventCount
```

Einfaches SQS-Ereignis

```
Events:
  SQSEvent:
    Type: SQS
    Properties:
      Queue: arn:aws:sqs:us-west-2:012345678901:my-queue
      BatchSize: 10
      Enabled: false
      FilterCriteria:
        Filters:
```

```
          - Pattern: '{"key": ["val1", "val2"]}'
```

Konfigurieren Sie die teilweise Batch-Berichterstattung für Ihre SQS-Warteschlange

```
Events:
  SQSEvent:
    Type: SQS
    Properties:
      Enabled: true
      FunctionResponseTypes:
        - ReportBatchItemFailures
      Queue: !GetAtt MySqsQueue.Arn
      BatchSize: 10
```

Lambda-Funktion mit einem SQS-Ereignis, für das die Skalierung konfiguriert ist

```
MyFunction:
  Type: AWS::Serverless::Function
  Properties:
    ...
    Events:
      MySQSEvent:
        Type: SQS
        Properties:
          ...
          ScalingConfig:
            MaximumConcurrency: 10
```

ProvisionedPollerConfig Beispiel

```
MyFunction:
  Type: AWS::Serverless::Function
  Properties:
    Handler: index.handler
    Runtime: nodejs18.x
    Timeout: 30
    Events:
      SQSEvent:
        Type: SQS
        Properties:
          Queue: !GetAtt MyQueue.Arn
          BatchSize: 10
```

```
          Enabled: True
          ProvisionedPollerConfig:
            MaximumPollers: 300
            MinimumPollers: 10
```

FunctionCode

Das Bereitstellungspaket für eine Lambda-Funktion.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer AWS Serverless Application Model (AWS SAM) -Vorlage zu deklarieren.

YAML

```
Bucket: String
Key: String
StorageMode: String
Version: String
```

Eigenschaften

`Bucket`

Ein Amazon S3-Bucket in derselben AWS Region wie Ihre Funktion.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `S3Bucket` Eigenschaft des `AWS::Lambda::Function Code` Datentyps übergeben.

`Key`

Der Amazon S3-Schlüssel des Bereitstellungspakets.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `S3Key` Eigenschaft des `AWS::Lambda::Function Code` Datentyps übergeben.

`StorageMode`

Steuert, wie Lambda das Bereitstellungspaket speichert.

- `COPY`— Lambda lädt eine Kopie Ihres Bereitstellungspakets in den Speicher hoch. Lambda-managed Sie können das Amazon S3-Objekt löschen, nachdem Lambda die Funktion erstellt hat.
- `REFERENCE`— Lambda verweist auf das Bereitstellungspaket aus Ihrem Amazon S3-Bucket und speichert keine Kopie. Das Objekt muss an Ort und Stelle bleiben, und Lambda muss während der gesamten Lebensdauer der Funktion darauf zugreifen können.

Um es zu verwenden`REFERENCE`, aktivieren Sie die Versionierung in Ihrem Amazon S3-Bucket und gewähren Sie dem Lambda-Serviceprinzipal Zugriff auf das Objekt. Weitere Informationen finden Sie im AWS Lambda Entwicklerhandbuch unter Self-managed Amazon S3-Codespeicher.

Typ: Zeichenfolge

Gültige Werte: `COPY` | `REFERENCE`

Required: No

Standardwert: `COPY`

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `S3ObjectStorageMode` Eigenschaft des `AWS::Lambda::Function Code` Datentyps übergeben.

`Version`

Für versionierte Objekte, die Version des zu verwendenden Bereitstellungspaketobjekts.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `S3ObjectVersion` Eigenschaft des `AWS::Lambda::Function Code` Datentyps übergeben.

Beispiele

FunctionCode

`CodeUri`: Beispiel für einen Funktionscode

YAML

```
CodeUri:
  Bucket: sam-s3-demo-bucket-name
  Key: mykey-name
  Version: 121212
```

Self-managed Amazon S3-Codespeicher

Im Folgenden wird `CodeUri` auf das Bereitstellungspaket aus Ihrem eigenen Amazon S3-Bucket verwiesen, anstatt es in den Lambda-managed Speicher zu kopieren.

YAML

```
CodeUri:
  Bucket: amzn-s3-demo-bucket-name
  Key: mykey-name
  Version: 121212
  StorageMode: REFERENCE
```

FunctionUrlConfig

Erzeugt eine AWS Lambda Funktions-URL mit den angegebenen Konfigurationsparametern. Eine Lambda-Funktions-URL ist ein HTTPS-Endpunkt, mit dem Sie Ihre Funktion aufrufen können.

Standardmäßig verwendet die von Ihnen erstellte Funktions-URL die `$LATEST` Version Ihrer Lambda-Funktion. Wenn Sie `AutoPublishAlias` für Ihre Lambda-Funktion eine angeben, stellt der Endpunkt eine Verbindung zum angegebenen Funktionsalias her.

Weitere Informationen finden Sie unter Lambda-Funktion URLs im AWS Lambda Entwicklerhandbuch.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
AuthType: String
Cors: Cors
```

```
InvokeMode: String
```

Eigenschaften

`AuthType`

Die Art der Autorisierung für Ihre Funktions-URL. Um AWS Identity and Access Management (IAM) zur Autorisierung von Anfragen zu verwenden, stellen Sie auf ein. `AWS_IAM` Stellen Sie für Open Access auf ein. `NONE`

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `AuthType` Eigenschaft einer `AWS::Lambda::Url` Ressource übergeben.

`Cors`

Die Cross-Origin Resource Sharing (CORS)-Einstellungen für Ihre Funktions-URL.

Type: Cors

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Cors` Eigenschaft einer `AWS::Lambda::Url` Ressource übergeben.

`InvokeMode`

Der Modus, in dem Ihre Funktions-URL aufgerufen wird. Damit Ihre Funktion die Antwort nach Abschluss des Aufrufs zurückgibt, setzen Sie die Einstellung auf. `BUFFERED` Damit Ihre Funktion die Antwort streamt, setzen Sie den Wert auf`RESPONSE_STREAM`. Der Standardwert ist `BUFFERED`.

Zulässige Werte: `BUFFERED` oder `RESPONSE_STREAM`.

Typ: Zeichenfolge

Required: No

AWS CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `InvokeMode`Eigenschaft einer `AWS::Lambda::Url` Ressource übergeben.

Beispiele

URL der Funktion

Das folgende Beispiel erstellt eine Lambda-Funktion mit einer Funktions-URL. Die Funktions-URL verwendet die IAM-Autorisierung.

YAML

```
HelloWorldFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: hello_world/
    Handler: index.handler
    Runtime: nodejs20.x
    FunctionUrlConfig:
      AuthType: AWS_IAM
      InvokeMode: RESPONSE_STREAM

Outputs:
  MyFunctionUrlEndpoint:
      Description: "My Lambda Function URL Endpoint"
      Value:
        Fn::GetAtt: HelloWorldFunctionUrl.FunctionUrl
```

CapacityProviderConfig

Konfiguriert den Kapazitätsanbieter, an den veröffentlichte Versionen der Funktion angehängt werden. Dadurch kann die Funktion auf kundeneigenen EC2 Instanzen ausgeführt werden, die von Lambda verwaltet werden.

> **Note**
>
> Diese Konfiguration bestimmt den Rechentyp einer Funktion und muss bei der ersten Funktionsbereitstellung angegeben werden. Sie kann nicht hinzugefügt oder entfernt werden, nachdem die Funktionsressource erstellt wurde.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
Arn: String
ExecutionEnvironmentMemoryGiBPerVCpu: Float
PerExecutionEnvironmentMaxConcurrency: Integer
```

Eigenschaften

`Arn`

Der ARN des Kapazitätsanbieters, der für diese Funktion verwendet werden soll.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft gibt es nur bei SAM.

`ExecutionEnvironmentMemoryGiBPerVCpu`

Das Verhältnis von Arbeitsspeicher (in GiB) zu vCPU für jede Ausführungsumgebung.

> **Note**
>
> Das Speicherverhältnis pro CPU darf den Gesamtspeicher der Funktion von 2048 MB nicht überschreiten. Die unterstützten memory-to-CPU Verhältnisse sind 2 GB, 4 GB oder 8 GB pro CPU.

Typ: Float

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `ExecutionEnvironmentMemoryGiBPerVCpu` Eigenschaft einer `AWS::Lambda::Function` Ressource übergeben.

`PerExecutionEnvironmentMaxConcurrency`

Die maximale Anzahl gleichzeitiger Ausführungen pro Ausführungsumgebung (Sandbox).

Typ: Ganzzahl

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `PerExecutionEnvironmentMaxConcurrency` Eigenschaft einer Ressource übergeben. `AWS::Lambda::Function`

Beispiele

Kapazitätsanbieter-Konfiguration

Das folgende Beispiel zeigt eine Kapazitätsanbieterkonfiguration, die auf eine Kapazitätsanbieterressource verweist.

```
CapacityProviderConfig:
  Arn: !GetAtt MyCapacityProvider.Arn
  ExecutionEnvironmentMemoryGiBPerVCpu: 4.0
  PerExecutionEnvironmentMaxConcurrency: 100
```

FunctionScalingConfig

Konfiguriert das Skalierungsverhalten für Lambda-Funktionsversionen und steuert die Anzahl der Ausführungsumgebungen (Sandboxen), die erstellt werden können. Diese Konfiguration gilt sowohl für $LATEST.PUBLISHED als auch für numerische Funktionsversionen.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer AWS Serverless Application Model (AWS SAM) -Vorlage zu deklarieren.

YAML

```
MinExecutionEnvironments: Integer
MaxExecutionEnvironments: Integer
```

Eigenschaften

`MinExecutionEnvironments`

Die Mindestanzahl von Ausführungsumgebungen, die für die Funktionsversion verwaltet werden müssen.

Typ: Ganzzahl

Required: No

Standardwert: `3`

Minimum: `0`

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `MinExecutionEnvironments` Eigenschaft einer `AWS::Lambda::Function` Ressource übergeben.

`MaxExecutionEnvironments`

Die maximale Anzahl von Ausführungsumgebungen, die für die Funktionsversion erstellt werden können.

Typ: Ganzzahl

Required: No

Standardwert: `3`

Minimum: `0`

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `MaxExecutionEnvironments` Eigenschaft einer `AWS::Lambda::Function` Ressource übergeben.

Beispiele

Konfiguration der Funktionsskalierung

Das folgende Beispiel zeigt eine Konfiguration zur Funktionsskalierung mit minimalen und maximalen Ausführungsumgebungen.

```
FunctionScalingConfig:
  MinExecutionEnvironments: 5
  MaxExecutionEnvironments: 100
```

AWS::Serverless::GraphQLApi

Verwenden Sie den `AWS::Serverless::GraphQLApi` Ressourcentyp AWS Serverless Application Model (AWS SAM), um eine AWS AppSync GraphQL API für Ihre serverlose Anwendung zu erstellen und zu konfigurieren.

Weitere Informationen dazu finden Sie AWS AppSync unter Was ist AWS AppSync? im AWS AppSync Entwicklerhandbuch.

Syntax

YAML

```
LogicalId:
  Type: AWS::Serverless::GraphQLApi
  Properties:
    ApiKeys: ApiKeys
    Auth: Auth
    Cache: AWS::AppSync::ApiCache
    DataSources: DataSource
    DomainName: AWS::AppSync::DomainName
    Functions: Function
    Logging: LogConfig
    Name: String
    Resolvers: Resolver
    SchemaInline: String
    SchemaUri: String
    Tags:
    - Tag
    XrayEnabled: Boolean
```

Eigenschaften

`ApiKeys`

Erstellen Sie einen eindeutigen Schlüssel, der verwendet werden kann, um GraphQL Operationen auszuführen, für die ein API-Schlüssel erforderlich ist.

Typ: ApiKeys

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Auth`

Konfigurieren Sie die Authentifizierung für Ihre GraphQL API.

Typ: Auth

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Cache`

Die Eingabe einer `CreateApiCache` Operation.

Typ: AWS::AppSync::ApiCache

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die AWS::AppSync::ApiCacheRessource übergeben.

`DataSources`

Erstellen Sie Datenquellen für Funktionen, mit denen AWS AppSync Sie eine Verbindung herstellen möchten. AWS SAM unterstützt Amazon DynamoDB und AWS Lambda Datenquellen.

Typ: DataSource

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`DomainName`

Benutzerdefinierter Domainname für Ihre GraphQL API.

Typ: AWS::AppSync::DomainName

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die AWS::AppSync::DomainNameRessource übergeben. AWS SAM generiert die AWS::AppSync::DomainNameApiAssociationRessource automatisch.

`Functions`

Konfigurieren Sie Funktionen GraphQL APIs , um bestimmte Operationen auszuführen.

Typ: Funktion

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Logging`

Konfiguriert die CloudWatch Amazon-Protokollierung für Ihre GraphQL API.

Wenn Sie diese Eigenschaft nicht angeben, AWS SAM werden die folgenden Werte generiert `CloudWatchLogsRoleArn` und festgelegt:

- `ExcludeVerboseContent: true`
- `FieldLogLevel: ALL`

Um die Protokollierung zu deaktivieren, geben Sie Folgendes an:

```
Logging: false
```

Typ: LogConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `LogConfig` Eigenschaft einer `AWS::AppSync::GraphQLApi` Ressource übergeben.

`LogicalId`

Der eindeutige Name Ihrer GraphQL API.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Name` Eigenschaft einer `AWS::AppSync::GraphQLApi` Ressource übergeben.

`Name`

Der Name Ihrer GraphQL API. Geben Sie diese Eigenschaft an, um den `LogicalId` Wert zu überschreiben.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Name` Eigenschaft einer `AWS::AppSync::GraphQLApi` Ressource übergeben.

`Resolvers`

Konfigurieren Sie Resolver für die Felder Ihrer GraphQL API. AWS SAM unterstützt JavaScriptPipeline-Resolver.

Typ: Resolver

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig AWS SAM und hat kein CloudFormation Äquivalent.

`SchemaInline`

Die Textdarstellung eines GraphQL Schemas im SDL Format.

Typ: Zeichenfolge

Erforderlich: Bedingt. Sie müssen `SchemaInline` oder angeben`SchemaUri`.

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Definition` Eigenschaft einer `AWS::AppSync::GraphQLSchema` Ressource übergeben.

`SchemaUri`

Die Amazon Simple Storage Service (Amazon S3) -Bucket-URI oder der Pfad zu einem lokalen Ordner des Schemas.

Wenn Sie einen Pfad zu einem lokalen Ordner angeben, CloudFormation muss die Datei vor der Bereitstellung zuerst auf Amazon S3 hochgeladen werden. Sie können den verwenden AWS SAMCLI, um diesen Vorgang zu vereinfachen. Weitere Informationen finden Sie unter So werden lokale Dateien bei der Bereitstellung AWS SAM hochgeladen.

Typ: Zeichenfolge

Erforderlich: Bedingt. Sie müssen `SchemaInline` oder angeben`SchemaUri`.

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `DefinitionS3Location` Eigenschaft einer `AWS::AppSync::GraphQLSchema` Ressource übergeben.

`Tags`

Tags (Schlüssel-Wert-Paare) für diese GraphQL API. Verwenden Sie Tags, um Ressourcen zu identifizieren und zu kategorisieren.

Typ: Liste von Tag

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Tag` Eigenschaft einer `AWS::AppSync::GraphQLApi` Ressource übergeben.

`XrayEnabled`

Geben Sie an, ob AWS X-Ray Tracing für diese Ressource verwendet werden soll.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `XrayEnabled` Eigenschaft einer `AWS::AppSync::GraphQLApi` Ressource übergeben.

Rückgabewerte

Eine Liste der Rückgabewerte finden Sie AWS::Serverless::GraphQLApiim CloudFormation Benutzerhandbuch.

Beispiele

GraphQL APImit DynamoDB-Datenquelle

In diesem Beispiel erstellen wir eine GraphQL API, die eine DynamoDB-Tabelle als Datenquelle verwendet.

schema.graphql

```
schema {
  query: Query
  mutation: Mutation
}
```

```
type Query {
  getPost(id: String!): Post
}

type Mutation {
  addPost(author: String!, title: String!, content: String!): Post!
}

type Post {
  id: String!
  author: String
  title: String
  content: String
  ups: Int!
  downs: Int!
  version: Int!
}
```

Vorlage.yaml

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  DynamoDBPostsTable:
    Type: AWS::Serverless::SimpleTable

  MyGraphQLAPI:
    Type: AWS::Serverless::GraphQLApi
    Properties:
      SchemaUri: ./sam_graphql_api/schema.graphql
      Auth:
        Type: AWS_IAM
      DataSources:
        DynamoDb:
          PostsDataSource:
            TableName: !Ref DynamoDBPostsTable
            TableArn: !GetAtt DynamoDBPostsTable.Arn
      Functions:
        preprocessPostItem:
          Runtime:
            Name: APPSYNC_JS
            Version: 1.0.0
```

```
          DataSource: NONE
          CodeUri: ./sam_graphql_api/preprocessPostItem.js
        createPostItem:
          Runtime:
            Name: APPSYNC_JS
            Version: "1.0.0"
          DataSource: PostsDataSource
          CodeUri: ./sam_graphql_api/createPostItem.js
        getPostFromTable:
          Runtime:
            Name: APPSYNC_JS
            Version: "1.0.0"
          DataSource: PostsDataSource
          CodeUri: ./sam_graphql_api/getPostFromTable.js
      Resolvers:
        Mutation:
          addPost:
            Runtime:
              Name: APPSYNC_JS
              Version: "1.0.0"
            Pipeline:
            - preprocessPostItem
            - createPostItem
        Query:
          getPost:
            CodeUri: ./sam_graphql_api/getPost.js
            Runtime:
              Name: APPSYNC_JS
              Version: "1.0.0"
            Pipeline:
            - getPostFromTable
```

createPostItem.js

```
import { util } from "@aws-appsync/utils";

export function request(ctx) {
  const { key, values } = ctx.prev.result;
  return {
    operation: "PutItem",
    key: util.dynamodb.toMapValues(key),
    attributeValues: util.dynamodb.toMapValues(values),
  };
```

```
}

export function response(ctx) {
  return ctx.result;
}
```

getPostFromTable.js

```
import { util } from "@aws-appsync/utils";

export function request(ctx) {
  return dynamoDBGetItemRequest({ id: ctx.args.id });
}

export function response(ctx) {
  return ctx.result;
}

/**
 * A helper function to get a DynamoDB item
 */
function dynamoDBGetItemRequest(key) {
  return {
    operation: "GetItem",
    key: util.dynamodb.toMapValues(key),
  };
}
```

preprocessPostItem.js

```
import { util } from "@aws-appsync/utils";

export function request(ctx) {
  const id = util.autoId();
  const { ...values } = ctx.args;
  values.ups = 1;
  values.downs = 0;
  values.version = 1;
  return { payload: { key: { id }, values: values } };
}

export function response(ctx) {
  return ctx.result;
```

```
}
```

Hier ist unser Resolver-Code:

getPost.js

```
export function request(ctx) {
  return {};
}

export function response(ctx) {
  return ctx.prev.result;
}
```

GraphQLAPI mit einer Lambda-Funktion als Datenquelle

In diesem Beispiel erstellen wir eine GraphQL API, die eine Lambda-Funktion als Datenquelle verwendet.

template.yaml

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyLambdaFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.handler
      Runtime: nodejs20.x
      CodeUri: ./lambda

  MyGraphQLAPI:
    Type: AWS::Serverless::GraphQLApi
    Properties:
      Name: MyApi
      SchemaUri: ./gql/schema.gql
      Auth:
        Type: API_KEY
      ApiKeys:
        MyApiKey:
          Description: my api key
      DataSources:
```

```
        Lambda:
          MyLambdaDataSource:
            FunctionArn: !GetAtt MyLambdaFunction.Arn
      Functions:
        lambdaInvoker:
          Runtime:
            Name: APPSYNC_JS
            Version: 1.0.0
          DataSource: MyLambdaDataSource
          CodeUri: ./gql/invoker.js
      Resolvers:
        Mutation:
          addPost:
            Runtime:
              Name: APPSYNC_JS
              Version: 1.0.0
            Pipeline:
            - lambdaInvoker
        Query:
          getPost:
            Runtime:
              Name: APPSYNC_JS
              Version: 1.0.0
            Pipeline:
            - lambdaInvoker

Outputs:
  MyGraphQLAPI:
    Description: AppSync API
    Value: !GetAtt MyGraphQLAPI.GraphQLUrl
  MyGraphQLAPIMyApiKey:
    Description: API Key for authentication
    Value: !GetAtt MyGraphQLAPIMyApiKey.ApiKey
```

schema.graphql

```
schema {
  query: Query
  mutation: Mutation
}
type Query {
  getPost(id: ID!): Post
}
```

```
type Mutation {
  addPost(id: ID!, author: String!, title: String, content: String): Post!
}
type Post {
  id: ID!
  author: String!
  title: String
  content: String
  ups: Int
  downs: Int
}
```

Hier sind unsere Funktionen:

lambda/index.js

```
exports.handler = async (event) => {
  console.log("Received event {}", JSON.stringify(event, 3));

  const posts = {
    1: {
      id: "1",
      title: "First book",
      author: "Author1",
      content: "Book 1 has this content",
      ups: "100",
      downs: "10",
    },
  };

  console.log("Got an Invoke Request.");
  let result;
  switch (event.field) {
    case "getPost":
      return posts[event.arguments.id];
    case "addPost":
      // return the arguments back
      return event.arguments;
    default:
      throw new Error("Unknown field, unable to resolve " + event.field);
  }
};
```

invoker.js

```
import { util } from "@aws-appsync/utils";

export function request(ctx) {
  const { source, args } = ctx;
  return {
    operation: "Invoke",
    payload: { field: ctx.info.fieldName, arguments: args, source },
  };
}

export function response(ctx) {
  return ctx.result;
}
```

ApiKeys

Erstellen Sie einen eindeutigen Schlüssel, mit dem GraphQL Operationen ausgeführt werden können, für die ein API-Schlüssel erforderlich ist.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
LogicalId:
  ApiKeyId: String
  Description: String
  ExpiresOn: Double
```

Eigenschaften

`ApiKeyId`

Der eindeutige Name Ihres API-Schlüssels. Geben Sie an, dass der `LogicalId` Wert überschrieben werden soll.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `ApiKeyId` Eigenschaft einer `AWS::AppSync::ApiKey` Ressource übergeben.

`Description`

Beschreibung Ihres API-Schlüssels.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Description` Eigenschaft einer `AWS::AppSync::ApiKey` Ressource übergeben.

`ExpiresOn`

Die Zeit, nach der der API-Schlüssel abläuft. Das Datum wird als Sekunden seit der Epoche dargestellt, abgerundet auf die nächste Stunde.

Type: Double

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Expires` Eigenschaft einer `AWS::AppSync::ApiKey` Ressource übergeben.

`LogicalId`

Der eindeutige Name Ihres API-Schlüssels.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `ApiKeyId` Eigenschaft einer `AWS::AppSync::ApiKey` Ressource übergeben.

Auth

Konfigurieren Sie die Autorisierung für Ihre GraphQL API.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
Additional:
- AuthProvider
LambdaAuthorizer: LambdaAuthorizerConfig
OpenIDConnect: OpenIDConnectConfig
Type: String
UserPool: UserPoolConfig
```

Eigenschaften

`Additional`

Eine Liste zusätzlicher Autorisierungstypen für Ihre GraphQL API.

Typ: Liste von AuthProvider

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`LambdaAuthorizer`

Geben Sie die optionale Autorisierungskonfiguration für Ihren Lambda-Funktionsautorisierer an. Sie können diese optionale Eigenschaft konfigurieren, wenn sie als angegeben `Type` ist. `AWS_LAMBDA`

Typ: LambdaAuthorizerConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `LambdaAuthorizerConfig` Eigenschaft einer `AWS::AppSync::GraphQLApi` Ressource übergeben.

`OpenIDConnect`

Geben Sie die optionale Autorisierungskonfiguration für Ihren OpenID Connect konformen Dienst an. Sie können diese optionale Eigenschaft konfigurieren, wenn sie als angegeben `Type` ist`OPENID_CONNECT`.

Typ: Open IDConnect Config

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `OpenIDConnectConfig` Eigenschaft einer `AWS::AppSync::GraphQLApi` Ressource übergeben.

`Type`

Der Standardautorisierungstyp zwischen Anwendungen und Ihrer AWS AppSync GraphQL API.

Eine Liste und Beschreibung der zulässigen Werte finden Sie unter Autorisierung und Authentifizierung im AWS AppSync Entwicklerhandbuch.

Wenn Sie einen Lambda-Autorisierer (`AWS_LAMBDA`) angeben, AWS SAM wird eine AWS Identity and Access Management (IAM-) Richtlinie erstellt, um Berechtigungen zwischen Ihrer GraphQL API und der Lambda-Funktion bereitzustellen.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `AuthenticationType` Eigenschaft einer Ressource übergeben. `AWS::AppSync::GraphQLApi`

`UserPool`

Geben Sie die optionale Autorisierungskonfiguration für die Verwendung von Amazon Cognito Cognito-Benutzerpools an. Sie können diese optionale Eigenschaft konfigurieren, wenn sie als `AMAZON_COGNITO_USER_POOLS` angegeben `Type` ist.

Typ: UserPoolConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `UserPoolConfig` Eigenschaft einer `AWS::AppSync::GraphQLApi` Ressource übergeben.

Beispiele

Konfigurieren Sie einen Standard- und einen zusätzlichen Autorisierungstyp

In diesem Beispiel konfigurieren wir zunächst einen Lambda-Autorisierungstyp als Standardautorisierungstyp für unsere GraphQL API.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyGraphQLAPI:
    Type: AWS::Serverless::GraphQLApi
    Properties:
      Auth:
        Type: AWS_LAMBDA
        LambdaAuthorizer:
          AuthorizerUri: !GetAtt Authorizer1.Arn
          AuthorizerResultTtlInSeconds: 10
          IdentityValidationExpression: hello
```

Als Nächstes konfigurieren wir zusätzliche Autorisierungstypen für unsere GraphQL API, indem wir unserer AWS SAM Vorlage Folgendes hinzufügen:

```
        Additional:
        - Type: AWS_IAM
        - Type: API_KEY
        - Type: OPENID_CONNECT
          OpenIDConnect:
            AuthTTL: 10
            ClientId: myId
            IatTTL: 10
            Issuer: prod
```

Dies führt zu der folgenden AWS SAM Vorlage:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyGraphQLAPI:
    Type: AWS::Serverless::GraphQLApi
    Properties:
      Auth:
        Type: AWS_LAMBDA
        LambdaAuthorizer:
          AuthorizerUri: !GetAtt Authorizer1.Arn
          AuthorizerResultTtlInSeconds: 10
          IdentityValidationExpression: hello
```

```
        Additional:
        - Type: AWS_IAM
        - Type: API_KEY
        - Type: OPENID_CONNECT
          OpenIDConnect:
            AuthTTL: 10
            ClientId: myId
            IatTTL: 10
            Issuer: prod
```

AuthProvider

Optionale Autorisierungskonfiguration für Ihre zusätzlichen GraphQL API-Autorisierungstypen.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
LambdaAuthorizer: LambdaAuthorizerConfig
OpenIDConnect: OpenIDConnectConfig
Type: String
UserPool: UserPoolConfig
```

Eigenschaften

`LambdaAuthorizer`

Geben Sie die optionale Autorisierungskonfiguration für Ihren AWS Lambda Funktionsautorisierer an. Sie können diese optionale Eigenschaft konfigurieren, wenn sie als `AWS_LAMBDA` angegeben `Type` ist.

Typ: LambdaAuthorizerConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `LambdaAuthorizerConfig` Eigenschaft eines `AWS::AppSync::GraphQLApi` `AdditionalAuthenticationProvider` Objekts übergeben.

`OpenIDConnect`

Geben Sie die optionale Autorisierungskonfiguration für Ihren OpenID Connect konformen Dienst an. Sie können diese optionale Eigenschaft konfigurieren, wenn sie als angegeben `Type` ist`OPENID_CONNECT`.

Typ: Open IDConnect Config

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `OpenIDConnectConfig` Eigenschaft eines `AWS::AppSync::GraphQLApi` `AdditionalAuthenticationProvider` Objekts übergeben.

`Type`

Der Standardautorisierungstyp zwischen Anwendungen und Ihrer AWS AppSync GraphQL API.

Eine Liste und Beschreibung der zulässigen Werte finden Sie unter Autorisierung und Authentifizierung im AWS AppSync Entwicklerhandbuch.

Wenn Sie einen Lambda-Autorisierer (`AWS_LAMBDA`) angeben, AWS SAM wird eine AWS Identity and Access Management (IAM-) Richtlinie erstellt, um Berechtigungen zwischen Ihrer GraphQL API und der Lambda-Funktion bereitzustellen.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `AuthenticationType` Eigenschaft eines Objekts übergeben. `AWS::AppSync::GraphQLApi` `AdditionalAuthenticationProvider`

`UserPool`

Geben Sie die optionale Autorisierungskonfiguration für die Verwendung von Amazon Cognito Cognito-Benutzerpools an. Sie können diese optionale Eigenschaft konfigurieren, wenn sie als `AMAZON_COGNITO_USER_POOLS` angegeben `Type` ist.

Typ: `UserPoolConfig`

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `UserPoolConfig` Eigenschaft eines `AWS::AppSync::GraphQLApi` `AdditionalAuthenticationProvider` Objekts übergeben.

DataSource

Konfigurieren Sie eine Datenquelle, mit der Ihr GraphQL API-Resolver eine Verbindung herstellen kann. Sie können Vorlagen AWS Serverless Application Model (AWS SAM) verwenden, um Verbindungen zu den folgenden Datenquellen zu konfigurieren:

- Amazon DynamoDB
- AWS Lambda

Weitere Informationen zu Datenquellen finden Sie unter Anhängen einer Datenquelle im AWS AppSync Entwicklerhandbuch.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
DynamoDb: DynamoDb
Lambda: Lambda
```

Eigenschaften

`DynamoDb`

Konfigurieren Sie eine DynamoDB-Tabelle als Datenquelle für Ihren GraphQL API-Resolver.

Typ: DynamoDb

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein Äquivalent. CloudFormation

`Lambda`

Konfigurieren Sie eine Lambda-Funktion als Datenquelle für Ihren GraphQL API-Resolver.

Type: Lambda

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

DynamoDb

Konfigurieren Sie eine Amazon DynamoDB-Tabelle als Datenquelle für Ihren GraphQL API-Resolver.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
LogicalId:
  DeltaSync: DeltaSyncConfig
  Description: String
  Name: String
  Permissions: List
  Region: String
  ServiceRoleArn: String
  TableArn: String
  TableName: String
  UseCallerCredentials: Boolean
  Versioned: Boolean
```

Eigenschaften

`DeltaSync`

Beschreibt eine Delta Sync-Konfiguration.

Typ: DeltaSyncConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `DeltaSyncConfig` Eigenschaft eines `AWS::AppSync::DataSource DynamoDBConfig` Objekts übergeben.

`Description`

Die Beschreibung Ihrer Datenquelle.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Description` Eigenschaft einer `AWS::AppSync::DataSource` Ressource übergeben.

`LogicalId`

Der eindeutige Name Ihrer Datenquelle.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Name` Eigenschaft einer `AWS::AppSync::DataSource` Ressource übergeben.

`Name`

Der Name Ihrer Datenquelle. Geben Sie diese Eigenschaft an, um den `LogicalId` Wert zu überschreiben.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Name` Eigenschaft einer `AWS::AppSync::DataSource` Ressource übergeben.

`Permissions`

Erteilen Sie Berechtigungen für Ihre Datenquelle mithilfe vonAWS SAM Anschlüsse. Sie können jeden der folgenden Werte in einer Liste angeben:

- `Read`— Erlauben Sie Ihrem Resolver, Ihre Datenquelle zu lesen.
- `Write`— Erlauben Sie Ihrem Resolver, in Ihre Datenquelle zu schreiben.

AWS SAM verwendet eine `AWS::Serverless::Connector` Ressource, die bei der Bereitstellung transformiert wird, um Ihre Berechtigungen bereitzustellen. Weitere Informationen zu generierten Ressourcen finden Sie unterCloudFormation Ressourcen, die generiert werden, wenn Sie angeben AWS::Serverless::Connector.

> **Note**
>
> Sie können `Permissions` oder `ServiceRoleArn` angeben, aber nicht beides. Wenn keines der beiden angegeben ist, AWS SAM werden Standardwerte von `Read` und generiert`Write`. Um den Zugriff auf Ihre Datenquelle zu widerrufen, entfernen Sie das DynamoDB-Objekt aus Ihrer AWS SAM Vorlage.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent. Sie ähnelt der `Permissions` Eigenschaft einer `AWS::Serverless::Connector` Ressource.

`Region`

Die AWS-Region Ihrer DynamoDB-Tabelle. Wenn Sie es nicht angeben, AWS SAM verwendet. `AWS::Region`

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `AwsRegion` Eigenschaft eines `AWS::AppSync::DataSource DynamoDBConfig` Objekts übergeben.

`ServiceRoleArn`

Die AWS Identity and Access Management (IAM-) Dienstrolle ARN für die Datenquelle. Das System übernimmt diese Rolle beim Zugriff auf die Datenquelle.

Sie können `Permissions` oder `ServiceRoleArn` angeben, aber nicht beides.

Typ: Zeichenfolge

Erforderlich: Nein. Wenn nicht angegeben, wird der Standardwert für AWS SAM angewendet`Permissions`.

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `ServiceRoleArn` Eigenschaft einer `AWS::AppSync::DataSource` Ressource übergeben.

`TableArn`

Der ARN für die DynamoDB-Tabelle.

Typ: Zeichenfolge

Erforderlich: Bedingt. Wenn Sie es nicht angeben`ServiceRoleArn`, `TableArn` ist es erforderlich.

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`TableName`

Der Name der Tabelle.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `TableName` Eigenschaft eines `AWS::AppSync::DataSource DynamoDBConfig` Objekts übergeben.

`UseCallerCredentials`

Legt fest`true`, dass IAM mit dieser Datenquelle verwendet wird.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `UseCallerCredentials` Eigenschaft eines `AWS::AppSync::DataSource DynamoDBConfig` Objekts übergeben.

`Versioned`

Auf Konflikterkennung, Konfliktlösung und Synchronisation mit dieser Datenquelle eingestellt. `true`

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Versioned` Eigenschaft eines `AWS::AppSync::DataSource DynamoDBConfig` Objekts übergeben.

Lambda

Konfigurieren Sie eine AWS Lambda Funktion als Datenquelle für Ihren GraphQL API-Resolver.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
LogicalId:
  Description: String
  FunctionArn: String
  Name: String
  ServiceRoleArn: String
```

Eigenschaften

`Description`

Die Beschreibung Ihrer Datenquelle.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Description` Eigenschaft einer `AWS::AppSync::DataSource` Ressource übergeben.

`FunctionArn`

Der ARN für die Lambda-Funktion

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `LambdaFunctionArn` Eigenschaft eines `AWS::AppSync::DataSource LambdaConfig` Objekts übergeben.

`LogicalId`

Der eindeutige Name Ihrer Datenquelle.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Name` Eigenschaft einer `AWS::AppSync::DataSource` Ressource übergeben.

`Name`

Der Name Ihrer Datenquelle. Geben Sie diese Eigenschaft an, um den `LogicalId` Wert zu überschreiben.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Name` Eigenschaft einer `AWS::AppSync::DataSource` Ressource übergeben.

`ServiceRoleArn`

Die AWS Identity and Access Management (IAM-) Dienstrolle ARN für die Datenquelle. Das System übernimmt diese Rolle beim Zugriff auf die Datenquelle.

> **Note**
>
> Um den Zugriff auf Ihre Datenquelle zu widerrufen, entfernen Sie das Lambda-Objekt aus Ihrer AWS SAM Vorlage.

Typ: Zeichenfolge

Erforderlich: Nein. Wenn nicht angegeben, AWS SAM werden `Write` Berechtigungen mithilfe von bereitgestelltAWS SAM Anschlüsse.

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `ServiceRoleArn` Eigenschaft einer `AWS::AppSync::DataSource` Ressource übergeben.

Funktion

Konfigurieren Sie Funktionen GraphQL APIs , um bestimmte Operationen auszuführen.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
LogicalId:
  CodeUri: String
  DataSource: String
  Description: String
  Id: String
  InlineCode: String
  MaxBatchSize: Integer
  Name: String
  Runtime: Runtime
  Sync: SyncConfig
```

Eigenschaften

`CodeUri`

Die Amazon Simple Storage Service (Amazon S3) -URI oder der Pfad zum lokalen Ordner des Funktionscodes.

Wenn Sie einen Pfad zu einem lokalen Ordner angeben, CloudFormation muss die Datei vor der Bereitstellung zuerst auf Amazon S3 hochgeladen werden. Sie können den verwenden AWS SAMCLI, um diesen Vorgang zu vereinfachen. Weitere Informationen finden Sie unter So werden lokale Dateien bei der Bereitstellung AWS SAM hochgeladen.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `CodeS3Location` Eigenschaft einer `AWS::AppSync::FunctionConfiguration` Ressource übergeben.

`DataSource`

Der Name der Datenquelle, an die diese Funktion angehängt wird.

- Um auf eine Datenquelle innerhalb der `AWS::Serverless::GraphQLApi` Ressource zu verweisen, geben Sie deren logische ID an.

- Um auf eine Datenquelle außerhalb der `AWS::Serverless::GraphQLApi` Ressource zu verweisen, geben Sie ihr `Name` Attribut mithilfe der `Fn::GetAtt` systeminternen Funktion an. Beispiel, `!GetAtt MyLambdaDataSource.Name`.
- Um auf eine Datenquelle aus einem anderen Stapel zu verweisen, verwenden Sie. `Fn::ImportValue`

Wenn eine Variante von angegeben `[NONE | None | none]` ist, AWS SAM wird ein `None` Wert für das `AWS::AppSync::DataSource` `Type` Objekt generiert.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `DataSourceName` Eigenschaft einer `AWS::AppSync::FunctionConfiguration` Ressource übergeben.

`Description`

Die Beschreibung Ihrer Funktion.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Description` Eigenschaft einer `AWS::AppSync::FunctionConfiguration` Ressource übergeben.

`Id`

Die Funktions-ID für eine Funktion, die sich außerhalb der `AWS::Serverless::GraphQLApi` Ressource befindet.

- Um auf eine Funktion innerhalb derselben AWS SAM Vorlage zu verweisen, verwenden Sie die `Fn::GetAtt` systeminterne Funktion. Zum Beispiel `Id: !GetAtt createPostItemFunc.FunctionId`.
- Um auf eine Funktion aus einem anderen Stack zu verweisen, verwenden Sie. `Fn::ImportValue`

Bei der Verwendung `Id` sind alle anderen Eigenschaften nicht zulässig. AWS SAM übergibt automatisch die Funktions-ID Ihrer referenzierten Funktion.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`InlineCode`

Der Funktionscode, der die Anforderungs- und Antwortfunktionen enthält.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Code` Eigenschaft einer `AWS::AppSync::FunctionConfiguration` Ressource übergeben.

`LogicalId`

Der eindeutige Name Ihrer Funktion.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Name` Eigenschaft einer `AWS::AppSync::FunctionConfiguration` Ressource übergeben.

`MaxBatchSize`

Die maximale Anzahl der Resolver-Anforderungs-Eingaben, die an eine einzelne AWS Lambda -Funktion in einem `BatchInvoke`-Vorgang gesendet werden.

Typ: Ganzzahl

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die MaxBatchSizeEigenschaft einer `AWS::AppSync::FunctionConfiguration` Ressource übergeben.

`Name`

Der Name der Funktion. Geben Sie an, dass der `LogicalId` Wert überschrieben werden soll.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Name` Eigenschaft einer `AWS::AppSync::FunctionConfiguration` Ressource übergeben.

`Runtime`

Beschreibt eine Laufzeit, die von einem AWS AppSync Pipeline-Resolver oder einer AWS AppSync Pipeline-Funktion verwendet wird. Gibt den Namen und die Version der zu verwendenden Laufzeit an.

Typ: Runtime

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent. Sie ähnelt der `Runtime` Eigenschaft einer `AWS::AppSync::FunctionConfiguration` Ressource.

`Sync`

Beschreibt eine Sync-Konfiguration für eine Funktion.

Gibt an, welche Konflikterkennungs- und Lösungsstrategie verwendet werden soll, wenn die Funktion aufgerufen wird.

Typ: SyncConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `SyncConfig` Eigenschaft einer `AWS::AppSync::FunctionConfiguration` Ressource übergeben.

Laufzeit

Die Laufzeit Ihres Pipeline-Resolvers oder Ihrer Pipeline-Funktion. Gibt den Namen und die Version an, die verwendet werden sollen.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
Name: String
Version: String
```

Eigenschaften

`Name`

Der Name der zu verwendenden Laufzeit. Der einzige zulässige Wert ist derzeit `APPSYNC_JS`.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Name` Eigenschaft eines `AWS::AppSync::FunctionConfiguration AppSyncRuntime` Objekts übergeben.

`Version`

Die Version der zu verwendenden Runtime. Die einzige zulässige Version ist derzeit `1.0.0`.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `RuntimeVersion` Eigenschaft eines `AWS::AppSync::FunctionConfiguration AppSyncRuntime` Objekts übergeben.

Resolver

Konfigurieren Sie Resolver für die Felder Ihrer GraphQL API. AWS Serverless Application Model (AWS SAM) unterstützt JavaScript Pipeline-Resolver.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
OperationType:
```

```
  LogicalId:
    Caching: CachingConfig
    CodeUri: String
    FieldName: String
    InlineCode: String
    MaxBatchSize: Integer
    Pipeline: List
    Runtime: Runtime
    Sync: SyncConfig
```

Eigenschaften

`Caching`

Die Caching-Konfiguration für den Resolver, für den das Caching aktiviert ist.

Typ: CachingConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `CachingConfig` Eigenschaft einer Ressource übergeben. `AWS::AppSync::Resolver`

`CodeUri`

Die Amazon Simple Storage Service (Amazon S3) -URI oder der Pfad zu einem lokalen Ordner des Resolver-Funktionscodes.

Wenn Sie einen Pfad zu einem lokalen Ordner angeben, CloudFormation muss die Datei vor der Bereitstellung zuerst auf Amazon S3 hochgeladen werden. Sie können den verwenden AWS SAMCLI, um diesen Vorgang zu vereinfachen. Weitere Informationen finden Sie unter So werden lokale Dateien bei der Bereitstellung AWS SAM hochgeladen.

Falls keines `CodeUri` von beiden angegeben AWS SAM wird, wird generiert, `InlineCode` dass die Anfrage an die erste Pipeline-Funktion weitergeleitet wird und die Antwort von der letzten Pipeline-Funktion empfangen wird. `InlineCode`

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `CodeS3Location` Eigenschaft einer `AWS::AppSync::Resolver` Ressource übergeben.

`FieldName`

Der Name Ihres Resolvers. Geben Sie diese Eigenschaft an, um den `LogicalId` Wert zu überschreiben.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `FieldName` Eigenschaft einer `AWS::AppSync::Resolver` Ressource übergeben.

`InlineCode`

Der Resolver-Code, der die Anforderungs- und Antwortfunktionen enthält.

Wenn keiner `CodeUri` von beiden angegeben AWS SAM wird, wird generiert, `InlineCode` dass die Anfrage an die erste Pipeline-Funktion umgeleitet wird und die Antwort von der letzten Pipeline-Funktion empfangen wird. `InlineCode`

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Code` Eigenschaft einer `AWS::AppSync::Resolver` Ressource übergeben.

`LogicalId`

Der eindeutige Name für Ihren Resolver. In einem GraphQL Schema sollte Ihr Resolvername mit dem Feldnamen übereinstimmen, für den er verwendet wird. Verwenden Sie denselben Feldnamen für`LogicalId`.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`MaxBatchSize`

Die maximale Anzahl der Resolver-Anforderungs-Eingaben, die an eine einzelne AWS Lambda -Funktion in einem `BatchInvoke`-Vorgang gesendet werden.

Typ: Ganzzahl

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `MaxBatchSize` Eigenschaft einer `AWS::AppSync::Resolver` Ressource übergeben.

`OperationType`

Der GraphQL Vorgangstyp, der Ihrem Resolver zugeordnet ist. Beispiel: `Query`, `Mutation` oder `Subscription`. Sie können mehrere Resolver in einem `LogicalId` einzigen ineinander verschachteln. `OperationType`

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `TypeName` Eigenschaft einer `AWS::AppSync::Resolver` Ressource übergeben.

`Pipeline`

Funktionen im Zusammenhang mit dem Pipeline-Resolver. Geben Sie Funktionen anhand der logischen ID in einer Liste an.

Typ: Liste

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent. Sie ähnelt der `PipelineConfig` Eigenschaft einer `AWS::AppSync::Resolver` Ressource.

`Runtime`

Die Laufzeit Ihres Pipeline-Resolvers oder Ihrer Pipeline-Funktion. Gibt den Namen und die Version an, die verwendet werden sollen.

Typ: Runtime

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent. Sie ähnelt der `Runtime` Eigenschaft einer `AWS::AppSync::Resolver` Ressource.

Sync

Beschreibt eine Sync-Konfiguration für einen Resolver.

Gibt an, welche Konflikterkennungs- und Lösungsstrategie verwendet werden soll, wenn der Resolver aufgerufen wird.

Typ: SyncConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `SyncConfig` Eigenschaft einer `AWS::AppSync::Resolver` Ressource übergeben.

Beispiele

Verwenden Sie den AWS SAM generierten Resolver-Funktionscode und speichern Sie Felder als Variablen

Hier ist das GraphQL Schema für unser Beispiel:

```
schema {
  query: Query
  mutation: Mutation
}

type Query {
  getPost(id: ID!): Post
}

type Mutation {
  addPost(author: String!, title: String!, content: String!): Post!
}

type Post {
  id: ID!
  author: String
  title: String
  content: String
}
```

Hier ist ein Ausschnitt aus unserer AWS SAM Vorlage:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyGraphQLApi:
    Type: AWS::Serverless::GraphQLApi
    Properties:
      ...
      Functions:
        preprocessPostItem:
          ...
        createPostItem:
          ...
      Resolvers:
        Mutation:
          addPost:
            Runtime:
              Name: APPSYNC_JS
              Version: 1.0.0
            Pipeline:
            - preprocessPostItem
            - createPostItem
```

In unserer AWS SAM Vorlage geben wir oder nicht an`CodeUri`. `InlineCode` Generiert bei der Bereitstellung AWS SAM automatisch den folgenden Inline-Code für unseren Resolver:

```
export function request(ctx) {
  return {};
}

export function response(ctx) {
  return ctx.prev.result;
}
```

Dieser Standard-Resolver-Code leitet die Anfrage an die erste Pipeline-Funktion weiter und empfängt die Antwort von der letzten Pipeline-Funktion.

In unserer ersten Pipeline-Funktion können wir das bereitgestellte `args` Feld verwenden, um das Anforderungsobjekt zu analysieren und unsere Variablen zu erstellen. Wir können diese Variablen dann in unserer Funktion verwenden. Hier ist ein Beispiel für unsere `preprocessPostItem` Funktion:

```
import { util } from "@aws-appsync/utils";

export function request(ctx) {
  const author = ctx.args.author;
  const title = ctx.args.title;
  const content = ctx.args.content;

  // Use variables to process data

}

export function response(ctx) {
  return ctx.result;
}
```

Laufzeit

Die Laufzeit Ihres Pipeline-Resolvers oder Ihrer Pipeline-Funktion. Gibt den Namen und die Version an, die verwendet werden sollen.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
Name: String
Version: String
```

Eigenschaften

`Name`

Der Name der zu verwendenden Laufzeit. Der einzige zulässige Wert ist derzeit `APPSYNC_JS`.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Name` Eigenschaft eines `AWS::AppSync::Resolver AppSyncRuntime` Objekts übergeben.

`Version`

Die Version der zu verwendenden Runtime. Die einzige zulässige Version ist derzeit `1.0.0`.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `RuntimeVersion` Eigenschaft eines `AWS::AppSync::Resolver AppSyncRuntime` Objekts übergeben.

AWS::Serverless::HttpApi

Erstellt eine Amazon API Gateway Gateway-HTTP-API, mit der Sie RESTful APIs mit geringerer Latenz und geringeren Kosten als REST erstellen können APIs. Weitere Informationen finden Sie unter Working with HTTP APIs im API Gateway Developer Guide.

Wir empfehlen, CloudFormation Hooks oder IAM-Richtlinien zu verwenden, um zu überprüfen, ob an API-Gateway-Ressourcen Autorisatoren angehängt sind, um den Zugriff darauf zu kontrollieren.

Weitere Informationen zur Verwendung von CloudFormation Hooks finden Sie unter Hooks registrieren im CloudFormation CLI-Benutzerhandbuch und im apigw-enforce-authorizer GitHub Repository.

Weitere Informationen zur Verwendung von IAM-Richtlinien finden Sie unter Erfordern, dass API-Routen autorisiert sind im API Gateway Developer Guide.

> **Note**
>
> Bei der Bereitstellung auf werden AWS CloudFormation Ihre AWS SAM Ressourcen in Ressourcen umgewandelt CloudFormation . AWS SAM Weitere Informationen finden Sie unter Generiert CloudFormation Ressourcen für AWS SAM.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
Type: AWS::Serverless::HttpApi
Properties:
  AccessLogSettings: AccessLogSettings
  Auth: HttpApiAuth
  CorsConfiguration: String | HttpApiCorsConfiguration
  DefaultRouteSettings: RouteSettings
  DefinitionBody: JSON
  DefinitionUri: String | HttpApiDefinition
  Description: String
  DisableExecuteApiEndpoint: Boolean
  Domain: HttpApiDomainConfiguration
  FailOnWarnings: Boolean
  Name: String
  PropagateTags: Boolean
  RouteSettings: RouteSettings
  StageName: String
  StageVariables: Json
  Tags: Map
```

Eigenschaften

`AccessLogSettings`

Die Einstellungen für die Zugriffsprotokollierung in einer Phase.

Typ: AccessLogSettings

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `AccessLogSettings` Eigenschaft einer `AWS::ApiGatewayV2::Stage` Ressource übergeben.

`Auth`

Konfiguriert die Autorisierung für die Steuerung des Zugriffs auf Ihre API Gateway Gateway-HTTP-API.

Weitere Informationen finden Sie unter Steuern des Zugriffs auf HTTP APIs mit JWT-Autorisierern im API Gateway Developer Guide.

Typ: HttpApiAuth

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein Äquivalent. CloudFormation

`CorsConfiguration`

Verwaltet Cross-Origin Resource Sharing (CORS) für Ihr gesamtes API-Gateway-HTTP. APIs Geben Sie die Domäne, die zugelassen werden soll, als Zeichenfolge an, oder geben Sie ein Objekt an`HttpApiCorsConfiguration`. Beachten Sie, dass CORS Ihre OpenAPI-Definition ändern muss AWS SAM , sodass CORS nur funktioniert, wenn die `DefinitionBody` Eigenschaft angegeben ist.

Weitere Informationen finden Sie unter Konfiguration von CORS für eine HTTP-API im API Gateway Developer Guide.

> **Note**
>
> Wenn sowohl in einer OpenAPI-Definition als auch auf Eigenschaftsebene festgelegt `CorsConfiguration` ist, werden beide Konfigurationsquellen AWS SAM zusammengeführt, wobei die Eigenschaften Vorrang haben. Wenn diese Eigenschaft auf gesetzt ist`true`, sind alle Ursprünge zulässig.

Typ: Zeichenfolge | HttpApiCorsConfiguration

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`DefaultRouteSettings`

Die Standardrouteneinstellungen für diese HTTP-API. Diese Einstellungen gelten für alle Routen, sofern sie nicht durch die `RouteSettings` Eigenschaft für bestimmte Routen außer Kraft gesetzt werden.

Typ: RouteSettings

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `RouteSettings` Eigenschaft einer `AWS::ApiGatewayV2::Stage` Ressource übergeben.

`DefinitionBody`

Die OpenAPI-Definition, die Ihre HTTP-API beschreibt. Wenn Sie a `DefinitionUri` oder a nicht angeben`DefinitionBody`, AWS SAM generiert es auf der Grundlage Ihrer Vorlagenkonfiguration eine `DefinitionBody` für Sie.

Type: JSON

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `Body` Eigenschaft einer `AWS::ApiGatewayV2::Api` Ressource. Wenn bestimmte Eigenschaften angegeben sind, AWS SAM kann Inhalt in die Datei eingefügt oder geändert werden, `DefinitionBody` bevor sie an sie übergeben CloudFormation werden. Zu den Eigenschaften gehören ein `EventSource` Typ `Auth` und ein Typ HttpApi für eine entsprechende `AWS::Serverless::Function` Ressource.

`DefinitionUri`

Der Amazon Simple Storage Service (Amazon S3) -URI, der lokale Dateipfad oder das Speicherortobjekt der OpenAPI-Definition, die die HTTP-API definiert. Das Amazon S3 S3-Objekt, auf das diese Eigenschaft verweist, muss eine gültige OpenAPI-Definitionsdatei sein. Wenn Sie a nicht angeben `DefinitionUri` oder a angegeben `DefinitionBody` sind, AWS SAM generiert es eine `DefinitionBody` für Sie auf der Grundlage Ihrer Vorlagenkonfiguration.

Wenn Sie einen lokalen Dateipfad angeben, muss die Vorlage den Workflow durchlaufen, der den `sam package` Befehl `sam deploy` oder enthält, damit die Definition ordnungsgemäß transformiert wird.

Systeminterne Funktionen werden in externen OpenApi Definitionsdateien, auf die Sie verweisen, nicht unterstützt. `DefinitionUri` Um eine OpenApi Definition in die Vorlage zu importieren, verwenden Sie die `DefinitionBody` Eigenschaft zusammen mit der Include-Transformation.

Typ: Zeichenfolge | HttpApiDefinition

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `BodyS3Location` Eigenschaft einer `AWS::ApiGatewayV2::Api` Ressource. Die verschachtelten Amazon S3 S3-Eigenschaften sind unterschiedlich benannt.

`Description`

Die Beschreibung der HTTP-API-Ressource.

Wenn Sie angeben`Description`, AWS SAM wird die OpenApi Definition der HTTP-API-Ressource geändert, indem das `description` Feld festgelegt wird. Die folgenden Szenarien führen zu einem Fehler:

- Die `DefinitionBody` Eigenschaft wird mit dem in der Open API-Definition festgelegten `description` Feld angegeben. Dies führt zu einem Konflikt im `description` Feld, der AWS SAM nicht gelöst werden kann.
- Die `DefinitionUri` Eigenschaft ist angegeben — ändert AWS SAM keine Open-API-Definition, die von Amazon S3 abgerufen wird.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`DisableExecuteApiEndpoint`

Gibt an, ob Clients Ihre HTTP-API mithilfe des `execute-api` Standardendpunkts `https://{api_id}.execute-api.{region}.amazonaws.com` aufrufen können. Standardmäßig können Kunden Ihre API mit dem standardmäßigen -Endpunkt aufrufen. Um zu verlangen, dass Clients nur einen benutzerdefinierten Domainnamen verwenden, um Ihre API aufzurufen, deaktivieren Sie den Standardendpunkt.

Um diese Eigenschaft zu verwenden, müssen Sie `disableExecuteApiEndpoint` in Ihrer OpenAPI-Definition die `DefinitionBody` `DefinitionUri` Eigenschaft anstelle der Eigenschaft angeben oder `x-amazon-apigateway-endpoint-configuration` mit definieren.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `DisableExecuteApiEndpoint` Eigenschaft einer `AWS::ApiGatewayV2::Api` Ressource. Sie wird direkt an die `disableExecuteApiEndpoint` Eigenschaft einer `x-amazon-apigateway-`

endpoint-configuration Erweiterung übergeben, die der Body Eigenschaft einer AWS::ApiGatewayV2::Api Ressource hinzugefügt wird.

`Domain`

Konfiguriert eine benutzerdefinierte Domain für diese API Gateway Gateway-HTTP-API.

Typ: HttpApiDomainConfiguration

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`FailOnWarnings`

Gibt an, ob die HTTP-API-Erstellung rückgängig gemacht werden soll (`true`) oder nicht (`false`), wenn eine Warnung auftritt. Der Standardwert ist `false`.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `FailOnWarnings` Eigenschaft einer `AWS::ApiGatewayV2::Api` Ressource übergeben.

`Name`

Der Name der HTTP-API-Ressource.

Wenn Sie angeben`Name`, AWS SAM wird die OpenAPI-Definition der HTTP-API-Ressource geändert, indem das `title` Feld festgelegt wird. Die folgenden Szenarien führen zu einem Fehler:

- Die `DefinitionBody` Eigenschaft wird mit dem in der Open API-Definition festgelegten `title` Feld angegeben. Dies führt zu einem Konflikt im `title` Feld, der AWS SAM nicht gelöst werden kann.
- Die `DefinitionUri` Eigenschaft ist angegeben — ändert AWS SAM keine Open-API-Definition, die von Amazon S3 abgerufen wird.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`PropagateTags`

Geben Sie an, ob Tags von der `Tags` Eigenschaft an Ihre AWS::Serverless::HttpApi generierten Ressourcen weitergegeben werden sollen oder nicht. Geben Sie `True` an, dass Tags in Ihren generierten Ressourcen verbreitet werden sollen.

Typ: Boolesch

Required: No

Standardwert: `False`

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`RouteSettings`

Die Routeneinstellungen pro Route für diese HTTP-API. Weitere Informationen finden Sie unter Working with routes for HTTP APIs im API Gateway Developer Guide.

Typ: RouteSettings

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `RouteSettings` Eigenschaft einer `AWS::ApiGatewayV2::Stage` Ressource übergeben.

`StageName`

Der Name der API-Stufe. Wenn kein Name angegeben ist, wird der `$default` Stagingbereich von API Gateway AWS SAM verwendet.

Typ: Zeichenfolge

Required: No

Standard: $default

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `StageName` Eigenschaft einer `AWS::ApiGatewayV2::Stage` Ressource übergeben.

`StageVariables`

Eine Zuweisung, welche die Stufenvariablen definiert. Variablennamen können alphanumerische Zeichen und Unterstriche enthalten. Die Werte müssen mit [a-Za-Z0-9-._~:/? #&=,] +.

Type: Json

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `StageVariables` Eigenschaft einer `AWS::ApiGatewayV2::Stage` Ressource übergeben.

`Tags`

Eine Zuordnung (Zeichenfolge zu Zeichenfolge), die die Tags angibt, die zu diesem API-Gateway-Schritt hinzugefügt werden sollen. Schlüssel können 1 bis 128 Unicode-Zeichen lang sein und dürfen das Präfix nicht enthalten`aws:`. Sie können eines der folgenden Zeichen verwenden: Unicode-Zeichen, Ziffern, Leerraum, _, `.`, `/`, =, + und `-`. Werte können 1 bis 256 Unicode-Zeichen lang sein.

Typ: Karte

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Zusätzliche Hinweise: Die `Tags` Eigenschaft erfordert eine AWS SAM Änderung Ihrer OpenAPI-Definition, sodass Tags nur hinzugefügt werden, wenn die `DefinitionBody` Eigenschaft angegeben ist — es werden keine Tags hinzugefügt, wenn die Eigenschaft angegeben ist. `DefinitionUri` AWS SAM fügt automatisch ein Tag hinzu. `httpapi:createdBy:SAM` Der `AWS::ApiGatewayV2::Stage` Ressource und der `AWS::ApiGatewayV2::DomainName` Ressource (falls `DomainName` angegeben) werden auch Tags hinzugefügt.

Rückgabewerte

Punkt

Wenn Sie die logische ID dieser Ressource an die systeminterne `Ref` Funktion übergeben, wird die API-ID der zugrunde liegenden `AWS::ApiGatewayV2::Api` Ressource `Ref` zurückgegeben, zum Beispiel. `a1bcdef2gh`

Weitere Informationen zur Verwendung der `Ref` Funktion finden Sie `Ref`im AWS CloudFormation Benutzerhandbuch.

Beispiele

Einfach HttpApi

Das folgende Beispiel zeigt das Minimum, das für die Einrichtung eines HTTP-API-Endpunkts erforderlich ist, der von einer Lambda-Funktion unterstützt wird. In diesem Beispiel wird die standardmäßige HTTP-API verwendet, die AWS SAM erstellt.

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Description: AWS SAM template with a simple API definition
Resources:
  ApiFunction:
    Type: AWS::Serverless::Function
    Properties:
      Events:
        ApiEvent:
          Type: HttpApi
      Handler: index.handler
      InlineCode: |
        def handler(event, context):
            return {'body': 'Hello World!', 'statusCode': 200}
      Runtime: python3.7
Transform: AWS::Serverless-2016-10-31
```

HttpApi mit Auth

Das folgende Beispiel zeigt, wie die Autorisierung auf HTTP-API-Endpunkten eingerichtet wird.

YAML

```
Properties:
  FailOnWarnings: true
  Auth:
    DefaultAuthorizer: OAuth2
    Authorizers:
      OAuth2:
        AuthorizationScopes:
```

```
          - scope4
        JwtConfiguration:
          issuer: "https://www.example.com/v1/connect/oauth2"
          audience:
            - MyApi
        IdentitySource: "$request.querystring.param"
```

HttpApi mit OpenAPI-Definition

Das folgende Beispiel zeigt, wie Sie der Vorlage eine OpenAPI-Definition hinzufügen.

Beachten Sie, dass alle fehlenden Lambda-Integrationen für HttpApi Ereignisse, die auf diese HTTP-API verweisen, AWS SAM ausgefüllt werden. AWS SAM fügt auch alle fehlenden Pfade hinzu, auf die HttpApi Ereignisse verweisen.

YAML

```
Properties:
  FailOnWarnings: true
  DefinitionBody:
    info:
      version: '1.0'
      title:
        Ref: AWS::StackName
    paths:
      "/":
        get:
          security:
          - OpenIdAuth:
            - scope1
            - scope2
          responses: {}
    openapi: 3.0.1
    securitySchemes:
      OpenIdAuth:
        type: openIdConnect
        x-amazon-apigateway-authorizer:
          identitySource: "$request.querystring.param"
          type: jwt
          jwtConfiguration:
            audience:
            - MyApi
            issuer: https://www.example.com/v1/connect/oidc
```

```
          openIdConnectUrl: https://www.example.com/v1/connect/oidc/.well-known/openid-
configuration
```

HttpApi mit Konfigurationseinstellungen

Das folgende Beispiel zeigt, wie HTTP-API- und Staging-Konfigurationen zur Vorlage hinzugefügt werden.

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Parameters:
  StageName:
    Type: String
    Default: Prod

Resources:
  HttpApiFunction:
    Type: AWS::Serverless::Function
    Properties:
      InlineCode: |
          def handler(event, context):
              import json
              return {
                  "statusCode": 200,
                  "body": json.dumps(event),
              }
      Handler: index.handler
      Runtime: python3.7
      Events:
        ExplicitApi: # warning: creates a public endpoint
          Type: HttpApi
          Properties:
            ApiId: !Ref HttpApi
            Method: GET
            Path: /path
            TimeoutInMillis: 15000
            PayloadFormatVersion: "2.0"
            RouteSettings:
              ThrottlingBurstLimit: 600

  HttpApi:
```

```
    Type: AWS::Serverless::HttpApi
    Properties:
      StageName: !Ref StageName
      Tags:
        Tag: Value
      AccessLogSettings:
        DestinationArn: !GetAtt AccessLogs.Arn
        Format: $context.requestId
      DefaultRouteSettings:
        ThrottlingBurstLimit: 200
      RouteSettings:
        "GET /path":
          ThrottlingBurstLimit: 500 # overridden in HttpApi Event
      StageVariables:
        StageVar: Value
      FailOnWarnings: true

  AccessLogs:
    Type: AWS::Logs::LogGroup

Outputs:
  HttpApiUrl:
    Description: URL of your API endpoint
    Value:
      Fn::Sub: 'https://${HttpApi}.execute-api.${AWS::Region}.${AWS::URLSuffix}/
${StageName}/'
  HttpApiId:
    Description: Api id of HttpApi
    Value:
      Ref: HttpApi
```

HttpApiAuth

Konfigurieren Sie die Autorisierung, um den Zugriff auf Ihre Amazon API Gateway Gateway-HTTP-API zu kontrollieren.

Weitere Informationen zur Konfiguration des Zugriffs auf HTTP APIs finden Sie unter Steuern und Verwalten des Zugriffs auf eine HTTP-API in API Gateway im API Gateway Developer Guide.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
Authorizers: OAuth2Authorizer | LambdaAuthorizer
DefaultAuthorizer: String
EnableIamAuthorizer: Boolean
```

Eigenschaften

`Authorizers`

Der Autorisierer, der zur Steuerung des Zugriffs auf Ihre API-Gateway-API verwendet wird.

Typ: OAuth2Authorizer | LambdaAuthorizer

Required: No

Standard: Keiner

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Zusätzliche Hinweise: AWS SAM Fügt die Autorisierer zur OpenAPI-Definition hinzu.

`DefaultAuthorizer`

Geben Sie den Standard-Authorizer an, der für die Autorisierung von API-Aufrufen an Ihre API-Gateway-API verwendet werden soll. Sie können ihn `AWS_IAM` als Standardautorisierer angeben, wenn er auf gesetzt `EnableIamAuthorizer` ist. `true` Geben Sie andernfalls einen Autorisierer an, den Sie in definiert haben. `Authorizers`

Typ: Zeichenfolge

Required: No

Standard: Keiner

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`EnableIamAuthorizer`

Geben Sie an, ob die IAM-Autorisierung für die API-Route verwendet werden soll.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

OAuth 2.0 Autorisierer

OAuth Beispiel für einen 2.0-Authorizer

YAML

```
Auth:
  Authorizers:
    OAuth2Authorizer:
      AuthorizationScopes:
        - scope1
        - scope2
      JwtConfiguration:
        issuer: "https://www.example.com/v1/connect/oauth2"
        audience:
          - MyApi
      IdentitySource: "$request.querystring.param"
  DefaultAuthorizer: OAuth2Authorizer
```

Ein IAM-Autorisierer

Beispiel für einen IAM-Autorisierer

YAML

```
Auth:
  EnableIamAuthorizer: true
  DefaultAuthorizer: AWS_IAM
```

LambdaAuthorizer

Konfigurieren Sie einen Lambda-Authorizer, um den Zugriff auf Ihre Amazon API Gateway Gateway-HTTP-API mit einer AWS Lambda Funktion zu kontrollieren.

Weitere Informationen und Beispiele finden Sie unter Arbeiten mit AWS Lambda Autorisierern für HTTP APIs im API Gateway Developer Guide.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
AuthorizerPayloadFormatVersion: String
EnableFunctionDefaultPermissions: Boolean
EnableSimpleResponses: Boolean
FunctionArn: String
FunctionInvokeRole: String
Identity: LambdaAuthorizationIdentity
```

Eigenschaften

`AuthorizerPayloadFormatVersion`

Gibt das Format der Nutzlast an, die an einen HTTP-API-Lambda-Genehmiger gesendet wird. Für HTTP-API-Lambda-Genehmiger erforderlich.

Dies wird an den `authorizerPayloadFormatVersion` Abschnitt von und `x-amazon-apigateway-authorizer` im `securitySchemes` Abschnitt einer OpenAPI-Definition weitergegeben.

Zulässige Werte: `1.0` oder `2.0`.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`EnableFunctionDefaultPermissions`

Standardmäßig wird der HTTP-API-Ressource keine Berechtigung zum Aufrufen des Lambda-Autorisierers erteilt. Geben Sie diese Eigenschaft an`true`, um automatisch Berechtigungen zwischen Ihrer HTTP-API-Ressource und Ihrem Lambda-Authorizer zu erstellen.

Typ: Boolesch

Required: No

Standardwert: `false`

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`EnableSimpleResponses`

Gibt an, ob ein Lambda-Genehmiger eine Antwort in einem einfachen Format zurückgibt. Standardmäßig muss ein Lambda-Autorisierer eine AWS Identity and Access Management (IAM-) Richtlinie zurückgeben. Wenn diese Option aktiviert ist, kann der Lambda-Genehmiger anstelle einer IAM-Richtlinie einen booleschen Wert zurückgeben.

Dies wird an den `enableSimpleResponses` Abschnitt von und `x-amazon-apigateway-authorizer` im `securitySchemes` Abschnitt einer OpenAPI-Definition weitergegeben.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`FunctionArn`

Der Amazon-Ressourcenname (ARN) der Lambda-Funktion, die die Autorisierung für die API bereitstellt.

Dies wird an den `authorizerUri` Abschnitt von und `x-amazon-apigateway-authorizer` im `securitySchemes` Abschnitt einer OpenAPI-Definition weitergegeben.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`FunctionInvokeRole`

Der ARN der IAM-Rolle, die über die Anmeldeinformationen verfügt, die API Gateway benötigt, um die Autorisierungsfunktion aufzurufen. Geben Sie diesen Parameter an, wenn die

ressourcenbasierte Richtlinie Ihrer Funktion keine `lambda:InvokeFunction` API-Gateway-Berechtigung gewährt.

Dies wird an den `authorizerCredentials` Abschnitt von und `x-amazon-apigateway-authorizer` im `securitySchemes` Abschnitt einer OpenAPI-Definition weitergegeben.

Weitere Informationen finden Sie unter Create a Lambda Authorizer im API Gateway Developer Guide.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Identity`

Gibt `IdentitySource` in einer eingehenden Anfrage nach einem Autorisierer ein an.

Dies wird an den `identitySource` Abschnitt von und `x-amazon-apigateway-authorizer` im `securitySchemes` Abschnitt einer OpenAPI-Definition weitergegeben.

Typ: LambdaAuthorizationIdentity

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

LambdaAuthorizer

LambdaAuthorizer Beispiel

YAML

```
Auth:
  Authorizers:
    MyLambdaAuthorizer:
      AuthorizerPayloadFormatVersion: 2.0
      FunctionArn:
```

```
        Fn::GetAtt:
          - MyAuthFunction
          - Arn
      FunctionInvokeRole:
        Fn::GetAtt:
          - LambdaAuthInvokeRole
          - Arn
      Identity:
        Headers:
          - Authorization
```

LambdaAuthorizationIdentity

Die Use-Eigenschaft kann verwendet werden, um IdentitySource in einer eingehenden Anfrage für einen Lambda-Autorisierer einen anzugeben. Weitere Informationen zu Identitätsquellen finden Sie unter Identitätsquellen im API Gateway Developer Guide.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
Context: List
Headers: List
QueryStrings: List
ReauthorizeEvery: Integer
StageVariables: List
```

Eigenschaften

`Context`

Konvertiert die angegebenen Kontextzeichenfolgen in eine Liste von Mapping-Ausdrücken im Format`$context.contextString`.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Headers`

Konvertiert die Header in eine Liste von Zuordnungsausdrücken im Format`$request.header.name`.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`QueryStrings`

Konvertiert die angegebenen Abfragezeichenfolgen in eine Liste von Zuordnungsausdrücken im Format`$request.querystring.queryString`.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`ReauthorizeEvery`

Der Zeitraum time-to-live (TTL) in Sekunden, der angibt, wie lange API Gateway Autorisierungsergebnisse zwischenspeichert. Wenn Sie einen Wert größer als 0 festlegen, speichert API Gateway die Genehmigerantworten im Cache. Der maximale Wert ist 3600, oder 1 Stunde.

Typ: Ganzzahl

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein Äquivalent. CloudFormation

`StageVariables`

Konvertiert die angegebenen Stufenvariablen in eine Liste von Mapping-Ausdrücken im Format`$stageVariables.stageVariable`.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

LambdaRequestIdentity

Beispiel für eine Lambda-Anfrage

YAML

```
Identity:
  QueryStrings:
    - auth
  Headers:
    - Authorization
  StageVariables:
    - VARIABLE
  Context:
    - authcontext
  ReauthorizeEvery: 100
```

OAuth2Authorizer

Definition für einen OAuth 2.0-Authorizer, auch bekannt als JSON Web Token (JWT) -Authorizer.

Weitere Informationen finden Sie unter Steuern des Zugriffs auf HTTP APIs mit JWT-Autorisierern im API Gateway Developer Guide.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
  AuthorizationScopes: List
  IdentitySource: String
  JwtConfiguration: Map
```

Eigenschaften

`AuthorizationScopes`

Liste der Autorisierungsbereiche für diesen Autorisierer.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`IdentitySource`

Ausdruck der Identitätsquelle für diesen Autorisierer.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`JwtConfiguration`

JWT-Konfiguration für diesen Autorisierer.

Dies wird an den `jwtConfiguration` Abschnitt von und `x-amazon-apigateway-authorizer` im `securitySchemes` Abschnitt einer OpenAPI-Definition weitergegeben.

> **Note**
>
> Eigenschaften `issuer` und `audience` unterscheiden nicht zwischen Groß- und Kleinschreibung und können entweder in Kleinbuchstaben wie in OpenAPI oder in Großbuchstaben `Issuer` und wie in verwendet werden. `Audience` AWS::ApiGatewayV2::Authorizer

Typ: Karte

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

OAuth 2.0-Autorisierer

OAuth Beispiel für einen 2.0-Authorizer

YAML

```
Auth:
  Authorizers:
    OAuth2Authorizer:
      AuthorizationScopes:
        - scope1
      JwtConfiguration:
        issuer: "https://www.example.com/v1/connect/oauth2"
        audience:
          - MyApi
      IdentitySource: "$request.querystring.param"
  DefaultAuthorizer: OAuth2Authorizer
```

HttpApiCorsConfiguration

Verwalte Cross-Origin Resource Sharing (CORS) für dein HTTP. APIs Geben Sie die Domain, die zugelassen werden soll, als Zeichenfolge an oder geben Sie ein Wörterbuch mit zusätzlicher Cors-Konfiguration an. HINWEIS: Cors benötigt SAM, um Ihre OpenAPI-Definition zu ändern, sodass es nur mit Inline funktioniert, die in der OpenApi `DefinitionBody` Eigenschaft definiert ist.

Weitere Informationen zu CORS finden Sie unter Konfiguration von CORS für eine HTTP-API im API Gateway Developer Guide.

Hinweis: Wenn sowohl in OpenAPI als auch auf Eigenschaftenebene gesetzt HttpApiCorsConfiguration ist, werden sie mit den AWS SAM Eigenschaften zusammengeführt, die Vorrang haben.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
  AllowCredentials: Boolean
  AllowHeaders: List
  AllowMethods: List
  AllowOrigins: List
  ExposeHeaders: List
  MaxAge: Integer
```

Eigenschaften

`AllowCredentials`

Gibt an, ob Anmeldeinformationen in der CORS-Anforderung enthalten sind.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`AllowHeaders`

Stellt eine Sammlung zulässiger Header dar.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`AllowMethods`

Stellt eine Sammlung zulässiger HTTP-Methoden dar.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`AllowOrigins`

Stellt eine Sammlung zulässiger Ursprünge dar.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`ExposeHeaders`

Stellt eine Sammlung exponierter Header dar.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`MaxAge`

Die Anzahl der Sekunden, während der der Browser Preflight-Anforderungsergebnisse zwischenspeichern soll.

Typ: Ganzzahl

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

HttpApiCorsConfiguration

Beispiel für eine HTTP-API-Cors-Konfiguration.

YAML

```
CorsConfiguration:
  AllowOrigins:
```

```
    - "https://example.com"
  AllowHeaders:
    - x-apigateway-header
  AllowMethods:
    - GET
  MaxAge: 600
  AllowCredentials: true
```

HttpApiDefinition

Ein OpenAPI-Dokument, das die API definiert.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
  Bucket: String
  Key: String
  Version: String
```

Eigenschaften

`Bucket`

Der Name des Amazon S3 S3-Buckets, in dem die OpenAPI-Datei gespeichert ist.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Bucket` Eigenschaft des `AWS::ApiGatewayV2::Api` `BodyS3Location` Datentyps übergeben.

`Key`

Der Amazon S3 S3-Schlüssel der OpenAPI-Datei.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Key` Eigenschaft des `AWS::ApiGatewayV2::Api BodyS3Location` Datentyps übergeben.

`Version`

Für versionierte Objekte die Version der OpenAPI-Datei.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Version` Eigenschaft des `AWS::ApiGatewayV2::Api BodyS3Location` Datentyps übergeben.

Beispiele

Definition: Uri, Beispiel

Beispiel für eine API-Definition

YAML

```
DefinitionUri:
  Bucket: sam-s3-demo-bucket-name
  Key: mykey-name
  Version: 121212
```

HttpApiDomainConfiguration

Konfiguriert eine benutzerdefinierte Domain für eine API.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
  BasePath: List
  CertificateArn: String
  DomainName: String
  EndpointConfiguration: String
```

```
  MutualTlsAuthentication: MutualTlsAuthentication
  OwnershipVerificationCertificateArn: String
  Route53: Route53Configuration
  SecurityPolicy: String
```

Eigenschaften

`BasePath`

Eine Liste der Basepfade, die mit dem Amazon API Gateway Gateway-Domänennamen konfiguriert werden sollen.

Typ: Liste

Required: No

Standard:/

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `ApiMappingKey` Eigenschaft einer `AWS::ApiGatewayV2::ApiMapping` Ressource. AWS SAM erstellt mehrere `AWS::ApiGatewayV2::ApiMapping` Ressourcen, eine pro Wert, der in dieser Eigenschaft angegeben ist.

`CertificateArn`

Der Amazon-Ressourcenname (ARN) eines AWS verwalteten Zertifikats für den Endpunkt dieses Domainnamens. AWS Certificate Manager ist die einzige unterstützte Quelle.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `CertificateArn` Eigenschaft einer `AWS::ApiGateway2::DomainName DomainNameConfiguration` Ressource übergeben.

`DomainName`

Der benutzerdefinierte Domainname für Ihre API-Gateway-API. Großbuchstaben werden nicht unterstützt.

AWS SAM generiert eine `AWS::ApiGatewayV2::DomainName` Ressource, wenn diese Eigenschaft festgelegt ist. Hinweise zu diesem Szenario finden Sie unterDomainName

Eigenschaft ist spezifiziert. Hinweise zu generierten CloudFormation Ressourcen finden Sie unterGeneriert CloudFormation Ressourcen für AWS SAM.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `DomainName` Eigenschaft einer `AWS::ApiGateway2::DomainName` Ressource übergeben.

`EndpointConfiguration`

Definiert den Typ des API-Gateway-Endpunkts, der der benutzerdefinierten Domain zugeordnet werden soll. Der Wert dieser Eigenschaft bestimmt, wie die `CertificateArn` Eigenschaft zugeordnet wird. CloudFormation

Der einzig gültige Wert für HTTP APIs ist`REGIONAL`.

Typ: Zeichenfolge

Required: No

Standardwert: `REGIONAL`

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`MutualTlsAuthentication`

Die TLS-Authentifizierungskonfiguration (Mutual Transport Layer Security) für einen benutzerdefinierten Domainnamen.

Typ: MutualTlsAuthentication

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `MutualTlsAuthentication` Eigenschaft einer `AWS::ApiGatewayV2::DomainName` Ressource übergeben.

`OwnershipVerificationCertificateArn`

Die ARN des öffentlichen Zertifikats, das von ACM ausgestellt wurde, um den Besitz Ihrer benutzerdefinierten Domain zu überprüfen. Nur erforderlich, wenn Sie Mutual TLS konfigurieren und einen von ACM importierten oder privaten CA-Zertifikat-ARN für angeben. `CertificateArn`

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `OwnershipVerificationCertificateArn` Eigenschaft des `AWS::ApiGatewayV2::DomainName` `DomainNameConfiguration` Datentyps übergeben.

`Route53`

Definiert eine Amazon Route 53-Konfiguration.

Typ: Route53-Konfiguration

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein Äquivalent. CloudFormation

`SecurityPolicy`

Die TLS-Version der Sicherheitsrichtlinie für diesen Domainnamen.

Der einzig gültige Wert für HTTP APIs ist`TLS_1_2`.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `SecurityPolicy` Eigenschaft des `AWS::ApiGatewayV2::DomainName` `DomainNameConfiguration` Datentyps übergeben.

Beispiele

DomainName

DomainName Beispiel

YAML

```
Domain:
```

```
  DomainName: www.example.com
  CertificateArn: arn-example
  EndpointConfiguration: REGIONAL
  Route53:
    HostedZoneId: Z1PA6795UKMFR9
  BasePath:
    - foo
    - bar
```

Route53Configuration

Konfiguriert die Route53-Datensätze für eine API.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer AWS Serverless Application Model (AWS SAM) -Vorlage zu deklarieren.

YAML

```
DistributionDomainName: String
EvaluateTargetHealth: Boolean
HostedZoneId: String
HostedZoneName: String
IpV6: Boolean
Region: String
SetIdentifier: String
```

Eigenschaften

`DistributionDomainName`

Konfiguriert eine benutzerdefinierte Verteilung des benutzerdefinierten API-Domänennamens.

Typ: Zeichenfolge

Required: No

Standard: Verwenden Sie die API Gateway Gateway-Distribution.

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die DNSName Eigenschaft einer `AWS::Route53::RecordSetGroup AliasTarget` Ressource übergeben.

Zusätzliche Hinweise: Der Domainname einer CloudFrontDistribution.

`EvaluateTargetHealth`

Wenn EvaluateTargetHealth wahr, erbt ein Aliaseintrag den Zustand der referenzierten AWS Ressource, z. B. eines Elastic Load Balancing Balancing-Load Balancers oder eines anderen Datensatzes in der Hosting-Zone.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `EvaluateTargetHealth` Eigenschaft einer `AWS::Route53::RecordSetGroup AliasTarget` Ressource übergeben.

Zusätzliche Hinweise: Sie können den Wert nicht auf „true“ EvaluateTargetHealth setzen, wenn es sich bei dem Alias-Ziel um eine CloudFront Distribution handelt.

`HostedZoneId`

Die ID der gehosteten Zone, in der Sie Datensätze erstellen möchten.

Geben Sie `HostedZoneName` oder `HostedZoneId` an, jedoch nicht beides. Wenn Sie mehrere gehostete Zonen mit dem gleichen Domainnamen haben, müssen Sie die gehostete Zone mit der `HostedZoneId` angeben.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `HostedZoneId` Eigenschaft einer `AWS::Route53::RecordSetGroup RecordSet` Ressource übergeben.

`HostedZoneName`

Der Name der gehosteten Zone, in der Sie Datensätze erstellen möchten. Sie müssen einen abschließenden Punkt (z. B. `www.example.com.`) als Teil des `HostedZoneName` angeben.

Geben Sie `HostedZoneName` oder `HostedZoneId` an, jedoch nicht beides. Wenn Sie mehrere gehostete Zonen mit dem gleichen Domainnamen haben, müssen Sie die gehostete Zone mit der `HostedZoneId` angeben.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `HostedZoneName` Eigenschaft einer `AWS::Route53::RecordSetGroup RecordSet` Ressource übergeben.

`IpV6`

Wenn diese Eigenschaft gesetzt ist, wird eine `AWS::Route53::RecordSet` Ressource AWS SAM erstellt und `Type` `AAAA` für die angegebene Ressource auf gesetzt HostedZone.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Region`

Nur latenzbasierte Ressourcendatensätze: Die EC2 Amazon-Region, in der Sie die Ressource erstellt haben, auf die sich dieser Ressourcendatensatz bezieht. Bei der Ressource handelt es sich in der Regel um eine AWS Ressource, z. B. eine EC2 Instance oder einen ELB-Load Balancer, auf die je nach Datensatztyp mit einer IP-Adresse oder einem DNS-Domainnamen verwiesen wird.

Wenn Amazon Route 53 eine DNS-Abfrage für einen Domainnamen und einen Domain-Typ empfängt, für den Sie Latenzressourcendatensätze erstellt haben, wählt Route 53 den Latenzressourcendatensatz aus, der die niedrigste Latenz zwischen dem Endbenutzer und der zugehörigen EC2 Amazon-Region aufweist. Route 53 gibt dann den Wert zurück, der dem ausgewählten Ressourcendatensatz zugeordnet ist.

Beachten Sie Folgendes:

- Sie können nur einen `ResourceRecord` pro Latenz-Ressourcendatensatz angeben.
- Sie können nur einen Latenzressourcendatensatz für jede EC2 Amazon-Region erstellen.
- Sie müssen keine Latenzressourcendatensätze für alle EC2 Amazon-Regionen erstellen. Route 53 wählt die Region mit der besten Latenz aus den Regionen aus, für die Sie Latenz-Ressourcendatensätze erstellen.
- Sie können keine Nicht-Latenz-Ressourcendatensätze erstellen, die die gleichen Werte für die Elemente `Name` und `Type` haben wie Latenz-Ressourcendatensätze.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Region` Eigenschaft eines `AWS::Route53::RecordSetGroup` `RecordSet` Datentyps übergeben.

`SetIdentifier`

Ressourcendatensätze, die eine andere Routing-Richtlinie als "einfach" haben: Ein Bezeichner, der zwischen mehreren Ressourcendatensätzen unterscheidet, die die gleiche Kombination aus Name und Typ haben, wie beispielsweise mehrere gewichtete Ressourcendatensätze namens acme.example.com, die einen Typ A haben. In einer Gruppe von Ressourcendatensätzen, die den gleichen Namen und Typ haben, muss der Wert von `SetIdentifier` für jeden Ressourcen-Datensatz eindeutig sein.

Informationen zu Routing-Richtlinien finden Sie unter Auswahl einer Routing-Richtlinie im Amazon Route 53-Entwicklerhandbuch.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `SetIdentifier` Eigenschaft eines `AWS::Route53::RecordSetGroup` `RecordSet` Datentyps übergeben.

Beispiele

Konfigurationsbeispiel für Route 53

Dieses Beispiel zeigt, wie Route 53 konfiguriert wird.

YAML

```
Domain:
  DomainName: www.example.com
  CertificateArn: arn-example
  EndpointConfiguration: EDGE
  Route53:
    HostedZoneId: Z1PA6795UKMFR9
    EvaluateTargetHealth: true
    DistributionDomainName: xyz
```

AWS::Serverless::WebSocketApi

Erstellt eine Amazon API Gateway WebSocket Gateway-API, mit der Sie interaktive bidirektionale Kommunikationsanwendungen erstellen können. WebSocket APIs ermöglichen es dem Server, Nachrichten an Clients zu senden, ohne dass der Client sie anfordern muss. Weitere Informationen finden Sie unter Working with WebSocket APIs im API Gateway Developer Guide.

Wir empfehlen, CloudFormation Hooks oder IAM-Richtlinien zu verwenden, um zu überprüfen, ob an API-Gateway-Ressourcen Autorisatoren angehängt sind, um den Zugriff darauf zu kontrollieren.

Weitere Informationen zur Verwendung von CloudFormation Hooks finden Sie unter Hooks registrieren im CloudFormation CLI-Benutzerhandbuch und im GitHub apigw-enforce-authorizer-Repository.

Weitere Informationen zur Verwendung von IAM-Richtlinien finden Sie unter Erfordern, dass API-Routen autorisiert sind im API Gateway Developer Guide.

> **Note**
>
> Bei der Bereitstellung auf werden AWS CloudFormation Ihre AWS SAM Ressourcen in Ressourcen umgewandelt CloudFormation . AWS SAM Weitere Informationen finden Sie unter Generiert CloudFormation Ressourcen für AWS SAM.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
Type: AWS::Serverless::WebSocketApi
Properties:
  ApiKeySelectionExpression: String
  AccessLogSettings: AccessLogSettings
  Auth: WebSocketApiAuth
  DefaultRouteSettings: RouteSettings
  Description: String
  DisableExecuteApiEndpoint: Boolean
  DisableSchemaValidation: Boolean
```

```
  Domain: WebSocketApiDomainConfiguration
  IpAddressType: String
  Name: String
  PropagateTags: Boolean
  Routes: RouteConfiguration
  RouteSelectionExpression: String
  RouteSettings: RouteSettings
  StageName: String
  StageVariables: Json
  Tags: Map
```

Eigenschaften

`ApiKeySelectionExpression`

Ein API-Schlüssel-Auswahlausdruck. Weitere Informationen finden Sie unter API-Schlüsselauswahlausdrücke im API Gateway Developer Guide.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `ApiKeySelectionExpression` Eigenschaft einer `AWS::ApiGatewayV2::Api` Ressource übergeben.

`AccessLogSettings`

Die Einstellungen für die Zugriffsprotokollierung in einer Phase.

Typ: AccessLogSettings

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `AccessLogSettings` Eigenschaft einer `AWS::ApiGatewayV2::Stage` Ressource übergeben.

`Auth`

Konfiguriert die Autorisierung zur Steuerung des Zugriffs auf Ihre WebSocket API. Die Autorisierung wird auf die `$connect` Route angewendet.

Weitere Informationen finden Sie unter Steuern des Zugriffs auf WebSocket APIs im API Gateway Developer Guide.

Typ: WebSocketApiAuth

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`DefaultRouteSettings`

Die Standardrouteneinstellungen für diese WebSocket API. Diese Einstellungen gelten für alle Routen, sofern sie nicht durch die `RouteSettings` Eigenschaft für bestimmte Routen außer Kraft gesetzt werden.

Typ: RouteSettings

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `DefaultRouteSettings` Eigenschaft einer `AWS::ApiGatewayV2::Stage` Ressource übergeben.

`Description`

Eine Beschreibung der WebSocket API.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Description` Eigenschaft einer `AWS::ApiGatewayV2::Api` Ressource übergeben.

`DisableExecuteApiEndpoint`

Gibt an, ob Clients Ihre API mithilfe des `execute-api`-Standardendpunkts aufrufen können. Deaktivieren Sie den Standardendpunkt, um bei Clients die Verwendung eines benutzerdefinierten Domänennamen zum Aufrufen Ihrer API erforderlich zu machen.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `DisableExecuteApiEndpoint` Eigenschaft einer `AWS::ApiGatewayV2::Api` Ressource übergeben.

`DisableSchemaValidation`

Vermeiden Sie die Validierung von Modellen, wenn Sie eine Bereitstellung erstellen.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `DisableSchemaValidation` Eigenschaft einer `AWS::ApiGatewayV2::Api` Ressource übergeben.

`Domain`

Konfiguriert eine benutzerdefinierte Domain für diese WebSocket API.

> **Note**
>
> WebSocket APIs unterstützen keine gegenseitige TLS-Authentifizierung (MTLS). Wenn Sie `MutualTlsAuthentication` oder angeben`OwnershipVerificationCertificateArn`, AWS SAM wird ein Fehler zurückgegeben.

Typ: WebSocketApiDomainConfiguration

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`IpAddressType`

Der IP-Adresstyp für die API. Gültige Werte gelten nur `ipv4` für IPv4 sowie `dualstack` für IPv4 und IPv6.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `IpAddressType` Eigenschaft einer Ressource übergeben. `AWS::ApiGatewayV2::Api`

`Name`

Ein Name für die WebSocket API. Wenn Sie keinen Namen angeben, AWS SAM generiert es einen Namen für Sie.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Name` Eigenschaft einer `AWS::ApiGatewayV2::Api` Ressource übergeben.

`PropagateTags`

Wenn`true`, AWS SAM fügt die `Tags` Eigenschaft zu den `AWS::ApiGatewayV2::DomainName` Ressourcen `AWS::ApiGatewayV2::Stage` hinzu, die AWS SAM generiert werden.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Routes`

Die Routenkonfigurationen für diese WebSocket API. Routen definieren, wie Nachrichten an Lambda-Funktionen weitergeleitet werden. Jede Route besteht aus einem Routenschlüssel und einem Lambda-Funktions-ARN.

WebSocket APIs unterstützen drei vordefinierte Routen: `$connect``$disconnect`, und`$default`. Sie können auch benutzerdefinierte Routen definieren.

Typ: RouteConfiguration

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`RouteSelectionExpression`

Der Routenauswahlausdruck für die WebSocket API. Weitere Informationen finden Sie unter Route Selection Expressions im API Gateway Developer Guide.

Ein gängiger Wert ist`$request.body.action`, der Nachrichten auf der Grundlage eines `action` Felds im Nachrichtentext weiterleitet.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `RouteSelectionExpression` Eigenschaft einer `AWS::ApiGatewayV2::Api` Ressource übergeben.

`RouteSettings`

Die Routeneinstellungen für diese WebSocket API. Diese Einstellungen überschreiben die `DefaultRouteSettings` für bestimmte Routen.

Typ: RouteSettings

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `RouteSettings` Eigenschaft einer `AWS::ApiGatewayV2::Stage` Ressource übergeben.

`StageName`

Der Name der API-Stufe. Wenn Sie keinen Namen angeben, AWS SAM wird er `default` als Phasennamen verwendet.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `StageName` Eigenschaft einer `AWS::ApiGatewayV2::Stage` Ressource übergeben.

`StageVariables`

Eine Zuweisung, welche die Stufenvariablen definiert. Variablennamen können alphanumerische Zeichen und Unterstriche enthalten, und die Werte müssen übereinstimmen. `[A-Za-z0-9-._~:/?#&=,]+`

Type: Json

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `StageVariables` Eigenschaft einer `AWS::ApiGatewayV2::Stage` Ressource übergeben.

`Tags`

Eine Zuordnung (Zeichenfolge zu Zeichenfolge), die die Tags spezifiziert, die zu dieser WebSocket API hinzugefügt werden sollen. Einzelheiten zu gültigen Schlüsseln und Werten für Tags finden Sie unter Resource-Tag im CloudFormation Benutzerhandbuch.

Typ: Karte

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Rückgabewerte

Punkt

Wenn Sie die logische ID dieser Ressource an die systemeigene `Ref` Funktion übergeben, wird die API-ID der zugrunde liegenden `AWS::ApiGatewayV2::Api` Ressource `Ref` zurückgegeben, z. B. `a1bcdef2gh`

Weitere Informationen zur Verwendung der `Ref` Funktion finden Sie `Ref`im AWS CloudFormation Benutzerhandbuch.

Beispiele

Einfache WebSocket API

Das folgende Beispiel erstellt eine WebSocket API mit drei Routen.

```
Resources:
  MyWebSocketApi:
    Type: AWS::Serverless::WebSocketApi
    Properties:
      RouteSelectionExpression: $request.body.action
      Routes:
        $connect:
          FunctionArn: !GetAtt ConnectFunction.Arn
        $disconnect:
```

```
          FunctionArn: !GetAtt DisconnectFunction.Arn
        sendMessage:
          FunctionArn: !GetAtt SendMessageFunction.Arn

  ConnectFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.connect
      Runtime: nodejs20.x
      CodeUri: ./src

  DisconnectFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.disconnect
      Runtime: nodejs20.x
      CodeUri: ./src

  SendMessageFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.sendMessage
      Runtime: nodejs20.x
      CodeUri: ./src
```

WebSocket API mit Lambda Authorizer

Das folgende Beispiel erstellt eine WebSocket API mit einem Lambda-Authorizer.

```
Resources:
  MyWebSocketApi:
    Type: AWS::Serverless::WebSocketApi
    Properties:
      RouteSelectionExpression: $request.body.action
      Auth:
        AuthType: CUSTOM
        AuthArn: !GetAtt AuthorizerFunction.Arn
        IdentitySource:
          - route.request.header.Authorization
      Routes:
        $connect:
          FunctionArn: !GetAtt ConnectFunction.Arn
        sendMessage:
          FunctionArn: !GetAtt SendMessageFunction.Arn
```

```
  AuthorizerFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.authorize
      Runtime: nodejs20.x
      CodeUri: ./src

  ConnectFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.connect
      Runtime: nodejs20.x
      CodeUri: ./src

  SendMessageFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.sendMessage
      Runtime: nodejs20.x
      CodeUri: ./src
```

WebSocketApiAuth

Konfigurieren Sie die Autorisierung, um den Zugriff auf Ihre Amazon API Gateway WebSocket Gateway-API zu kontrollieren.

Weitere Informationen zur Konfiguration des Zugriffs auf WebSocket APIs finden Sie unter Steuern des Zugriffs auf WebSocket APIs im API Gateway Developer Guide.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
AuthArn: String
AuthType: String
IdentitySource: List
InvokeRole: String
Name: String
```

Eigenschaften

`AuthArn`

Der ARN der Lambda-Funktion, die für die Autorisierung verwendet werden soll. Erforderlich, wann `AuthType` ist`CUSTOM`.

Typ: Zeichenfolge

Required: Conditional

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`AuthType`

Die Art der Autorisierung. Gültige Werte sind `NONE`, `AWS_IAM` oder `CUSTOM`.

- `NONE`- Keine Autorisierung
- `AWS_IAM`- IAM-Autorisierung
- `CUSTOM`- Lambda-Autorisierer

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`IdentitySource`

Die Identitätsquelle, für die die Autorisierung angefordert wird. Beispiel, `route.request.header.Authorization`.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `IdentitySource` Eigenschaft einer `AWS::ApiGatewayV2::Authorizer` Ressource übergeben.

`InvokeRole`

Der ARN der IAM-Rolle, die API Gateway beim Aufrufen der Autorisierungsfunktion annimmt. Wenn nicht angegeben, AWS SAM wird automatisch eine ressourcenbasierte Berechtigung erstellt, die es API Gateway ermöglicht, die Autorisierungsfunktion aufzurufen.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `AuthorizerCredentialsArn` Eigenschaft einer Ressource übergeben. `AWS::ApiGatewayV2::Authorizer`

`Name`

Der Name des Genehmigers.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Name` Eigenschaft einer `AWS::ApiGatewayV2::Authorizer` Ressource übergeben.

Beispiele

Lambda-Autorisierer

Im folgenden Beispiel wird ein Lambda-Authorizer für eine API konfiguriert. WebSocket

```
Auth:
  AuthType: CUSTOM
  AuthArn: !GetAtt AuthorizerFunction.Arn
  IdentitySource:
    - route.request.header.Authorization
```

IAM-Autorisierung

Im folgenden Beispiel wird die IAM-Autorisierung für eine API konfiguriert. WebSocket

```
Auth:
  AuthType: AWS_IAM
```

WebSocketApiDomainConfiguration

Konfiguriert eine benutzerdefinierte Domain für eine WebSocket API.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
BasePath: List
CertificateArn: String
DomainName: String
EndpointConfiguration: String
Route53: Route53Configuration
SecurityPolicy: String
```

Eigenschaften

`BasePath`

Eine Liste der Basepfade, die mit dem Amazon API Gateway Gateway-Domänennamen konfiguriert werden sollen.

Typ: Liste

Required: No

Standard:/

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `ApiMappingKey` Eigenschaft einer `AWS::ApiGatewayV2::ApiMapping` Ressource. AWS SAM erstellt mehrere `AWS::ApiGatewayV2::ApiMapping` Ressourcen, eine pro Wert, der in dieser Eigenschaft angegeben ist.

`CertificateArn`

Der Amazon-Ressourcenname (ARN) eines AWS verwalteten Zertifikats für den Endpunkt dieses Domainnamens. AWS Certificate Manager ist die einzige unterstützte Quelle.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `CertificateArn` Eigenschaft einer `AWS::ApiGatewayV2::DomainName DomainNameConfiguration` Ressource übergeben.

`DomainName`

Der benutzerdefinierte Domainname für Ihre WebSocket API-Gateway-API. Großbuchstaben werden nicht unterstützt.

AWS SAM generiert eine `AWS::ApiGatewayV2::DomainName` Ressource, wenn diese Eigenschaft festgelegt ist. Hinweise zu generierten CloudFormation Ressourcen finden Sie unterGeneriert CloudFormation Ressourcen für AWS SAM.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `DomainName` Eigenschaft einer `AWS::ApiGatewayV2::DomainName` Ressource übergeben.

`EndpointConfiguration`

Definiert den Typ des API-Gateway-Endpunkts, der der benutzerdefinierten Domain zugeordnet werden soll. Der Wert dieser Eigenschaft bestimmt, wie die `CertificateArn` Eigenschaft zugeordnet wird. CloudFormation

Der einzig gültige Wert für WebSocket APIs ist`REGIONAL`.

Typ: Zeichenfolge

Required: No

Standardwert: `REGIONAL`

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Route53`

Definiert eine Route 53-Konfiguration.

Typ: Route53-Konfiguration

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein Äquivalent. CloudFormation

`SecurityPolicy`

Die TLS-Version der Sicherheitsrichtlinie für diesen Domainnamen.

Der einzig gültige Wert für WebSocket APIs ist`TLS_1_2`.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `SecurityPolicy` Eigenschaft des `AWS::ApiGatewayV2::DomainName` `DomainNameConfiguration` Datentyps übergeben.

Beispiele

DomainName

DomainName Beispiel

YAML

```
Domain:
  DomainName: ws.example.com
  CertificateArn: arn:aws:acm:us-east-1:123456789012:certificate/example
  EndpointConfiguration: REGIONAL
  Route53:
    HostedZoneId: Z1PA6795UKMFR9
  BasePath:
    - v1
    - v2
```

Route53Configuration

Konfiguriert die Route53-Datensätze für eine API.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer AWS Serverless Application Model (AWS SAM) -Vorlage zu deklarieren.

YAML

```
  EvaluateTargetHealth: Boolean
```

```
  HostedZoneId: String
  HostedZoneName: String
  IpV6: Boolean
  Region: String
  SetIdentifier: String
```

Eigenschaften

`EvaluateTargetHealth`

Wenn EvaluateTargetHealth wahr, erbt ein Aliaseintrag den Zustand der referenzierten AWS Ressource, z. B. eines Elastic Load Balancing Balancing-Load Balancers oder eines anderen Datensatzes in der Hosting-Zone.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `EvaluateTargetHealth` Eigenschaft einer `AWS::Route53::RecordSetGroup AliasTarget` Ressource übergeben.

Zusätzliche Hinweise: Sie können den Wert nicht auf „true“ EvaluateTargetHealth setzen, wenn es sich bei dem Alias-Ziel um eine CloudFront Distribution handelt.

`HostedZoneId`

Die ID der gehosteten Zone, in der Sie Datensätze erstellen möchten.

Geben Sie `HostedZoneName` oder `HostedZoneId` an, jedoch nicht beides. Wenn Sie mehrere gehostete Zonen mit dem gleichen Domainnamen haben, müssen Sie die gehostete Zone mit der `HostedZoneId` angeben.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `HostedZoneId` Eigenschaft einer `AWS::Route53::RecordSetGroup RecordSet` Ressource übergeben.

`HostedZoneName`

Der Name der gehosteten Zone, in der Sie Datensätze erstellen möchten. Sie müssen einen abschließenden Punkt (z. B. `www.example.com.`) als Teil des `HostedZoneName` angeben.

Geben Sie `HostedZoneName` oder `HostedZoneId` an, jedoch nicht beides. Wenn Sie mehrere gehostete Zonen mit dem gleichen Domainnamen haben, müssen Sie die gehostete Zone mit der `HostedZoneId` angeben.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `HostedZoneName` Eigenschaft einer `AWS::Route53::RecordSetGroup RecordSet` Ressource übergeben.

`IpV6`

Wenn diese Eigenschaft gesetzt ist, wird eine `AWS::Route53::RecordSet` Ressource AWS SAM erstellt und Type AAAA für die angegebene Ressource auf gesetzt HostedZone.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Region`

Nur latenzbasierte Ressourcendatensätze: Die Amazon EC2-Region, in der Sie die Ressource erstellt haben, auf die sich dieser Ressourcendatensatz bezieht. Bei der Ressource handelt es sich in der Regel um eine AWS Ressource, z. B. eine EC2-Instance oder einen ELB-Load Balancer, auf die je nach Datensatztyp mit einer IP-Adresse oder einem DNS-Domainnamen verwiesen wird.

Wenn Amazon Route 53 eine DNS-Abfrage nach einem Domainnamen und -typ erhält, für den Sie Latenzressourcen-Datensätze erstellt haben, wählt Route 53 den Latenzressourcen-Datensatz aus, der die niedrigste Latenzzeit zwischen dem Endbenutzer und der zugehörigen Amazon EC2-Region aufweist.Input Route 53 gibt dann den Wert zurück, der dem ausgewählten Ressourcendatensatz zugeordnet ist.

Beachten Sie Folgendes:

- Sie können nur einen `ResourceRecord` pro Latenz-Ressourcendatensatz angeben.
- Sie können nur einen Latenz-Ressourcendatensatz für jede Amazon EC2-Region erstellen.

- Sie sind nicht verpflichtet, Latenz-Ressourcendatensätze für alle Amazon EC2-Regionen zu erstellen. Route 53 wählt die Region mit der besten Latenz aus den Regionen aus, für die Sie Latenz-Ressourcendatensätze erstellen.
- Sie können keine Nicht-Latenz-Ressourcendatensätze erstellen, die die gleichen Werte für die Elemente `Name` und `Type` haben wie Latenz-Ressourcendatensätze.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Region` Eigenschaft eines `AWS::Route53::RecordSetGroup` `RecordSet` Datentyps übergeben.

`SetIdentifier`

Ressourcendatensätze, die eine andere Routing-Richtlinie als "einfach" haben: Ein Bezeichner, der zwischen mehreren Ressourcendatensätzen unterscheidet, die die gleiche Kombination aus Name und Typ haben, wie beispielsweise mehrere gewichtete Ressourcendatensätze namens acme.example.com, die einen Typ A haben. In einer Gruppe von Ressourcendatensätzen, die den gleichen Namen und Typ haben, muss der Wert von `SetIdentifier` für jeden Ressourcen-Datensatz eindeutig sein.

Informationen zu Routing-Richtlinien finden Sie unter Auswahl einer Routing-Richtlinie im Amazon Route 53-Entwicklerhandbuch.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `SetIdentifier` Eigenschaft eines `AWS::Route53::RecordSetGroup` `RecordSet` Datentyps übergeben.

Beispiele

Konfigurationsbeispiel für Route 53

Dieses Beispiel zeigt, wie Route 53 konfiguriert wird.

YAML

```
Domain:
```

```
  DomainName: www.example.com
  CertificateArn: arn-example
  EndpointConfiguration: REGIONAL
  Route53:
    HostedZoneId: Z1PA6795UKMFR9
    EvaluateTargetHealth: true
```

RouteConfiguration

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
ApiKeyRequired: Boolean
FunctionArn: String
IntegrationTimeout: Integer
ModelSelectionExpression: String
OperationName: String
RequestModels: Json
RequestParameters: Json
RouteResponseSelectionExpression: String
```

Eigenschaften

`ApiKeyRequired`

Gibt an, ob ein API-Schlüssel für diese Route erforderlich ist.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `ApiKeyRequired` Eigenschaft einer `AWS::ApiGatewayV2::Route` Ressource übergeben.

`FunctionArn`

Der ARN der Lambda-Funktion, die diese Route verarbeitet. AWS SAM erstellt eine Integration und die erforderlichen Berechtigungen für API Gateway, um die Funktion aufzurufen.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`IntegrationTimeout`

Das Timeout für die Integration in Millisekunden. Der Höchstwert beträgt 29.000 Millisekunden (29 Sekunden).

Typ: Ganzzahl

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `TimeoutInMillis` Eigenschaft einer Ressource übergeben. `AWS::ApiGatewayV2::Integration`

`ModelSelectionExpression`

Der Modell-Auswahlausdruck für die Route.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `ModelSelectionExpression` Eigenschaft einer `AWS::ApiGatewayV2::Route` Ressource übergeben.

`OperationName`

Der Operationsname für die Route.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `OperationName` Eigenschaft einer `AWS::ApiGatewayV2::Route` Ressource übergeben.

`RequestModels`

Die Anforderungsmodelle für die Route.

Type: Json

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `RequestModels` Eigenschaft einer `AWS::ApiGatewayV2::Route` Ressource übergeben.

`RequestParameters`

Die Anforderungsparameter für die Route.

Type: Json

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `RequestParameters` Eigenschaft einer `AWS::ApiGatewayV2::Route` Ressource übergeben.

`RouteResponseSelectionExpression`

Der Routing-Antwort-Auswahlausdruck für die Route.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `RouteResponseSelectionExpression` Eigenschaft einer `AWS::ApiGatewayV2::Route` Ressource übergeben.

Beispiele

Einfache Route

Das folgende Beispiel konfiguriert eine einfache Route.

```
Routes:
  $connect:
    FunctionArn: !GetAtt ConnectFunction.Arn
  sendMessage:
    FunctionArn: !GetAtt SendMessageFunction.Arn
    OperationName: SendMessage
```

AWS::Serverless::LayerVersion

Erzeugt ein Lambda LayerVersion , das Bibliotheks- oder Laufzeitcode enthält, der von einer Lambda-Funktion benötigt wird.

Die AWS::Serverless::LayerVersion Ressource unterstützt auch das `Metadata` Ressourcenattribut, sodass Sie anweisen können, Ebenen AWS SAM zu erstellen, die in Ihrer Anwendung enthalten sind. Weitere Informationen zum Erstellen von Layern finden Sie unterAufbau von Lambda-Schichten in AWS SAM.

Wichtiger Hinweis: Seit der Veröffentlichung des UpdateReplacePolicyRessourcenattributs in CloudFormation bietet AWS::Lambda::LayerVersion(empfohlen) dieselben Vorteile wieAWS::Serverless::LayerVersion.

Wenn ein Serverless transformiert LayerVersion wird, transformiert SAM auch die logische ID der Ressource, sodass alte Ressourcen nicht automatisch gelöscht LayerVersions werden, CloudFormation wenn die Ressource aktualisiert wird.

> **Note**
>
> Bei der Bereitstellung auf werden AWS CloudFormation Ihre AWS SAM Ressourcen in AWS SAM Ressourcen umgewandelt. CloudFormation Weitere Informationen finden Sie unter Generiert CloudFormation Ressourcen für AWS SAM.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
Type: AWS::Serverless::LayerVersion
Properties:
  CompatibleArchitectures: List
  CompatibleRuntimes: List
  ContentUri: String | LayerContent
  Description: String
  LayerName: String
  LicenseInfo: String
  PublishLambdaVersion: Boolean
```

```
  RetentionPolicy: String
```

Eigenschaften

`CompatibleArchitectures`

Gibt die unterstützten Befehlssatzarchitekturen für die Layer-Version an.

Weitere Informationen zu dieser Eigenschaft finden Sie unter Lambda-Befehlssatzarchitekturen im AWS Lambda Developer Guide.

Zulässige Werte: `x86_64`, `arm64`

Typ: Liste

Required: No

Standardwert: `x86_64`

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `CompatibleArchitectures` Eigenschaft einer `AWS::Lambda::LayerVersion` Ressource übergeben.

`CompatibleRuntimes`

Liste der damit kompatiblen Laufzeiten. LayerVersion

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `CompatibleRuntimes` Eigenschaft einer `AWS::Lambda::LayerVersion` Ressource übergeben.

`ContentUri`

Amazon S3 S3-URI, Pfad zum lokalen Ordner oder LayerContent Objekt des Layer-Codes.

Wenn eine Amazon S3 S3-Uri oder ein Amazon LayerContent S3-Objekt bereitgestellt wird, muss es sich bei dem referenzierten Amazon S3-Objekt um ein gültiges ZIP-Archiv handeln, das den Inhalt einer Lambda-Schicht enthält.

Wenn ein Pfad zu einem lokalen Ordner angegeben wird, muss die Vorlage, damit der Inhalt ordnungsgemäß transformiert werden kann, den gleicher Build folgenden Workflow durchlaufen:

entweder sam deploy odersam package. Standardmäßig werden relative Pfade in Bezug auf den Speicherort der AWS SAM Vorlage aufgelöst.

Typ: Zeichenfolge | LayerContent

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `Content` Eigenschaft einer `AWS::Lambda::LayerVersion` Ressource. Die verschachtelten Amazon S3 S3-Eigenschaften sind unterschiedlich benannt.

`Description`

Beschreibung dieser Ebene.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Description` Eigenschaft einer `AWS::Lambda::LayerVersion` Ressource übergeben.

`LayerName`

Der Name oder der Amazon-Ressourcenname (ARN) des Layers.

Typ: Zeichenfolge

Required: No

Standard: Logische Ressourcen-ID

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `LayerName` Eigenschaft einer `AWS::Lambda::LayerVersion` Ressource. Wenn Sie keinen Namen angeben, wird die logische ID der Ressource als Name verwendet.

`LicenseInfo`

Informationen zur Lizenz dafür LayerVersion.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `LicenseInfo` Eigenschaft einer `AWS::Lambda::LayerVersion` Ressource übergeben.

`PublishLambdaVersion`

Eine Opt-in-Eigenschaft, die bei jeder Änderung der referenzierten `LayerVersion` Ressource eine neue Lambda-Version erstellt. Wenn sie mit `AutoPublishAlias` und `AutoPublishAliasAllProperties` in der verbundenen Lambda-Funktion aktiviert ist, wird für jede Änderung an der Ressource eine neue Lambda-Version erstellt. `LayerVersion`

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`RetentionPolicy`

Diese Eigenschaft gibt an, ob alte Versionen von Ihnen `LayerVersion` beibehalten oder gelöscht werden, wenn Sie eine Ressource löschen. Wenn Sie `LayerVersion` beim Aktualisieren oder Ersetzen einer Ressource alte Versionen von Ihrem beibehalten müssen, muss das `UpdateReplacePolicy` Attribut aktiviert sein. Informationen dazu finden Sie unter `UpdateReplacePolicyAttribut` im AWS CloudFormation Benutzerhandbuch.

Zulässige Werte: `Retain` oder `Delete`.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Zusätzliche Hinweise: Wenn Sie angeben`Retain`, AWS SAM wird der transformierten `DeletionPolicy: Retain` `AWS::Lambda::LayerVersion` Ressource ein Ressourcenattribute, unterstützt von AWS SAM of hinzugefügt.

Rückgabewerte

Punkt

Wenn die logische ID dieser Ressource für die `Ref` intrinsische Funktion bereitgestellt wird, gibt sie den Ressourcen-ARN des zugrunde liegenden Lambda zurück. LayerVersion

Weitere Informationen zur Verwendung der *Ref* Funktion finden Sie *Ref*im AWS CloudFormation Benutzerhandbuch.

Beispiele

LayerVersionExample

Beispiel für ein LayerVersion

YAML

```
Properties:
  LayerName: MyLayer
  Description: Layer description
  ContentUri: 's3://sam-s3-demo-bucket/my-layer.zip'
  CompatibleRuntimes:
    - nodejs10.x
    - nodejs12.x
  LicenseInfo: 'Available under the MIT-0 license.'
  RetentionPolicy: Retain
```

LayerContent

Ein ZIP-Archiv, das die Inhalte einer Lambda-Ebene enthält.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer AWS Serverless Application Model (AWS SAM) -Vorlage zu deklarieren.

YAML

```
  Bucket: String
  Key: String
  StorageMode: String
  Version: String
```

Eigenschaften

`Bucket`

Der Amazon S3-Bucket des Layer-Archivs.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `S3Bucket` Eigenschaft des `AWS::Lambda::LayerVersion` `Content` Datentyps übergeben.

`Key`

Der Amazon S3-Schlüssel des Layer-Archivs.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `S3Key` Eigenschaft des `AWS::Lambda::LayerVersion` `Content` Datentyps übergeben.

`StorageMode`

Steuert, wie Lambda das Layer-Archiv speichert.

- `COPY`— Lambda lädt eine Kopie Ihres Layer-Archivs in den Speicher hoch. Lambda-managed Sie können das Amazon S3-Objekt löschen, nachdem Lambda die Layer-Version erstellt hat.
- `REFERENCE`— Lambda verweist auf das Layer-Archiv aus Ihrem Amazon S3-Bucket und speichert keine Kopie. Das Objekt muss an Ort und Stelle bleiben, und Lambda muss während der gesamten Lebensdauer der Layer-Version darauf zugreifen können.

Um es zu verwenden`REFERENCE`, aktivieren Sie die Versionierung in Ihrem Amazon S3-Bucket und gewähren Sie dem Lambda-Serviceprinzipal Zugriff auf das Objekt. Weitere Informationen finden Sie im AWS Lambda Entwicklerhandbuch unter Self-managed Amazon S3-Codespeicher.

Typ: Zeichenfolge

Gültige Werte: `COPY` | `REFERENCE`

Required: No

Standardwert: `COPY`

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `S3ObjectStorageMode` Eigenschaft des `AWS::Lambda::LayerVersion` `Content` Datentyps übergeben.

`Version`

Für versionierte Objekte, die Version des zu verwendenden Layer-Archiv-Objekts.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `S3ObjectVersion` Eigenschaft des `AWS::Lambda::LayerVersion` `Content` Datentyps übergeben.

Beispiele

LayerContent

Beispiel für Ebeneninhalt

YAML

```
LayerContent:
  Bucket: amzn-s3-demo-bucket-name
  Key: mykey-name
  Version: 121212
```

Self-managed Amazon S3-Codespeicher

Im Folgenden wird `LayerContent` auf das Layer-Archiv aus Ihrem eigenen Amazon S3-Bucket verwiesen, anstatt es in den Lambda-managed Speicher zu kopieren.

YAML

```
LayerContent:
  Bucket: amzn-s3-demo-bucket-name
  Key: mykey-name
  Version: 121212
  StorageMode: REFERENCE
```

AWS::Serverless::MicrovmImage

Erstellt ein AWS Lambda MicroVM-Image aus einem in Amazon S3 gespeicherten Codeartefakt. Ein MicroVM-Image ist ein Firecracker Snapshot, der aus einem Anwendungscode erstellt wird Dockerfile und den Start von MicroVM-Instanzen in Sekundenbruchteilen ermöglicht.

> **Note**
>
> Bei der Bereitstellung auf werden Ihre AWS CloudFormation AWS SAM Ressourcen in AWS SAM Ressourcen umgewandelt. CloudFormation Weitere Informationen finden Sie unter Generiert CloudFormation Ressourcen für AWS SAM.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
Type: AWS::Serverless::MicrovmImage
Properties:
  AdditionalOsCapabilities: List
  BaseImageArn: String
  BaseImageVersion: String
  BuildRoleArn: String
  CodeUri: String
  CpuConfigurations: List
  Description: String
  EgressNetworkConnectors: List
  EnvironmentVariables: Map
  Hooks: Hooks
  Logging: Logging
  Name: String
  PropagateTags: Boolean
  Resources: List
  Tags: Map
```

Eigenschaften

`AdditionalOsCapabilities`

Zusätzliche Betriebssystemfunktionen, die der MicroVM-Laufzeitumgebung gewährt werden sollen. Wenn Sie keinen Wert angeben, gewährt der Dienst keine zusätzlichen Betriebssystemfunktionen.

Gültige Werte: `ALL`

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `AdditionalOsCapabilities` Eigenschaft einer `AWS::Lambda::MicrovmImage` Ressource übergeben.

`BaseImageArn`

Der ARN des MicroVM-Basisimages, aus dem erstellt werden soll.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `BaseImageArn` Eigenschaft einer `AWS::Lambda::MicrovmImage` Ressource übergeben.

`BaseImageVersion`

Die Version des zu verwendenden MicroVM-Basisimages.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `BaseImageVersion` Eigenschaft einer `AWS::Lambda::MicrovmImage` Ressource übergeben.

`BuildRoleArn`

Der ARN der IAM-Rolle, die der MicroVM-Build-Service annimmt, um Ihr Code-Artefakt von Amazon S3 herunterzuladen und Build-Logs zu schreiben.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `BuildRoleArn` Eigenschaft einer Ressource. `AWS::Lambda::MicrovmImage` Dies ist in erforderlich CloudFormation , aber nicht in AWS SAM. Wenn Sie keine Rolle angeben, AWS SAM erstellt eine Rolle mit der logischen ID*<resource-logical-id>*`BuildRole`. Wenn Ihr Codeartefakt mit einem vom

Kunden verwalteten Schlüssel verschlüsselt ist AWS KMS, müssen Sie Ihrer eigenen Rolle die entsprechenden `kms:Decrypt` Berechtigungen zuweisen.

`CodeUri`

Die Amazon S3 S3-URI des ZIP-Artefakts, das den Dockerfile und den Anwendungscode enthält.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird in die `CodeArtifact` Eigenschaft einer `AWS::Lambda::MicrovmImage` Ressource umgewandelt. AWS SAM schließt den URI in die `CodeArtifact.Uri` Struktur ein.

`CpuConfigurations`

Eine Liste von CpuConfigurationObjekten, die die unterstützten CPU-Architekturen für die MicroVM spezifizieren. Wenn Sie keinen Wert angeben, verwendet der Dienst die Standard-CPU-Architektur.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `CpuConfigurations` Eigenschaft einer `AWS::Lambda::MicrovmImage` Ressource übergeben.

`Description`

Eine Beschreibung des MicroVM-Images.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Description` Eigenschaft einer `AWS::Lambda::MicrovmImage` Ressource übergeben.

`EgressNetworkConnectors`

Die Liste der ausgehenden Netzwerk-Connector-ARNs, die der microVM zur Laufzeit zur Verfügung stehen.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `EgressNetworkConnectors` Eigenschaft einer Ressource übergeben. `AWS::Lambda::MicrovmImage`

`EnvironmentVariables`

Umgebungsvariablen, die in der MicroVM-Laufzeitumgebung festgelegt sind.

Typ: Karte

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `EnvironmentVariables` Eigenschaft einer `AWS::Lambda::MicrovmImage` Ressource. Geben Sie in AWS SAM als Map an (z. B.`KEY: value`). AWS SAM konvertiert in das CloudFormation Array-Format`[{Key: KEY, Value: value}]`.

`Hooks`

Lifecycle-Hook-Konfiguration für MicroVMs und MicroVM-Images.

Typ: Hooks

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Hooks` Eigenschaft einer `AWS::Lambda::MicrovmImage` Ressource übergeben.

`Logging`

Konfiguration für die MicroVM-Logging-Ausgabe. Geben Sie genau eine an: `CloudWatch` um die CloudWatch Protokollierung zu aktivieren oder `Disabled` um die Protokollierung zu deaktivieren.

Typ: Protokollierung

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Logging` Eigenschaft einer `AWS::Lambda::MicrovmImage` Ressource übergeben.

`Name`

Der Name des MicroVM-Images.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Name` Eigenschaft einer `AWS::Lambda::MicrovmImage` Ressource übergeben.

`PropagateTags`

Gibt an, ob Tags von der `Tags` Eigenschaft an generierte Ressourcen übergeben werden sollen, z. B. an die automatisch generierte Build-Rolle.

Typ: Boolesch

Required: No

Standardwert: `False`

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Resources`

Eine Liste von Resources-Objekten, die die Ressourcenanforderungen für die MicroVM spezifizieren.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Resources` Eigenschaft einer `AWS::Lambda::MicrovmImage` Ressource übergeben.

`Tags`

Eine Zuordnung von Schlüssel-Wert-Paaren, die die Tags angibt, die diesem MicroVM-Image hinzugefügt wurden.

Typ: Karte

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `Tags` Eigenschaft einer `AWS::Lambda::MicrovmImage` Ressource. Die `Tags` Eigenschaft in AWS SAM besteht aus Schlüssel-Wert-Paaren (wohingegen CloudFormation diese Eigenschaft aus einer Liste von `Tag` Objekten besteht). Fügt diesem MicroVM-Image und den dafür generierten Standardrollen außerdem AWS SAM automatisch ein `lambda:createdBy:SAM` Tag hinzu.

Rückgabewerte

Punkt

Wenn die logische ID dieser Ressource für die `Ref` intrinsische Funktion bereitgestellt wird, gibt sie den ARN des MicroVM-Images zurück.

Weitere Informationen zur Verwendung der *`Ref`* Funktion finden Sie *`Ref`*im AWS CloudFormation Benutzerhandbuch.

Fn:: GetAtt

`Fn::GetAtt` gibt einen Wert für ein angegebenes Attribut dieses Typs zurück. Im Folgenden sehen Sie die verfügbaren Attribute und Beispielrückgabewerte.

Weitere Informationen zur Verwendung `Fn::GetAtt` finden Sie `Fn::GetAtt`im AWS CloudFormation Benutzerhandbuch.

`CreatedAt`

Der Zeitstempel, als die MicroVM-Image-Version erstellt wurde.

`ImageArn`

Der ARN des MicroVM-Images.

`LatestActiveImageVersion`

Die neueste aktive Version des MicroVM-Images.

`LatestFailedImageVersion`

Die letzte fehlgeschlagene Version des MicroVM-Images, falls vorhanden.

`State`

Der aktuelle Status des MicroVM-Images.

`UpdatedAt`

Der Zeitstempel, zu dem die MicroVM-Image-Version zuletzt aktualisiert wurde.

Beispiele

Minimales MicroVM-Image

Im folgenden Beispiel wird ein MicroVM-Image erstellt, das nur die erforderlichen Eigenschaften enthält. AWS SAM generiert automatisch eine Build-Rolle.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31

Resources:
  MyMicroVM:
    Type: AWS::Serverless::MicrovmImage
    Properties:
      Name: my-app
      CodeUri: s3://my-bucket/app.zip
      BaseImageArn: arn:aws:lambda:us-east-2:aws:microvm-image:al2023-1
      BaseImageVersion: "0"
```

Vollständige MicroVM-Image-Konfiguration

Im folgenden Beispiel wird ein MicroVM-Image erstellt, bei dem alle verfügbaren Eigenschaften konfiguriert sind.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31

Resources:
  MyMicroVM:
    Type: AWS::Serverless::MicrovmImage
    Properties:
      Name: my-app
      CodeUri: s3://my-bucket/app.zip
      BaseImageArn: arn:aws:lambda:us-east-2:aws:microvm-image:al2023-1
      BaseImageVersion: "0"
      Description: "Production MicroVM application"
      CpuConfigurations:
```

```
        - Architecture: ARM_64
      Resources:
        - MinimumMemoryInMiB: 1024
      AdditionalOsCapabilities:
        - ALL
      EgressNetworkConnectors:
        - arn:aws:lambda:us-east-2:aws:network-connector:aws-network-
connector:INTERNET_EGRESS
      Hooks:
        Port: 9000
        MicrovmHooks:
          Run: ENABLED
          RunTimeoutInSeconds: 30
          Suspend: ENABLED
          Resume: ENABLED
        MicrovmImageHooks:
          Ready: ENABLED
          ReadyTimeoutInSeconds: 60
      EnvironmentVariables:
        LOG_LEVEL: info
        APP_ENV: production
      Logging:
        CloudWatch:
          LogGroup: /aws/lambda-microvms/my-app
      Tags:
        Environment: Production
        Team: Platform
```

AWS::Serverless::NetworkConnector

Erstellt einen AWS Lambda Network Connector, der die Netzwerkkonnektivität für Lambda-Rechenressourcen definiert. Ein Netzwerkconnector gibt an, wie Rechenressourcen mit Netzwerken verbunden werden. Beispielsweise wird ausgehender Datenverkehr über eine Kunden-VPC mithilfe von Elastic Network Interfaces (ENIs) weitergeleitet, die der Dienst in bestimmten Subnetzen und Sicherheitsgruppen bereitstellt.

> **Note**
>
> Bei der Bereitstellung werden Ihre Ressourcen AWS CloudFormation in AWS SAM Ressourcen umgewandelt. AWS SAM CloudFormation Weitere Informationen finden Sie unter Generiert CloudFormation Ressourcen für AWS SAM.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
Type: AWS::Serverless::NetworkConnector
Properties:
  Name: String
  OperatorRole: String
  PropagateTags: Boolean
  Tags: Map
  VpcConfig: VpcConfig
```

Eigenschaften

`Name`

Der Name des Netzwerk-Connectors. Der Name muss innerhalb Ihres Kontos und Ihrer Region eindeutig sein.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Name` Eigenschaft einer `AWS::Lambda::NetworkConnector` Ressource übergeben.

`OperatorRole`

Der ARN der IAM-Rolle, die der Network Connector-Service für die Erstellung und Verwaltung von Elastic Network Interfaces (ENIs) annimmt.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `OperatorRole` Eigenschaft einer `AWS::Lambda::NetworkConnector` Ressource. Dies ist in erforderlich CloudFormation , aber nicht in AWS SAM. Wenn Sie keine Rolle angeben, AWS SAM erstellt eine Rolle mit der logischen ID`<resource-logical-id>OperatorRole`.

`PropagateTags`

Gibt an, ob Tags von der `Tags` Eigenschaft an generierte Ressourcen übergeben werden sollen, z. B. an die automatisch generierte Operatorrolle.

Typ: Boolesch

Required: No

Standardwert: `False`

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Tags`

Eine Zuordnung von Schlüssel-Wert-Paaren, die die Tags angibt, die diesem Netzwerkconnector hinzugefügt wurden.

Typ: Karte

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `Tags` Eigenschaft einer `AWS::Lambda::NetworkConnector` Ressource. Die `Tags` Eigenschaft in AWS SAM besteht aus Schlüssel-Wert-Paaren (wohingegen CloudFormation diese Eigenschaft aus einer Liste von `Tag` Objekten besteht). Fügt diesem Netzwerkconnector und den Standardrollen, die für ihn generiert wurden, außerdem AWS SAM automatisch ein `lambda:createdBy:SAM` Tag hinzu.

`VpcConfig`

Die VPC-Ausgangskonfiguration für den Netzwerkconnector. Gibt die Subnetze, Sicherheitsgruppen und das Netzwerkprotokoll für das Routing von ausgehendem Datenverkehr über Ihre VPC an.

Typ: VpcConfig

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird in die `Configuration.VpcEgressConfiguration` Eigenschaft einer Ressource

umgewandelt. `AWS::Lambda::NetworkConnector` AWS SAM injiziert auch `AssociatedComputeResourceTypes: [MicroVm]` automatisch.

Rückgabewerte

Punkt

Wenn die logische ID dieser Ressource für die `Ref` intrinsische Funktion bereitgestellt wird, gibt sie den ARN des Netzwerkkonnektors zurück.

Weitere Informationen zur Verwendung der `Ref` Funktion finden Sie Refim AWS CloudFormation Benutzerhandbuch.

Fn:: GetAtt

`Fn::GetAtt` gibt einen Wert für ein angegebenes Attribut dieses Typs zurück. Im Folgenden sehen Sie die verfügbaren Attribute und Beispielrückgabewerte.

Weitere Informationen zur Verwendung `Fn::GetAtt` finden Sie Fn::GetAttim AWS CloudFormation Benutzerhandbuch.

`Arn`

Der Amazon-Ressourcenname (ARN) des Netzwerkkonnektors.

`State`

Der aktuelle Status des Netzwerkkonnektors. Gültige Werte sind `PENDINGACTIVE`,`INACTIVE`,`FAILED`,`DELETING`, und`DELETE_FAILED`.

Beispiele

Einfacher Netzwerkkonnektor mit automatisch generierter Rolle

Im folgenden Beispiel wird ein VPC-Ausgangsnetzwerkconnector erstellt. AWS SAM generiert automatisch die Operatorrolle.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
```

```
Resources:
  MyVpcConnector:
    Type: AWS::Serverless::NetworkConnector
    Properties:
      Name: my-vpc-connector
      VpcConfig:
        SubnetIds:
          - subnet-0abc123def4567890
          - subnet-0def456abc7890123
        SecurityGroupIds:
          - sg-0abc1234abcde0001
        NetworkProtocol: IPv4
      Tags:
        Environment: Production
```

VpcConfig

Konfiguriert die VPC-Ausgangseinstellungen für einen Netzwerkconnector, einschließlich der Subnetze, Sicherheitsgruppen und des Netzwerkprotokolls für die Bereitstellung von elastic network interface (ENI).

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer AWS Serverless Application Model (AWS SAM) -Vorlage zu deklarieren.

YAML

```
NetworkProtocol: String
SecurityGroupIds: List
SubnetIds: List
```

Eigenschaften

`NetworkProtocol`

Das Netzwerkprotokoll für die VPC-Ausgangsverbindung.

Zulässige Werte: `IPv4`, `DualStack`

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `NetworkProtocol` Eigenschaft `Configuration.VpcEgressConfiguration` einer `AWS::Lambda::NetworkConnector` Ressource übergeben.

`SecurityGroupIds`

Eine Liste von Sicherheitsgruppen-IDs, die den Elastic Network Interfaces (ENIs) zugeordnet werden sollen.

Typ: Liste

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `SecurityGroupIds` Eigenschaft `Configuration.VpcEgressConfiguration` einer `AWS::Lambda::NetworkConnector` Ressource übergeben.

`SubnetIds`

Eine Liste von Subnetz-IDs, für die der Dienst ENIs für die VPC-Ausgangskonnektivität bereitstellt.

Typ: Liste

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `SubnetIds` Eigenschaft einer `Configuration.VpcEgressConfiguration` Ressource übergeben. `AWS::Lambda::NetworkConnector`

Beispiele

VpcConfig Beispiel

```
VpcConfig:
  SubnetIds:
    - subnet-0abc123def4567890
    - subnet-0def456abc7890123
  SecurityGroupIds:
    - sg-0abc1234def567890
```

```
  NetworkProtocol: IPv4
```

AWS::Serverless::SimpleTable

Erstellt eine DynamoDB-Tabelle mit einem einzelnen Attribut-Primärschlüssel. Dies ist nützlich, wenn auf Daten nur über einen Primärschlüssel zugegriffen werden muss.

Für erweiterte Funktionen verwenden Sie eine AWS::DynamoDB::TableRessource in CloudFormation. Diese Ressourcen können in verwendet werden. AWS SAM Sie sind umfassend und ermöglichen weitere Anpassungen, einschließlich key schemaAnpassungen. resource policy

> **Note**
>
> Bei der Bereitstellung von AWS CloudFormation werden Ihre AWS SAM Ressourcen in AWS SAM CloudFormation Ressourcen umgewandelt. Weitere Informationen finden Sie unter Generiert CloudFormation Ressourcen für AWS SAM.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
Type: AWS::Serverless::SimpleTable
Properties:
  PointInTimeRecoverySpecification: PointInTimeRecoverySpecification
  PrimaryKey: PrimaryKeyObject
  ProvisionedThroughput: ProvisionedThroughputObject
  SSESpecification: SSESpecification
  TableName: String
  Tags: Map
```

Eigenschaften

`PointInTimeRecoverySpecification`

Die Einstellungen, die zum Aktivieren der zeitpunktbezogenen Wiederherstellung verwendet werden.

Geben Sie ein: PointInTimeRecoverySpecification

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `PointInTimeRecoverySpecification` Eigenschaft einer `AWS::DynamoDB::Table` Ressource übergeben.

`PrimaryKey`

Name und Typ des Attributs, die als Primärschlüssel der Tabelle verwendet werden sollen. Wenn nicht angegeben, ist der Primärschlüssel a `String` mit dem Wert`id`.

> **Note**
>
> Der Wert dieser Eigenschaft kann nicht geändert werden, nachdem diese Ressource erstellt wurde.

Typ: PrimaryKeyObject

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`ProvisionedThroughput`

Lesen und Schreiben von Informationen zur Durchsatzbereitstellung.

Falls `ProvisionedThroughput` nicht angegeben, `BillingMode` wird als `PAY_PER_REQUEST` angegeben.

Typ: ProvisionedThroughputObject

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `ProvisionedThroughput` Eigenschaft einer `AWS::DynamoDB::Table` Ressource übergeben.

`SSESpecification`

Gibt die Einstellungen zum Aktivieren der serverseitigen Verschlüsselung an.

Typ: SSESpecification

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `SSESpecification` Eigenschaft einer `AWS::DynamoDB::Table` Ressource übergeben.

`TableName`

Name für die DynamoDB-Tabelle.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `TableName` Eigenschaft einer `AWS::DynamoDB::Table` Ressource übergeben.

`Tags`

Eine Zuordnung (von Zeichenfolge zu Zeichenfolge), die die Tags angibt, die dieser hinzugefügt werden sollen SimpleTable. Einzelheiten zu gültigen Schlüsseln und Werten für Tags finden Sie unter Resource-Tag im AWS CloudFormation Benutzerhandbuch.

Typ: Karte

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `Tags` Eigenschaft einer `AWS::DynamoDB::Table` Ressource. Die Tags-Eigenschaft in SAM besteht aus Key:Value-Paaren; darin besteht CloudFormation sie aus einer Liste von Tag-Objekten.

Rückgabewerte

Punkt

Wenn die logische ID dieser Ressource für die intrinsische Funktion Ref bereitgestellt wird, gibt sie den Ressourcennamen der zugrunde liegenden DynamoDB-Tabelle zurück.

Weitere Informationen zur Verwendung der *`Ref`* Funktion finden Sie *`Ref`*im Benutzerhandbuch.AWS CloudFormation

Beispiele

SimpleTableExample

Beispiel für ein SimpleTable

YAML

```
Properties:
  TableName: my-table
  PrimaryKey:
    Name: MyPrimaryKey
    Type: String
  ProvisionedThroughput:
    ReadCapacityUnits: 5
    WriteCapacityUnits: 5
  Tags:
    Department: Engineering
    AppType: Serverless
```

PrimaryKeyObject

Das Objekt, das die Eigenschaften eines Primärschlüssels beschreibt.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
  Name: String
  Type: String
```

Eigenschaften

`Name`

Attributname des Primärschlüssels.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `AttributeName` Eigenschaft des `AWS::DynamoDB::Table` `AttributeDefinition` Datentyps übergeben.

Zusätzliche Hinweise: Diese Eigenschaft wird auch an die AttributeNameEigenschaft eines `AWS::DynamoDB::Table KeySchema` Datentyps übergeben.

`Type`

Der Datentyp für den Primärschlüssel.

Gültige Werte:`String`,`Number`, `Binary`

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `AttributeType` Eigenschaft des `AWS::DynamoDB::Table` `AttributeDefinition` Datentyps übergeben.

Beispiele

PrimaryKey

Beispiel für einen Primärschlüssel.

YAML

```
Properties:
  PrimaryKey:
    Name: MyPrimaryKey
    Type: String
```

ProvisionedThroughputObject

Das Objekt, das die Eigenschaften eines bereitgestellten Durchsatzes beschreibt.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
  ReadCapacityUnits: Integer
  WriteCapacityUnits: Integer
```

Eigenschaften

`ReadCapacityUnits`

Die maximale Anzahl der Strongly Consistent-Lesevorgänge, die pro Sekunde verbraucht werden, bevor DynamoDB eine `ThrottlingException` zurückgibt.

Typ: Ganzzahl

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `ReadCapacityUnits` Eigenschaft des `AWS::DynamoDB::Table` `ProvisionedThroughput` Datentyps übergeben.

`WriteCapacityUnits`

Die maximale Anzahl der Schreibvorgänge, die pro Sekunde verbraucht werden, bevor DynamoDB eine `ThrottlingException` zurückgibt.

Typ: Ganzzahl

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `WriteCapacityUnits` Eigenschaft des `AWS::DynamoDB::Table` `ProvisionedThroughput` Datentyps übergeben.

Beispiele

ProvisionedThroughput

Beispiel für einen bereitgestellten Durchsatz.

YAML

```
Properties:
   ProvisionedThroughput:
     ReadCapacityUnits: 5
     WriteCapacityUnits: 5
```

AWS::Serverless::StateMachine

Erstellt eine AWS Step Functions Zustandsmaschine, mit der Sie AWS Lambda Funktionen und andere AWS Ressourcen orchestrieren können, um komplexe und robuste Workflows zu bilden.

Weitere Informationen zu Step Functions finden Sie im AWS Step Functions Developer Guide.

> **Note**
>
> Bei der Bereitstellung werden AWS CloudFormation Ihre AWS SAM Ressourcen in Ressourcen umgewandelt CloudFormation . AWS SAM Weitere Informationen finden Sie unter Generiert CloudFormation Ressourcen für AWS SAM.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer AWS Serverless Application Model (AWS SAM) -Vorlage zu deklarieren.

YAML

```
Type: AWS::Serverless::StateMachine
Properties:
  AutoPublishAlias: String
  UseAliasAsEventTarget: Boolean
  Definition: Map
  DefinitionSubstitutions: Map
  DefinitionUri: String | S3Location
  DeploymentPreference: DeploymentPreference
  Events: EventSource
  Logging: LoggingConfiguration
  Name: String
  PermissionsBoundary: String
  Policies: String | List | Map
  PropagateTags: Boolean
  RolePath: String
  Role: String
  Tags: Map
  Tracing: TracingConfiguration
  Type: String
```

Eigenschaften

`AutoPublishAlias`

Der Name des Alias für die Zustandsmaschine. Weitere Informationen zur Verwendung von Step Functions Functions-Zustandsmaschinenaliasen finden Sie unter Manage Continuous Deployments with versions and aliases im AWS Step Functions Developer Guide.

Wird verwendet`DeploymentPreference`, um die Bereitstellungseinstellungen für Ihren Alias zu konfigurieren. Wenn Sie nichts angeben`DeploymentPreference`, AWS SAM wird der Datenverkehr so konfiguriert, dass er auf einmal zur neueren State-Machine-Version wechselt.

AWS SAM legt `Retain` standardmäßig die Version `DeletionPolicy` und `UpdateReplacePolicy` auf fest. Frühere Versionen werden nicht automatisch gelöscht.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Name` Eigenschaft einer `AWS::StepFunctions::StateMachineAlias` Ressource übergeben.

`UseAliasAsEventTarget`

Geben Sie an, ob der mit der `AutoPublishAlias` Eigenschaft erstellte Alias an das mit Events definierte Ziel der Ereignisquelle übergeben werden soll oder nicht.

Geben Sie `True` an, dass der Alias als Ziel der Ereignisse verwendet werden soll.

Typ: Boolesch

Required: No

Standardwert: `False`

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Definition`

Die State-Machine-Definition ist ein Objekt, bei dem das Format des Objekts dem Format Ihrer AWS SAM Vorlagendatei entspricht, z. B. JSON oder YAML. State Machine-Definitionen halten sich an die Amazon States Language.

Ein Beispiel für eine Inline-State-Machine-Definition finden Sie unterBeispiele.

Sie müssen entweder a `Definition` oder a angeben`DefinitionUri`.

Typ: Karte

Required: Conditional

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`DefinitionSubstitutions`

Eine string-to-string Map, die die Zuordnungen für Platzhaltervariablen in der State-Machine-Definition spezifiziert. Auf diese Weise können Sie Werte, die Sie zur Laufzeit erhalten haben (z. B. aus systeminternen Funktionen), in die Zustandsmaschinen-Definition einfügen.

Typ: Karte

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `DefinitionSubstitutions` Eigenschaft einer `AWS::StepFunctions::StateMachine` Ressource. Wenn in einer Inline-State-Machine-Definition irgendwelche systemeigenen Funktionen angegeben sind, AWS SAM fügt dieser Eigenschaft Einträge hinzu, um sie in die Zustandsmaschinen-Definition einzufügen.

`DefinitionUri`

Die Amazon Simple Storage Service (Amazon S3) -URI oder der lokale Dateipfad der Zustandsmaschinen-Definition, geschrieben in der Sprache Amazon States.

Wenn Sie einen lokalen Dateipfad angeben, muss die Vorlage den Workflow durchlaufen, der den `sam package` Befehl `sam deploy` or enthält, um die Definition korrekt zu transformieren. Dazu müssen Sie Version 0.52.0 oder höher der AWS SAM CLI verwenden.

Sie müssen entweder a `Definition` oder a `DefinitionUri` angeben.

Typ: Zeichenfolge | S3Location

Required: Conditional

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `DefinitionS3Location` Eigenschaft einer `AWS::StepFunctions::StateMachine` Ressource übergeben.

`DeploymentPreference`

Die Einstellungen, die die schrittweise Bereitstellung von Zustandsmaschinen ermöglichen und konfigurieren. Weitere Informationen zu schrittweisen Bereitstellungen von Step Functions finden Sie unter Manage Continuous Deployments with versions and aliases im AWS Step Functions Developer Guide.

Geben Sie dies an, `AutoPublishAlias` bevor Sie diese Eigenschaft konfigurieren. Ihre `DeploymentPreference` Einstellungen werden auf den mit angegebenen Alias angewendet`AutoPublishAlias`.

Wenn Sie angeben`DeploymentPreference`, AWS SAM wird der Wert der `StateMachineVersionArn` Untereigenschaft automatisch generiert.

Typ: DeploymentPreference

Required: No

CloudFormation Kompatibilität: AWS SAM generiert den `StateMachineVersionArn` Eigenschaftswert, hängt ihn an `DeploymentPreference` und leitet ihn `DeploymentPreference` an die `DeploymentPreference` Eigenschaft einer `AWS::StepFunctions::StateMachineAlias` Ressource weiter.

`Events`

Gibt die Ereignisse an, die diese Zustandsmaschine auslösen. Ereignisse bestehen aus einem Typ und einer Reihe von Eigenschaften, die vom Typ abhängen.

Typ: EventSource

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Logging`

Definiert, welche Ereignisse im Ausführungsverlauf protokolliert werden und wo sie protokolliert werden.

Typ: LoggingConfiguration

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `LoggingConfiguration` Eigenschaft einer `AWS::StepFunctions::StateMachine` Ressource übergeben.

`Name`

Der Name des Zustandsautomaten.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `StateMachineName` Eigenschaft einer `AWS::StepFunctions::StateMachine` Ressource übergeben.

`PermissionsBoundary`

Der ARN einer Berechtigungsgrenze, die für die Ausführungsrolle dieser Zustandsmaschine verwendet werden soll. Diese Eigenschaft funktioniert nur, wenn die Rolle für Sie generiert wurde.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `PermissionsBoundary` Eigenschaft einer `AWS::IAM::Role` Ressource übergeben.

`Policies`

Berechtigungsrichtlinien für diesen Zustandsmaschine. Richtlinien werden an die standardmäßige Ausführungsrolle AWS Identity and Access Management (IAM) des Zustandsmaschinen angehängt.

Diese Eigenschaft akzeptiert einen einzelnen Wert oder eine Liste von Werten. Gültige Werte sind:

- AWS SAM Richtlinienvorlagen.
- Die ARN einer AWS verwalteten Richtlinie oder einer vom Kunden verwalteten Richtlinie.
- Der Name einer AWS verwalteten Richtlinie aus der folgenden Liste.
- Eine Inline-IAM-Richtlinie, die YAML als Map formatiert ist.

> **Note**
>
> Wenn Sie die `Role` Eigenschaft festlegen, wird diese Eigenschaft ignoriert.

Typ: Zeichenfolge | Liste | Karte

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`PropagateTags`

Geben Sie an, ob Tags von der `Tags` Eigenschaft an Ihre AWS::Serverless::StateMachine generierten Ressourcen weitergegeben werden sollen oder nicht. Geben Sie `True` an, dass Tags in Ihren generierten Ressourcen verbreitet werden sollen.

Typ: Boolesch

Required: No

Standardwert: `False`

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Role`

Der ARN einer IAM-Rolle, die als Ausführungsrolle dieser Zustandsmaschine verwendet werden soll.

Typ: Zeichenfolge

Required: Conditional

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `RoleArn` Eigenschaft einer `AWS::StepFunctions::StateMachine` Ressource übergeben.

`RolePath`

Der Pfad zur IAM-Ausführungsrolle der Zustandsmaschine.

Verwenden Sie diese Eigenschaft, wenn die Rolle für Sie generiert wird. Nicht verwenden, wenn die Rolle mit der `Role` Eigenschaft angegeben ist.

Typ: Zeichenfolge

Required: Conditional

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Path` Eigenschaft einer `AWS::IAM::Role` Ressource übergeben.

`Tags`

Eine string-to-string Map, die die zur Zustandsmaschine hinzugefügten Tags und die entsprechende Ausführungsrolle angibt. Informationen zu gültigen Schlüsseln und Werten für Tags finden Sie in der Tags-Eigenschaft einer AWS::StepFunctions::StateMachineRessource.

Typ: Karte

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `Tags` Eigenschaft einer `AWS::StepFunctions::StateMachine` Ressource. AWS SAM fügt dieser Ressource und der dafür generierten Standardrolle automatisch ein `stateMachine:createdBy:SAM` Tag hinzu.

`Tracing`

Wählt aus, ob sie für die Zustandsmaschine aktiviert AWS X-Ray ist oder nicht. Weitere Informationen zur Verwendung von X-Ray mit Step Functions finden Sie unter AWS X-Ray und Step Functions im AWS Step Functions Entwicklerhandbuch.

Typ: TracingConfiguration

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `TracingConfiguration` Eigenschaft einer `AWS::StepFunctions::StateMachine` Ressource übergeben.

`Type`

Der Typ der Zustandsmaschine.

Zulässige Werte: `STANDARD` oder `EXPRESS`.

Typ: Zeichenfolge

Required: No

Standardwert: `STANDARD`

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `StateMachineType` Eigenschaft einer `AWS::StepFunctions::StateMachine` Ressource übergeben.

Rückgabewerte

Punkt

Wenn Sie die logische ID dieser Ressource für die intrinsische Funktion Ref angeben, gibt Ref den Amazon-Ressourcennamen (ARN) der zugrunde liegenden `AWS::StepFunctions::StateMachine` Ressource zurück.

Weitere Informationen zur Verwendung der `Ref` Funktion finden Sie Refim AWS CloudFormation Benutzerhandbuch.

Fn:: GetAtt

`Fn::GetAtt` gibt einen Wert für ein angegebenes Attribut dieses Typs zurück. Im Folgenden sehen Sie die verfügbaren Attribute und Beispielrückgabewerte.

Weitere Informationen zur Verwendung `Fn::GetAtt` finden Sie Fn::GetAttim AWS CloudFormation Benutzerhandbuch.

`Name`

Gibt den Namen der Zustandsmaschine zurück, z. `HelloWorld-StateMachine` B.

Beispiele

Definitionsdatei für Zustandsmaschinen

Im Folgenden finden Sie ein Beispiel für eine Inline-State-Machine-Definition, die es einer Lambda-Funktion ermöglicht, eine Zustandsmaschine aufzurufen. Beachten Sie, dass in diesem Beispiel davon ausgegangen wird, dass die `Role` Eigenschaft die richtige Richtlinie konfiguriert, um den Aufruf zu ermöglichen. Die `my_state_machine.asl.json` Datei muss in der Sprache der Amazonas-Staaten verfasst sein.

In diesem Beispiel ermöglichen die `DefinitionSubstitution` Einträge der Zustandsmaschine, Ressourcen einzubeziehen, die in der AWS SAM Vorlagendatei deklariert sind.

YAML

```
MySampleStateMachine:
  Type: AWS::Serverless::StateMachine
  Properties:
    DefinitionUri: statemachine/my_state_machine.asl.json
```

```
    Role: arn:aws:iam::123456123456:role/service-role/my-sample-role
    Tracing:
      Enabled: true
    DefinitionSubstitutions:
      MyFunctionArn: !GetAtt MyFunction.Arn
      MyDDBTable: !Ref TransactionTable
```

Definition von Inline-Zustandsmaschinen

Im Folgenden finden Sie ein Beispiel für eine Inline-State-Machine-Definition.

In diesem Beispiel ist die AWS SAM Vorlagendatei in YAML geschrieben, sodass die Zustandsmaschinen-Definition auch in YAML geschrieben ist. Um eine Inline-State-Machine-Definition in JSON zu deklarieren, schreiben Sie Ihre AWS SAM Vorlagendatei in JSON.

YAML

```
MySampleStateMachine:
  Type: AWS::Serverless::StateMachine
  Properties:
    Definition:
      StartAt: MyLambdaState
      States:
        MyLambdaState:
          Type: Task
          Resource: arn:aws:lambda:us-east-1:123456123456:function:my-sample-lambda-app
          End: true
    Role: arn:aws:iam::123456123456:role/service-role/my-sample-role
    Tracing:
      Enabled: true
```

EventSource

Das Objekt, das die Quelle der Ereignisse beschreibt, die die Zustandsmaschine auslösen. Jedes Ereignis besteht aus einem Typ und einer Reihe von Eigenschaften, die von diesem Typ abhängen. Weitere Informationen zu den Eigenschaften der einzelnen Ereignisquellen finden Sie in dem Unterthema zum jeweiligen Typ.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
Properties: Schedule | ScheduleV2 | CloudWatchEvent | EventBridgeRule | Api
Type: String
```

Eigenschaften

`Properties`

Ein Objekt, das die Eigenschaften dieser Event-Mapping beschreibt. Der Satz von Eigenschaften muss den definierten `Type` entsprechen.

Typ: Schedule | ScheduleV2 | | | Api CloudWatchEventEventBridgeRule

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Type`

Der Ereignistyp.

Gültige Werte:`Api`,`Schedule`,`ScheduleV2`,`CloudWatchEvent`, `EventBridgeRule`

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

API

Das Folgende ist ein Beispiel für ein Ereignis dieses `API` Typs.

YAML

```
ApiEvent:
  Type: Api
```

```
  Properties:
    Method: get
    Path: /group/{user}
    RestApiId:
      Ref: MyApi
```

Api

Das Objekt, das einen `Api` Ereignisquellentyp beschreibt. Wenn eine AWS::Serverless::Api Ressource definiert ist, müssen der Pfad und die Methodenwerte einer Operation in der OpenAPI-Definition der API entsprechen.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
Auth: ApiStateMachineAuth
Method: String
Path: String
RestApiId: String
UnescapeMappingTemplate: Boolean
```

Eigenschaften

`Auth`

Die Autorisierungskonfiguration für diese API, diesen Pfad und diese Methode.

Verwenden Sie diese Eigenschaft, um die `DefaultAuthorizer` API-Einstellung für einen einzelnen Pfad zu überschreiben, wenn kein Pfad angegeben `DefaultAuthorizer` ist, oder um die `ApiKeyRequired` Standardeinstellung zu überschreiben.

Typ: ApiStateMachineAuth

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Method`

Die HTTP-Methode, für die diese Funktion aufgerufen wird.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Path`

Der URI-Pfad, für den diese Funktion aufgerufen wird. Der Wert muss mit `/` beginnen.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`RestApiId`

Der Bezeichner einer `RestApi` Ressource, die eine Operation mit dem angegebenen Pfad und der angegebenen Methode enthalten muss. In der Regel ist dies so eingestellt, dass es auf eine AWS::Serverless::Api Ressource verweist, die in dieser Vorlage definiert ist.

Wenn Sie diese Eigenschaft nicht definieren, AWS SAM wird mithilfe eines generierten `OpenApi` Dokuments eine AWS::Serverless::Api Standardressource erstellt. Diese Ressource enthält eine Vereinigung aller Pfade und Methoden, die durch `Api` Ereignisse in derselben Vorlage definiert wurden, ohne dass a angegeben ist`RestApiId`.

Diese Eigenschaft kann nicht auf eine AWS::Serverless::Api Ressource verweisen, die in einer anderen Vorlage definiert ist.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`UnescapeMappingTemplate`

Macht einfache Anführungszeichen rückgängig, indem sie `\'` bei der Eingabe`'`, die an die Zustandsmaschine übergeben wird, durch ersetzt werden. Wird verwendet, wenn Ihre Eingabe einfache Anführungszeichen enthält.

> **Note**
>
> Wenn dieser Wert auf gesetzt ist `False` und Ihre Eingabe einfache Anführungszeichen enthält, tritt ein Fehler auf.

Typ: Boolesch

Required: No

Standard: False

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

ApiEvent

Das Folgende ist ein Beispiel für ein Ereignis dieses `Api` Typs.

YAML

```
Events:
  ApiEvent:
    Type: Api
    Properties:
      Path: /path
      Method: get
```

ApiStateMachineAuth

Konfiguriert die Autorisierung auf Ereignisebene für eine bestimmte API, einen bestimmten Pfad und eine Methode.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer AWS Serverless Application Model (AWS SAM) -Vorlage zu deklarieren.

YAML

```
ApiKeyRequired: Boolean
AuthorizationScopes: List
Authorizer: String
ResourcePolicy: ResourcePolicyStatement
```

Eigenschaften

`ApiKeyRequired`

Erfordert einen API-Schlüssel für diese API, diesen Pfad und diese Methode.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`AuthorizationScopes`

Die Autorisierungsbereiche, die für diese API, diesen Pfad und diese Methode gelten sollen.

Die von Ihnen angegebenen Bereiche haben Vorrang vor allen Bereichen, die von der `DefaultAuthorizer` Eigenschaft angewendet werden, sofern Sie sie angegeben haben.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Authorizer`

Die `Authorizer` für eine bestimmte Zustandsmaschine.

Wenn Sie einen globalen Autorisierer für die API angegeben haben und diesen Zustandsmaschine öffentlich machen möchten, überschreiben Sie den globalen Autorisierer, indem Sie auf setzen`Authorizer`. `NONE`

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`ResourcePolicy`

Konfigurieren Sie die Ressourcenrichtlinie für diese API und diesen Pfad.

Typ: ResourcePolicyStatement

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

StateMachine-Authentifizierung

Das folgende Beispiel spezifiziert die Autorisierung auf State-Machine-Ebene.

YAML

```
Auth:
  ApiKeyRequired: true
  Authorizer: NONE
```

ResourcePolicyStatement

Konfiguriert eine Ressourcenrichtlinie für alle Methoden und Pfade einer API. Weitere Informationen zu Ressourcenrichtlinien finden Sie unter Steuern des Zugriffs auf eine API mit API-Gateway-Ressourcenrichtlinien im API Gateway Developer Guide.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
  AwsAccountBlacklist: List
  AwsAccountWhitelist: List
  CustomStatements: List
  IntrinsicVpcBlacklist: List
  IntrinsicVpcWhitelist: List
  IntrinsicVpceBlacklist: List
  IntrinsicVpceWhitelist: List
  IpRangeBlacklist: List
  IpRangeWhitelist: List
  SourceVpcBlacklist: List
  SourceVpcWhitelist: List
```

Eigenschaften

`AwsAccountBlacklist`

Die zu sperrenden AWS Konten.

Typ: Liste von Zeichenfolgen

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig AWS SAM und hat kein CloudFormation Äquivalent.

`AwsAccountWhitelist`

Die AWS Konten, die zugelassen werden sollen. Ein Beispiel für die Verwendung dieser Eigenschaft finden Sie im Abschnitt Beispiele unten auf dieser Seite.

Typ: Liste von Zeichenfolgen

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`CustomStatements`

Eine Liste von benutzerdefinierten Ressourcenrichtlinien-Anweisungen, die auf diese API angewendet werden sollen. Ein Beispiel für die Verwendung dieser Eigenschaft finden Sie im Abschnitt Beispiele unten auf dieser Seite.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`IntrinsicVpcBlacklist`

Die Liste der zu blockierenden virtuellen privaten Clouds (VPCs), wobei jede VPC als Referenz angegeben ist, z. B. als dynamische Referenz oder als `Ref` intrinsische Funktion. Ein Beispiel für die Verwendung dieser Eigenschaft finden Sie im Abschnitt Beispiele unten auf dieser Seite.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`IntrinsicVpcWhitelist`

Die Liste der VPCs Zulässigen, wobei jede VPC als Referenz angegeben wird, z. B. als dynamische Referenz oder als `Ref` intrinsische Funktion.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig AWS SAM und hat kein Äquivalent. CloudFormation

`IntrinsicVpceBlacklist`

Die Liste der zu blockierenden VPC-Endpunkte, wobei jeder VPC-Endpunkt als Referenz angegeben ist, z. B. als dynamische Referenz oder als intrinsische Funktion. `Ref`

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein Äquivalent. CloudFormation

`IntrinsicVpceWhitelist`

Die Liste der VPC-Endpunkte, die zugelassen werden sollen, wobei jeder VPC-Endpunkt als Referenz angegeben ist, z. B. als dynamische Referenz oder als systemeigene Funktion. `Ref` Ein Beispiel für die Verwendung dieser Eigenschaft finden Sie im Abschnitt Beispiele unten auf dieser Seite.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`IpRangeBlacklist`

Die IP-Adressen oder Adressbereiche, die blockiert werden sollen. Ein Beispiel für die Verwendung dieser Eigenschaft finden Sie im Abschnitt Beispiele unten auf dieser Seite.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`IpRangeWhitelist`

Die IP-Adressen oder Adressbereiche, die zugelassen werden sollen.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`SourceVpcBlacklist`

Die Quell-VPC oder die VPC-Endpoints, die blockiert werden sollen. Quell-VPC-Namen müssen mit beginnen `"vpc-"` und Quell-VPC-Endpunktnamen müssen mit beginnen. `"vpce-"` Ein Beispiel für die Verwendung dieser Eigenschaft finden Sie im Abschnitt Beispiele unten auf dieser Seite.

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`SourceVpcWhitelist`

Die Quell-VPC oder VPC-Endpoints, die zugelassen werden sollen. Quell-VPC-Namen müssen mit beginnen `"vpc-"` und Quell-VPC-Endpunktnamen müssen mit beginnen. `"vpce-"`

Typ: Liste

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

Beispiel für eine Ressourcenrichtlinie

Das folgende Beispiel blockiert zwei IP-Adressen und eine Quell-VPC und lässt ein AWS Konto zu.

YAML

```
Auth:
  ResourcePolicy:
    CustomStatements: [{
                        "Effect": "Allow",
                        "Principal": "*",
                        "Action": "execute-api:Invoke",
                        "Resource": "execute-api:/Prod/GET/pets",
                        "Condition": {
                          "IpAddress": {
                            "aws:SourceIp": "1.2.3.4"
                          }
                        }
                      }]
    IpRangeBlacklist:
      - "10.20.30.40"
      - "1.2.3.4"
    SourceVpcBlacklist:
      - "vpce-1a2b3c4d"
```

```
    AwsAccountWhitelist:
      - "111122223333"
    IntrinsicVpcBlacklist:
      - "{{resolve:ssm:SomeVPCReference:1}}"
      - !Ref MyVPC
    IntrinsicVpceWhitelist:
      - "{{resolve:ssm:SomeVPCEReference:1}}"
      - !Ref MyVPCE
```

CloudWatchEvent

Das Objekt, das einen `CloudWatchEvent` Ereignisquellentyp beschreibt.

AWS Serverless Application Model (AWS SAM) generiert eine AWS::Events::RuleRessource, wenn dieser Ereignistyp gesetzt ist.

Wichtiger Hinweis: EventBridgeRule ist der bevorzugte Typ der Ereignisquelle, der anstelle von verwendet wird`CloudWatchEvent`. `EventBridgeRule`und `CloudWatchEvent` verwenden Sie denselben zugrunde liegenden Dienst, dieselbe API und dieselben CloudFormation Ressourcen. AWS SAM Wird jedoch nur Unterstützung für neue Funktionen hinzufügen`EventBridgeRule`.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
EventBusName: String
Input: String
InputPath: String
Pattern: EventPattern
```

Eigenschaften

`EventBusName`

Der Ereignisbus, der dieser Regel zugeordnet werden soll. Wenn Sie diese Eigenschaft weglassen, AWS SAM wird der Standard-Event-Bus verwendet.

Typ: Zeichenfolge

Required: No

Standard: Standard-Event-Bus

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `EventBusName` Eigenschaft einer `AWS::Events::Rule` Ressource übergeben.

`Input`

Gültiger JSON-Text wurde an das Ziel übergeben. Wenn Sie diese Eigenschaft verwenden, wird nichts aus dem Ereignistext selbst an das Ziel weitergeleitet.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Input` Eigenschaft einer `AWS::Events::Rule Target` Ressource übergeben.

`InputPath`

Wenn Sie nicht das gesamte übereinstimmende Ereignis an das Ziel übergeben möchten, verwenden Sie die `InputPath` Eigenschaft, um zu beschreiben, welcher Teil des Ereignisses übergeben werden soll.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `InputPath` Eigenschaft einer `AWS::Events::Rule Target` Ressource übergeben.

`Pattern`

Beschreibt, welche Ereignisse an das angegebene Ziel weitergeleitet werden. Weitere Informationen finden Sie unter Ereignisse und Ereignismuster EventBridge im EventBridge Amazon-Benutzerhandbuch.

Typ: EventPattern

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `EventPattern` Eigenschaft einer `AWS::Events::Rule` Ressource übergeben.

Beispiele

CloudWatchEvent

Im Folgenden finden Sie ein Beispiel für einen `CloudWatchEvent` Ereignisquellentyp.

YAML

```
CWEvent:
  Type: CloudWatchEvent
  Properties:
    Input: '{"Key": "Value"}'
    Pattern:
      detail:
        state:
          - running
```

EventBridgeRule

Das Objekt, das einen `EventBridgeRule` Ereignisquellentyp beschreibt, der Ihren Zustandsmaschine als Ziel für eine EventBridge Amazon-Regel festlegt. Weitere Informationen finden Sie unter Was ist Amazon EventBridge? im EventBridge Amazon-Benutzerhandbuch.

AWS SAM generiert eine AWS::Events::RuleRessource, wenn dieser Ereignistyp festgelegt ist.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
  DeadLetterConfig: DeadLetterConfig
  EventBusName: String
  Input: String
  InputPath: String
  InputTransformer: InputTransformer
  Pattern: EventPattern
  RetryPolicy: RetryPolicy
  RuleName: String
  State: String
  Target: Target
```

Eigenschaften

`DeadLetterConfig`

Konfigurieren Sie die Amazon Simple Queue Service (Amazon SQS) -Warteschlange, über die Ereignisse nach einem fehlgeschlagenen Zielaufruf EventBridge gesendet werden. Der Aufruf kann beispielsweise fehlschlagen, wenn ein Ereignis an eine Lambda-Funktion gesendet wird, die nicht existiert, oder wenn EventBridge die Berechtigungen zum Aufrufen der Lambda-Funktion nicht ausreichen. Weitere Informationen finden Sie unter Richtlinien zur Wiederholung von Ereignissen und Verwenden von Warteschlangen mit unerlaubten Briefen im Amazon- Benutzerhandbuch. EventBridge

Typ: DeadLetterConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `DeadLetterConfig` Eigenschaft des Datentyps. `AWS::Events::Rule Target` Die AWS SAM Version dieser Eigenschaft enthält zusätzliche Untereigenschaften für den Fall, dass Sie die Warteschlange AWS SAM für unzustellbare Briefe erstellen möchten.

`EventBusName`

Der Ereignisbus, der dieser Regel zugeordnet werden soll. Wenn Sie diese Eigenschaft weglassen, wird der AWS SAM Standardereignisbus verwendet.

Typ: Zeichenfolge

Required: No

Standard: Standard-Event-Bus

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `EventBusName` Eigenschaft einer `AWS::Events::Rule` Ressource übergeben.

`Input`

Gültiger JSON-Text wurde an das Ziel übergeben. Wenn Sie diese Eigenschaft verwenden, wird nichts aus dem Ereignistext selbst an das Ziel weitergeleitet.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Input` Eigenschaft einer `AWS::Events::Rule Target` Ressource übergeben.

`InputPath`

Wenn Sie nicht das gesamte übereinstimmende Ereignis an das Ziel übergeben möchten, verwenden Sie die `InputPath` Eigenschaft, um zu beschreiben, welcher Teil des Ereignisses übergeben werden soll.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `InputPath` Eigenschaft einer `AWS::Events::Rule Target` Ressource übergeben.

`InputTransformer`

Einstellungen, mit denen Sie benutzerdefinierte Eingaben für ein Ziel basierend auf bestimmten Ereignisdaten bereitstellen können. Sie können einzelne oder mehrere Schlüssel-Wert-Paare aus dem Ereignis extrahieren und diese Daten dann verwenden, um benutzerdefinierte Eingaben an das Ziel zu senden. Weitere Informationen finden Sie unter Amazon EventBridge Input Transformation im EventBridge Amazon-Benutzerhandbuch.

Typ: InputTransformer

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `InputTransformer` Eigenschaft eines `AWS::Events::Rule Target` Datentyps übergeben.

`Pattern`

Beschreibt, welche Ereignisse an das angegebene Ziel weitergeleitet werden. Weitere Informationen finden Sie unter Ereignisse und Ereignismuster EventBridge im EventBridge Amazon-Benutzerhandbuch.

Typ: EventPattern

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `EventPattern` Eigenschaft einer `AWS::Events::Rule` Ressource übergeben.

`RetryPolicy`

Ein `RetryPolicy`-Objekt, das Informationen zu den Richtlinieneinstellungen für Wiederholungsversuche enthält. Weitere Informationen finden Sie unter Richtlinien zur Wiederholung von Ereignissen und Verwenden von Warteschlangen mit unerlaubten Briefen im Amazon-Benutzerhandbuch. EventBridge

Typ: RetryPolicy

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `RetryPolicy` Eigenschaft des Datentyps übergeben. `AWS::Events::Rule Target`

`RuleName`

Der Name der Regel.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Name` Eigenschaft einer `AWS::Events::Rule` Ressource übergeben.

`State`

Der Status der Regel.

Gültige Werte: `[ DISABLED | ENABLED ]`

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `State` Eigenschaft einer `AWS::Events::Rule` Ressource übergeben.

`Target`

Die AWS Ressource, die EventBridge aufgerufen wird, wenn eine Regel ausgelöst wird. Sie können diese Eigenschaft verwenden, um die logische ID des Ziels anzugeben. Wenn diese Eigenschaft nicht angegeben ist, wird die logische ID des Ziels AWS SAM generiert.

Typ: Ziel

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `Targets` Eigenschaft einer `AWS::Events::Rule` Ressource. In der AWS SAM Version dieser Eigenschaft können Sie nur die logische ID eines einzelnen Ziels angeben.

Beispiele

EventBridgeRule

Im Folgenden finden Sie ein Beispiel für einen `EventBridgeRule` Ereignisquellentyp.

YAML

```
EBRule:
  Type: EventBridgeRule
  Properties:
    Input: '{"Key": "Value"}'
    Pattern:
      detail:
        state:
          - terminated
```

DeadLetterConfig

Das Objekt, das zur Angabe der Amazon Simple Queue Service (Amazon SQS) -Warteschlange verwendet wird, über die Ereignisse nach einem fehlgeschlagenen Zielaufruf EventBridge gesendet werden. Der Aufruf kann beispielsweise fehlschlagen, wenn ein Ereignis an eine Zustandsmaschine gesendet wird, die nicht existiert, oder wenn die Berechtigungen zum Aufrufen der Zustandsmaschine nicht ausreichen. Weitere Informationen finden Sie unter Richtlinien zur Wiederholung von Ereignissen und Verwenden von Warteschlangen mit unerlaubten Briefen im Amazon-Benutzerhandbuch. EventBridge

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
  Arn: String
```

```
  QueueLogicalId: String
  Type: String
```

Eigenschaften

`Arn`

Der Amazon-Ressourcenname (ARN) der Amazon SQS-Warteschlange, der als Ziel für die Warteschlange mit unzustellbaren Briefen angegeben wurde.

> **Note**
>
> Geben Sie entweder die `Type` Eigenschaft oder `Arn` die Eigenschaft an, aber nicht beide.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Arn` Eigenschaft des `AWS::Events::Rule` `DeadLetterConfig` Datentyps übergeben.

`QueueLogicalId`

Der benutzerdefinierte Name der Warteschlange mit unerlaubten Briefen, die sie AWS SAM erstellt, `Type` ist angegeben.

> **Note**
>
> Wenn die `Type` Eigenschaft nicht festgelegt ist, wird diese Eigenschaft ignoriert.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Type`

Der Typ der Warteschlange. Wenn diese Eigenschaft festgelegt ist, wird AWS SAM automatisch eine Warteschlange für unzustellbare Nachrichten erstellt und die erforderliche ressourcenbasierte

Richtlinie angehängt, um der Regelressource die Erlaubnis zu erteilen, Ereignisse an die Warteschlange zu senden.

> **Note**
>
> Geben Sie entweder die `Type` Eigenschaft oder die Eigenschaft an, aber nicht beide`Arn`.

Gültige Werte: `SQS`

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

DeadLetterConfig

DeadLetterConfig

YAML

```
DeadLetterConfig:
  Type: SQS
  QueueLogicalId: MyDLQ
```

Target

Konfiguriert die AWS Ressource, die EventBridge aufgerufen wird, wenn eine Regel ausgelöst wird.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
  Id: String
```

Eigenschaften

`Id`

Die logische ID des Ziels.

Der Wert von `Id` kann alphanumerische Zeichen, Punkte (`.`), Bindestriche () und Unterstriche (`-`) enthalten. _

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Id` Eigenschaft des Datentyps übergeben. `AWS::Events::Rule` `Target`

Beispiele

Target

YAML

```
EBRule:
  Type: EventBridgeRule
  Properties:
    Target:
      Id: MyTarget
```

Schedule

Das Objekt, das einen `Schedule` Ereignisquellentyp beschreibt, der Ihren Zustandsmaschine als Ziel einer EventBridge Regel festlegt, die nach einem Zeitplan ausgelöst wird. Weitere Informationen finden Sie unter Was ist Amazon EventBridge? im EventBridge Amazon-Benutzerhandbuch.

AWS Serverless Application Model (AWS SAM) generiert eine AWS::Events::RuleRessource, wenn dieser Ereignistyp festgelegt ist.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
  DeadLetterConfig: DeadLetterConfig
  Description: String
  Enabled: Boolean
  Input: String
  Name: String
  RetryPolicy: RetryPolicy
  RoleArn: String
  Schedule: String
  State: String
  Target: Target
```

Eigenschaften

`DeadLetterConfig`

Konfigurieren Sie die Amazon Simple Queue Service (Amazon SQS) -Warteschlange, über die Ereignisse nach einem fehlgeschlagenen Zielaufruf EventBridge gesendet werden. Der Aufruf kann beispielsweise fehlschlagen, wenn ein Ereignis an eine Lambda-Funktion gesendet wird, die nicht existiert, oder wenn EventBridge nicht genügend Berechtigungen zum Aufrufen der Lambda-Funktion vorhanden sind. Weitere Informationen finden Sie unter Richtlinien zur Wiederholung von Ereignissen und Verwenden von Warteschlangen mit unerlaubten Briefen im Amazon-Benutzerhandbuch. EventBridge

Typ: DeadLetterConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `DeadLetterConfig` Eigenschaft des Datentyps. `AWS::Events::Rule Target` Die AWS SAM Version dieser Eigenschaft enthält zusätzliche Untereigenschaften für den Fall, dass Sie die Warteschlange AWS SAM für unzustellbare Briefe erstellen möchten.

`Description`

Eine Beschreibung der Regel.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Description` Eigenschaft einer Ressource übergeben. `AWS::Events::Rule`

`Enabled`

Gibt an, ob die Regel aktiviert ist.

Um die Regel zu deaktivieren, setzen Sie diese Eigenschaft auf`false`.

> **Note**
>
> Geben Sie entweder die `State` Eigenschaft `Enabled` oder an, aber nicht beide.

Typ: Boolesch

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `State` Eigenschaft einer `AWS::Events::Rule` Ressource. Wenn diese Eigenschaft auf `true` gesetzt ist, gilt sie AWS SAM als erfolgreich`ENABLED`, andernfalls gilt sie als erfolgreich`DISABLED`.

`Input`

Gültiger JSON-Text wurde an das Ziel übergeben. Wenn Sie diese Eigenschaft verwenden, wird nichts aus dem Ereignistext selbst an das Ziel weitergeleitet.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Input` Eigenschaft einer `AWS::Events::Rule Target` Ressource übergeben.

`Name`

Der Name der Regel. Wenn Sie keinen Namen angeben, CloudFormation generiert es eine eindeutige physische ID und verwendet diese ID als Regelnamen.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Name` Eigenschaft einer `AWS::Events::Rule` Ressource übergeben.

`RetryPolicy`

Ein `RetryPolicy`-Objekt, das Informationen zu den Richtlinieneinstellungen für Wiederholungsversuche enthält. Weitere Informationen finden Sie unter Richtlinien zur Wiederholung von Ereignissen und Verwenden von Warteschlangen mit unerlaubten Briefen im Amazon-Benutzerhandbuch. EventBridge

Typ: RetryPolicy

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `RetryPolicy` Eigenschaft des Datentyps übergeben. `AWS::Events::Rule Target`

`RoleArn`

Der ARN der IAM-Rolle, die der EventBridge Scheduler für das Ziel verwendet, wenn der Zeitplan aufgerufen wird.

Typ: RoleArn

Erforderlich: Nein. Falls nicht angegeben, wird eine neue Rolle erstellt und verwendet.

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `RoleArn` Eigenschaft des `AWS::Scheduler::Schedule Target` Datentyps übergeben.

`Schedule`

Der Planungsausdruck, der bestimmt, wann und wie oft die Regel ausgeführt wird. Weitere Informationen finden Sie unter Planen von Ausdrücken für Regeln.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `ScheduleExpression` Eigenschaft einer `AWS::Events::Rule` Ressource übergeben.

`State`

Der Status der Regel.

Zulässige Werte: `DISABLED | ENABLED`

> **Note**
>
> Geben Sie entweder die `State` Eigenschaft `Enabled` oder an, aber nicht beide.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `State` Eigenschaft einer `AWS::Events::Rule` Ressource übergeben.

`Target`

Die AWS Ressource, die EventBridge aufgerufen wird, wenn eine Regel ausgelöst wird. Sie können diese Eigenschaft verwenden, um die logische ID des Ziels anzugeben. Wenn diese Eigenschaft nicht angegeben ist, wird die logische ID des Ziels AWS SAM generiert.

Typ: Ziel

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `Targets` Eigenschaft einer `AWS::Events::Rule` Ressource. In der AWS SAM Version dieser Eigenschaft können Sie nur die logische ID eines einzelnen Ziels angeben.

Beispiele

CloudWatch Veranstaltung planen

CloudWatch Beispiel für einen Termin planen

YAML

```
CWSchedule:
  Type: Schedule
  Properties:
    Schedule: 'rate(1 minute)'
```

```
    Name: TestSchedule
    Description: test schedule
    Enabled: false
```

DeadLetterConfig

Das Objekt, das zur Angabe der Amazon Simple Queue Service (Amazon SQS) -Warteschlange verwendet wird, über die Ereignisse nach einem fehlgeschlagenen Zielaufruf EventBridge gesendet werden. Der Aufruf kann beispielsweise fehlschlagen, wenn ein Ereignis an eine Zustandsmaschine gesendet wird, die nicht existiert, oder wenn die Berechtigungen zum Aufrufen der Zustandsmaschine nicht ausreichen. Weitere Informationen finden Sie unter Richtlinien zur Wiederholung von Ereignissen und Verwenden von Warteschlangen mit unerlaubten Briefen im Amazon-Benutzerhandbuch. EventBridge

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
Arn: String
QueueLogicalId: String
Type: String
```

Eigenschaften

`Arn`

Der Amazon-Ressourcenname (ARN) der Amazon SQS-Warteschlange, der als Ziel für die Warteschlange mit unzustellbaren Briefen angegeben wurde.

> **Note**
>
> Geben Sie entweder die `Type` Eigenschaft oder `Arn` die Eigenschaft an, aber nicht beide.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Arn` Eigenschaft des `AWS::Events::Rule` `DeadLetterConfig` Datentyps übergeben.

`QueueLogicalId`

Der benutzerdefinierte Name der Warteschlange mit unerlaubten Briefen, die sie AWS SAM erstellt, `Type` ist angegeben.

> **Note**
>
> Wenn die `Type` Eigenschaft nicht festgelegt ist, wird diese Eigenschaft ignoriert.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`Type`

Der Typ der Warteschlange. Wenn diese Eigenschaft festgelegt ist, wird AWS SAM automatisch eine Warteschlange für unzustellbare Nachrichten erstellt und die erforderliche ressourcenbasierte Richtlinie angehängt, um der Regelressource die Erlaubnis zu erteilen, Ereignisse an die Warteschlange zu senden.

> **Note**
>
> Geben Sie entweder die `Type` Eigenschaft oder die Eigenschaft an, aber nicht beide`Arn`.

Gültige Werte: `SQS`

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

Beispiele

DeadLetterConfig

DeadLetterConfig

YAML

```
DeadLetterConfig:
  Type: SQS
  QueueLogicalId: MyDLQ
```

Target

Konfiguriert die AWS Ressource, die EventBridge aufgerufen wird, wenn eine Regel ausgelöst wird.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
  Id: String
```

Eigenschaften

`Id`

Die logische ID des Ziels.

Der Wert von `Id` kann alphanumerische Zeichen, Punkte (`.`), Bindestriche () und Unterstriche (`-`) enthalten. _

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Id` Eigenschaft des Datentyps übergeben. `AWS::Events::Rule Target`

Beispiele

Target

YAML

```
EBRule:
  Type: Schedule
  Properties:
    Target:
      Id: MyTarget
```

ScheduleV2

Das Objekt, das einen `ScheduleV2` Ereignisquellentyp beschreibt, der Ihren Zustandsmaschine als Ziel eines Amazon EventBridge Scheduler-Ereignisses festlegt, das nach einem Zeitplan ausgelöst wird. Weitere Informationen finden Sie unter Was ist Amazon EventBridge Scheduler? im EventBridge Scheduler-Benutzerhandbuch.

AWS Serverless Application Model (AWS SAM) generiert eine AWS::Scheduler::ScheduleRessource, wenn dieser Ereignistyp festgelegt ist.

Syntax

Verwenden Sie die folgende Syntax, um diese Entität in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) zu deklarieren.

YAML

```
DeadLetterConfig: DeadLetterConfig
Description: String
EndDate: String
FlexibleTimeWindow: FlexibleTimeWindow
GroupName: String
Input: String
KmsKeyArn: String
Name: String
OmitName: Boolean
PermissionsBoundary: String
RetryPolicy: RetryPolicy
RoleArn: String
ScheduleExpression: String
```

```
ScheduleExpressionTimezone: String
StartDate: String
State: String
```

Eigenschaften

`DeadLetterConfig`

Konfigurieren Sie die Amazon Simple Queue Service (Amazon SQS) -Warteschlange, über die Ereignisse nach einem fehlgeschlagenen Zielaufruf EventBridge gesendet werden. Der Aufruf kann beispielsweise fehlschlagen, wenn ein Ereignis an eine Lambda-Funktion gesendet wird, die nicht existiert, oder wenn EventBridge die Berechtigungen zum Aufrufen der Lambda-Funktion nicht ausreichen. Weitere Informationen finden Sie im Scheduler-Benutzerhandbuch unter Konfiguration einer Warteschlange mit uneingeschränkten Briefen für den EventBridge Scheduler. EventBridge

Typ: DeadLetterConfig

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft ähnelt der `DeadLetterConfig` Eigenschaft des Datentyps. `AWS::Scheduler::Schedule Target` Die AWS SAM Version dieser Eigenschaft enthält zusätzliche Untereigenschaften für den Fall, dass Sie die Warteschlange AWS SAM für unzustellbare Briefe erstellen möchten.

`Description`

Eine Beschreibung des Zeitplans.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Description` Eigenschaft einer `AWS::Scheduler::Schedule` Ressource übergeben.

`EndDate`

Das Datum in UTC, bevor der Zeitplan sein Ziel aufrufen kann. Abhängig vom Wiederholungsausdruck des Zeitplans können Aufrufe an oder vor dem von Ihnen angegebenen EndDate anhalten.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `EndDate` Eigenschaft einer `AWS::Scheduler::Schedule` Ressource übergeben.

`FlexibleTimeWindow`

Ermöglicht die Konfiguration eines Fensters, in dem ein Zeitplan aufgerufen werden kann.

Typ: FlexibleTimeWindow

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `FlexibleTimeWindow` Eigenschaft einer `AWS::Scheduler::Schedule` Ressource übergeben.

`GroupName`

Der Name der Zeitplangruppe, die diesem Zeitplan zugeordnet werden soll. Wenn nicht definiert, wird die Standardgruppe verwendet.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `GroupName` Eigenschaft einer `AWS::Scheduler::Schedule` Ressource übergeben.

`Input`

Gültiger JSON-Text wurde an das Ziel übergeben. Wenn Sie diese Eigenschaft verwenden, wird nichts aus dem Ereignistext selbst an das Ziel weitergeleitet.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Input` Eigenschaft einer `AWS::Scheduler::Schedule Target` Ressource übergeben.

`KmsKeyArn`

Der ARN für einen KMS-Schlüssel, der zur Verschlüsselung von Kundendaten verwendet wird.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `KmsKeyArn` Eigenschaft einer `AWS::Scheduler::Schedule` Ressource übergeben.

`Name`

Der Name des Plans. Wenn Sie keinen Namen angeben, AWS SAM generiert es einen Namen im Format *`StateMachine-Logical-IDEvent-Source-Name`* und verwendet diese ID als Namen des Zeitplans.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `Name` Eigenschaft einer `AWS::Scheduler::Schedule` Ressource übergeben.

`OmitName`

AWS SAM Generiert und verwendet standardmäßig einen Zeitplannamen im Format von*`<State-machine-logical-ID><event-source-name>`*. Stellen Sie diese Eigenschaft so ein`true`, dass eine eindeutige physische ID CloudFormation generiert und diese stattdessen für den Zeitplannamen verwendet wird.

Typ: Boolesch

Required: No

Standardwert: `false`

CloudFormation Kompatibilität: Diese Eigenschaft ist einzigartig für AWS SAM und hat kein CloudFormation Äquivalent.

`PermissionsBoundary`

Der ARN der Richtlinie, mit der die Berechtigungsgrenze für die Rolle festgelegt wurde.

> **Note**
>
> Wenn sie definiert `PermissionsBoundary` ist, AWS SAM werden dieselben Grenzen auf die IAM-Zielrolle des Scheduler-Zeitplans angewendet.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `PermissionsBoundary` Eigenschaft einer `AWS::IAM::Role` Ressource übergeben.

`RetryPolicy`

Ein `RetryPolicy`-Objekt, das Informationen zu den Richtlinieneinstellungen für Wiederholungsversuche enthält.

Typ: RetryPolicy

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `RetryPolicy` Eigenschaft des `AWS::Scheduler::Schedule` `Target` Datentyps übergeben.

`RoleArn`

Der ARN der IAM-Rolle, die der EventBridge Scheduler für das Ziel verwendet, wenn der Zeitplan aufgerufen wird.

Typ: RoleArn

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `RoleArn` Eigenschaft des Datentyps übergeben. `AWS::Scheduler::Schedule` `Target`

`ScheduleExpression`

Der Planungsausdruck, der bestimmt, wann und wie oft der Zeitplan ausgeführt wird.

Typ: Zeichenfolge

Erforderlich: Ja

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `ScheduleExpression` Eigenschaft einer `AWS::Scheduler::Schedule` Ressource übergeben.

`ScheduleExpressionTimezone`

Die Zeitzone, in der der Planungsausdruck ausgewertet wird.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `ScheduleExpressionTimezone` Eigenschaft einer `AWS::Scheduler::Schedule` Ressource übergeben.

`StartDate`

Das Datum in UTC, nach dem der Zeitplan mit dem Aufrufen eines Ziels beginnen kann. Abhängig vom Wiederholungsausdruck des Zeitplans können Aufrufe an oder nach dem von Ihnen angegebenen StartDate erfolgen.

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `StartDate` Eigenschaft einer `AWS::Scheduler::Schedule` Ressource übergeben.

`State`

Der Status des Zeitplans.

Zulässige Werte: `DISABLED | ENABLED`

Typ: Zeichenfolge

Required: No

CloudFormation Kompatibilität: Diese Eigenschaft wird direkt an die `State` Eigenschaft einer `AWS::Scheduler::Schedule` Ressource übergeben.

Beispiele

Einfaches Beispiel für die Definition einer ScheduleV2-Ressource

```
StateMachine:
  Type: AWS::Serverless::StateMachine
  Properties:
    Name: MyStateMachine
    Events:
      ScheduleEvent:
        Type: ScheduleV2
        Properties:
          ScheduleExpression: "rate(1 minute)"
```

```
      ComplexScheduleEvent:
        Type: ScheduleV2
        Properties:
          ScheduleExpression: rate(1 minute)
          FlexibleTimeWindow:
            Mode: FLEXIBLE
            MaximumWindowInMinutes: 5
          StartDate: '2022-12-28T12:00:00.000Z'
          EndDate: '2023-01-28T12:00:00.000Z'
          ScheduleExpressionTimezone: UTC
          RetryPolicy:
            MaximumRetryAttempts: 5
            MaximumEventAgeInSeconds: 300
          DeadLetterConfig:
            Type: SQS
    DefinitionUri:
      Bucket: sam-sam-s3-demo-bucket
      Key: my-state-machine.asl.json
      Version: 3
    Policies:
      - LambdaInvokePolicy:
          FunctionName: !Ref MyFunction
```

Generiert CloudFormation Ressourcen für AWS SAM

Dieser Abschnitt enthält Einzelheiten zu den CloudFormation Ressourcen, die bei der Verarbeitung Ihrer AWS SAM AWS Vorlage erstellt werden. Welche CloudFormation Ressourcen AWS SAM generiert werden, hängt von den von Ihnen angegebenen Szenarien ab. Ein Szenario ist die Kombination von AWS SAM Ressourcen und Eigenschaften, die in Ihrer Vorlagendatei angegeben sind. Sie können an anderer Stelle in Ihrer Vorlagendatei auf die generierten CloudFormation Ressourcen verweisen, ähnlich wie Sie auf Ressourcen verweisen, die Sie in Ihrer Vorlagendatei explizit deklarieren.

Wenn Sie beispielsweise eine `AWS::Serverless::Function` Ressource in Ihrer AWS SAM Vorlagendatei angeben, wird AWS SAM immer eine `AWS::Lambda::Function` Basisressource generiert. Wenn Sie auch die optionale `AutoPublishAlias` Eigenschaft angeben, AWS SAM werden zusätzlich `AWS::Lambda::Alias` auch `AWS::Lambda::Version` Ressourcen generiert.

In diesem Abschnitt werden die Szenarien und die CloudFormation Ressourcen, die sie generieren, aufgeführt. Außerdem wird gezeigt, wie Sie in Ihrer AWS SAM Vorlagendatei auf die generierten CloudFormation Ressourcen verweisen können.

Die Referenzierung wurde generiert CloudFormation Ressourcen

Sie haben zwei Möglichkeiten, generierte CloudFormation Ressourcen in Ihrer AWS SAM Vorlagendatei zu referenzieren: nach `LogicalId` oder nach referenzierbarer Eigenschaft.

Die Referenzierung wurde generiert CloudFormation Ressourcen von LogicalId

Die CloudFormation Ressourcen, die AWS SAM generiert werden, haben jeweils eine`LogicalId`, was eine alphanumerische (A-Z, a-z, 0-9) Kennung ist, die innerhalb einer Vorlagendatei eindeutig ist. AWS SAM verwendet die `LogicalIds` AWS SAM Ressourcen in Ihrer Vorlagendatei, um die `LogicalIds` generierten CloudFormation Ressourcen zu erstellen. Sie können die `LogicalId` einer generierten CloudFormation Ressource verwenden, um auf Eigenschaften dieser Ressource in Ihrer Vorlagendatei zuzugreifen, genau wie Sie es für eine CloudFormation Ressource tun würden, die Sie explizit deklariert haben. Weitere Informationen zu `LogicalIds` Gin CloudFormation und AWS SAM Vorlagen finden Sie unter Ressourcen im AWS CloudFormation Benutzerhandbuch.

> **Note**
>
> Einige generierte Ressourcen enthalten einen eindeutigen Hashwert, um Namespace-Konflikte zu vermeiden. `LogicalIds` Diese Ressourcen werden bei `LogicalIds` der Erstellung des Stacks abgeleitet. Sie können sie erst abrufen, nachdem der Stack mit den AWS-Managementkonsole AWS CLI, oder einem der AWS SDKs erstellt wurde. Es wird nicht empfohlen, diese Ressourcen mit zu referenzieren`LogicalId`, da sich die Hashwerte ändern könnten.

Die Referenzierung wurde generiert CloudFormation Ressourcen nach referenzierbarem Eigentum

AWS SAM Stellt für einige generierte Ressourcen eine referenzierbare Eigenschaft der Ressource bereit. AWS SAM Sie können diese Eigenschaft verwenden, um in Ihrer AWS SAM Vorlagendatei auf eine generierte CloudFormation Ressource und ihre Eigenschaften zu verweisen.

> **Note**
>
> Nicht alle generierten CloudFormation Ressourcen haben referenzierbare Eigenschaften. Für diese Ressourcen müssen Sie die `LogicalId` verwenden.

Generiert CloudFormation Ressourcenszenarien

In der folgenden Tabelle sind die AWS SAM Ressourcen und Eigenschaften zusammengefasst, aus denen sich die Szenarien zusammensetzen, mit denen CloudFormation Ressourcen generiert werden. Die Themen in der Spalte Szenarien enthalten Einzelheiten zu den zusätzlichen CloudFormation Ressourcen, die für dieses Szenario AWS SAM generiert werden.

AWS SAM Ressource	CloudFormation Basisressource	Szenarien
AWS::Serverless::Api	AWS::ApiGateway::RestApi	• DomainName Eigenschaft ist angegeben • UsagePlan Eigenschaft ist spezifiziert
AWS::Serverless::Application	AWS::CloudFormation::Stack	• Abgesehen von der Generierung der CloudFormation Basisressource gibt es keine weiteren Szenarien für diese serverlose Ressource.
AWS::Serverless::CapacityProvider	AWS::Lambda::CapacityProvider	• OperatorRole Eigenschaft ist nicht angegeben
AWS::Serverless::MicrovmImage	AWS::Lambda::MicrovmImage	• BuildRoleArn Eigenschaft ist nicht angegeben
AWS::Serverless::NetworkConnector	AWS::Lambda::NetworkConnector	• OperatorRole Eigenschaft ist nicht angegeben
AWS::Serverless::Function	AWS::Lambda::Function	• AutoPublishAlias Eigenschaft ist angegeben •

AWS SAM Ressource	CloudFormation Basisressource	Szenarien
		Die Rolleneigenschaft ist nicht angegeben • DeploymentPreference Eigenschaft ist angegeben • Eine API-Ereignisquelle ist angegeben • Eine HttpApi Ereignisquelle ist angegeben • Eine Quelle für Streaming-Ereignisse ist angegeben • Eine Ereignisquelle für eine Event-Bridge (oder einen Event-Bus) wurde angegeben • Eine IotRule Ereignisquelle ist angegeben • OnSuccess (oder OnFailure) Die Eigenschaft ist für Amazon SNS SNS-Ereignisse angegeben • OnSuccess (oder OnFailure) Die Eigenschaft ist für Amazon SQS SQS-Ereignisse angegeben
AWS::Serverless::HttpApi	AWS::ApiGatewayV2::Api	• StageName Eigenschaft ist angegeben • StageName Eigenschaft ist nicht spezifiziert • DomainName Eigenschaft ist spezifiziert
AWS::Serverless::LayerVersion	AWS::Lambda::LayerVersion	• Abgesehen von der Generierung der CloudFormation Basisressource gibt es keine weiteren Szenarien für diese serverlose Ressource.

AWS SAM Ressource	CloudFormation Basisressource	Szenarien
AWS::Serverless::SimpleTable	AWS::DynamoDB::Table	• Abgesehen von der Generierung der CloudFormation Basisressource gibt es keine weiteren Szenarien für diese serverlose Ressource.
AWS::Serverless::StateMachine	AWS::StepFunctions::StateMachine	• Die Rolleneigenschaft ist nicht angegeben • Eine API-Ereignisquelle ist angegeben • Eine Ereignisquelle für eine Event-Bridge (oder einen Event-Bus) wurde angegeben

Themen

- CloudFormation Ressourcen, die generiert werdenAWS::Serverless::Api, wenn angegeben
- CloudFormation Ressourcen, die generiert werdenAWS::Serverless::Application, wenn angegeben
- CloudFormation Ressourcen, die generiert werdenAWS::Serverless::CapacityProvider, wenn angegeben
- CloudFormation Ressourcen, die generiert werden, wenn Sie angeben AWS::Serverless::Connector
- CloudFormation Ressourcen, die generiert werdenAWS::Serverless::Function, wenn angegeben
- CloudFormation Ressourcen, die generiert werdenAWS::Serverless::GraphQLApi, wenn angegeben
- CloudFormation Ressourcen, die generiert werden AWS::Serverless::HttpApi , wenn angegeben
- CloudFormation Ressourcen, die generiert wurden, wenn AWS::Serverless::MicrovmImage ist angegeben
- CloudFormation Ressourcen, die generiert wurden, wenn AWS::Serverless::NetworkConnector ist angegeben
- CloudFormation Ressourcen, die generiert werden AWS::Serverless::WebSocketApi , wenn angegeben

- CloudFormation Ressourcen, die generiert werdenAWS::Serverless::LayerVersion, wenn angegeben
- CloudFormation Ressourcen, die generiert werdenAWS::Serverless::SimpleTable, wenn angegeben
- CloudFormation Ressourcen, die generiert werdenAWS::Serverless::StateMachine, wenn angegeben

CloudFormation Ressourcen, die generiert werdenAWS::Serverless::Api, wenn angegeben

Wenn an angegeben `AWS::Serverless::Api` ist, generiert AWS Serverless Application Model (AWS SAM) immer eine `AWS::ApiGateway::RestApi` CloudFormation Basisressource. Darüber hinaus generiert es auch immer eine `AWS::ApiGateway::Stage` und eine `AWS::ApiGateway::Deployment` Ressource.

`AWS::ApiGateway::RestApi`

`LogicalId`: *`<api-LogicalId>`*

Referenzierbare Eigenschaft: N/A (Sie müssen den verwenden, um auf diese `LogicalId` Ressource zu verweisen CloudFormation)

`AWS::ApiGateway::Stage`

`LogicalId`: *`<api-LogicalId><stage-name>`*`Stage`

*`<stage-name>`*ist die Zeichenfolge, auf die die `StageName` Eigenschaft gesetzt ist. Wenn Sie beispielsweise `StageName` auf einstellen`Gamma`, `LogicalId` ist der*`MyRestApiGamma`*`Stage`.

Referenzierbare Eigenschaft: *`<api-LogicalId>`*`.Stage`

`AWS::ApiGateway::Deployment`

`LogicalId`: *`<api-LogicalId>`*`Deployment`*`<sha>`*

*`<sha>`*ist ein eindeutiger Hashwert, der bei der Erstellung des Stacks generiert wird. Beispiel, *`MyRestApi`*`Deployment`*`926eeb5ff1`*.

Referenzierbare Eigenschaft: *`<api-LogicalId>`*`.Deployment`

Zusätzlich zu diesen CloudFormation Ressourcen werden, sofern `AWS::Serverless::Api` angegeben, zusätzliche CloudFormation Ressourcen für die folgenden Szenarien AWS SAM generiert.

Szenarien

- DomainName Eigenschaft ist angegeben
- UsagePlan Eigenschaft ist spezifiziert

DomainName Eigenschaft ist angegeben

Wenn die `DomainName` Eigenschaft der `Domain` Eigenschaft von an angegeben `AWS::Serverless::Api` ist, wird die `AWS::ApiGateway::DomainName` CloudFormation Ressource AWS SAM generiert.

`AWS::ApiGateway::DomainName`

`LogicalId`: `ApiGatewayDomainName`*`<sha>`*

*`<sha>`*ist ein eindeutiger Hashwert, der generiert wird, wenn der Stack erstellt wird. Beispiel: `ApiGatewayDomainName`*`926eeb5ff1`*.

Referenzierbare Eigenschaft: *`<api-LogicalId>`*`.DomainName`

UsagePlan Eigenschaft ist spezifiziert

Wenn die `UsagePlan` `Auth` Eigenschaft einer angegeben `AWS::Serverless::Api` ist, werden die folgenden CloudFormation Ressourcen AWS SAM generiert: `AWS::ApiGateway::UsagePlan`AWS::ApiGateway::UsagePlanKey, und`AWS::ApiGateway::ApiKey`.

`AWS::ApiGateway::UsagePlan`

`LogicalId`: *`<api-LogicalId>`*`UsagePlan`

Referenzierbare Eigenschaft: *`<api-LogicalId>`*`.UsagePlan`

`AWS::ApiGateway::UsagePlanKey`

`LogicalId`: *`<api-LogicalId>`*`UsagePlanKey`

Referenzierbares Eigentum: *`<api-LogicalId>`*`.UsagePlanKey`

`AWS::ApiGateway::ApiKey`

`LogicalId`: *`<api-LogicalId>`*`ApiKey`

Referenzierbares Eigentum: *`<api-LogicalId>`*`.ApiKey`

CloudFormation Ressourcen, die generiert werdenAWS::Serverless::Application, wenn angegeben

Wenn an angegeben `AWS::Serverless::Application` ist, generiert AWS Serverless Application Model (AWS SAM) eine `AWS::CloudFormation::Stack` CloudFormation Basisressource.

`AWS::CloudFormation::Stack`

`LogicalId`: *`<application-LogicalId>`*

Referenzierbare Eigenschaft: N/A (Sie müssen den verwenden, um auf diese `LogicalId` Ressource zu verweisen CloudFormation)

CloudFormation Ressourcen, die generiert werdenAWS::Serverless::CapacityProvider, wenn angegeben

Wenn an angegeben `AWS::Serverless::CapacityProvider` ist, generiert AWS Serverless Application Model (AWS SAM) eine `AWS::Lambda::CapacityProvider` CloudFormation Basisressource.

`AWS::Lambda::CapacityProvider`

`LogicalId`: *`<capacityprovider-LogicalId>`*

Referenzierbare Eigenschaft: N/A (Sie müssen den verwenden, um auf diese `LogicalId` Ressource zu verweisen CloudFormation)

Zusätzlich zu dieser CloudFormation Ressource werden, wenn `AWS::Serverless::CapacityProvider` angegeben, AWS SAM auch CloudFormation Ressourcen für die folgenden Szenarien generiert:

Szenarien

- OperatorRole Eigenschaft ist nicht angegeben

OperatorRole Eigenschaft ist nicht angegeben

Wenn die `OperatorRole` Eigenschaft von nicht angegeben `AWS::Serverless::CapacityProvider` ist, wird eine `AWS::IAM::Role` CloudFormation Ressource mit der angehängten `AWSLambdaManagedEC2ResourceOperator` verwalteten Richtlinie AWS SAM generiert.

`AWS::IAM::Role`

`LogicalId`: *`<capacityprovider-LogicalId>`*`OperatorRole`

Referenzierbare Eigenschaft: N/A (Sie müssen den verwenden, um auf diese `LogicalId` Ressource zu verweisen CloudFormation)

CloudFormation Ressourcen, die generiert werden, wenn Sie angeben AWS::Serverless::Connector

> **Note**
>
> Wenn Sie Konnektoren mithilfe der eingebetteten `Connectors` Eigenschaft definieren, wird diese zunächst in eine `AWS::Serverless::Connector` Ressource umgewandelt, bevor diese Ressourcen generiert werden.

Wenn Sie eine `AWS::Serverless::Connector` Ressource in einer AWS SAM Vorlage angeben, werden bei Bedarf die folgenden AWS CloudFormation Ressourcen AWS SAM generiert.

`AWS::IAM::ManagedPolicy`

`LogicalId`:*`<connector-LogicalId>`*`Policy`

Referenzierbare Eigenschaft: N/A (Um auf diese CloudFormation Ressource zu verweisen, müssen Sie die verwendenLogicalId.)

`AWS::SNS::TopicPolicy`

`LogicalId`:*`<connector-LogicalId>`*`TopicPolicy`

Referenzierbare Eigenschaft: N/A (Um auf diese CloudFormation Ressource zu verweisen, müssen Sie die `LogicalId` verwenden.)

`AWS::SQS::QueuePolicy`

`LogicalId`:*`<connector-LogicalId>`*`QueuePolicy`

Referenzierbare Eigenschaft: N/A (Um auf diese CloudFormation Ressource zu verweisen, müssen Sie die `LogicalId` verwenden.)

`AWS::Lambda::Permission`

`LogicalId`:*`<connector-LogicalId><permission>`*`LambdaPermission`

*`<permission>`*ist eine von der `Permissions` Eigenschaft angegebene Berechtigung. Beispiel, `Write`.

Referenzierbare Eigenschaft: N/A (Um auf diese CloudFormation Ressource zu verweisen, müssen Sie die verwenden`LogicalId`.)

CloudFormation Ressourcen, die generiert werden`AWS::Serverless::Function`, wenn angegeben

Wenn an angegeben `AWS::Serverless::Function` ist, erstellt AWS Serverless Application Model (AWS SAM) immer eine `AWS::Lambda::Function` CloudFormation Basisressource.

`AWS::Lambda::Function`

`LogicalId`: *`<function-LogicalId>`*

Referenzierbare Eigenschaft: N/A (Sie müssen den verwenden, um auf diese `LogicalId` Ressource zu verweisen CloudFormation)

Zusätzlich zu dieser CloudFormation Ressource werden, wenn `AWS::Serverless::Function` angegeben, AWS SAM auch CloudFormation Ressourcen für die folgenden Szenarien generiert.

Szenarien

- Eigenschaften der Kernfunktion
- Ereignisquellen

- Konfiguration des Ereignisses

Eigenschaften der Kernfunktion

In den folgenden Szenarien werden CloudFormation Ressourcen auf der Grundlage von Kernfunktionseigenschaften generiert:

Die Rolleneigenschaft ist nicht angegeben

Wenn die `Role` Eigenschaft von nicht angegeben `AWS::Serverless::Function` ist, wird eine `AWS::IAM::Role` CloudFormation Ressource AWS SAM generiert.

`AWS::IAM::Role`

`LogicalId`: *`<function-LogicalId>`*`Role`

Referenzierbare Eigenschaft: N/A (Sie müssen den verwenden, um auf diese `LogicalId` Ressource zu verweisen CloudFormation)

AutoPublishAlias Eigenschaft ist angegeben

Wenn die `AutoPublishAlias` Eigenschaft von an angegeben `AWS::Serverless::Function` ist, werden die folgenden CloudFormation Ressourcen AWS SAM generiert: `AWS::Lambda::Alias` und`AWS::Lambda::Version`.

`AWS::Lambda::Alias`

`LogicalId`: *`<function-LogicalId>`*`Alias`*`<alias-name>`*

*`<alias-name>`*ist die Zeichenfolge, die auf gesetzt `AutoPublishAlias` ist. Wenn Sie beispielsweise `AutoPublishAlias` auf einstellen`live`, `LogicalId` lautet das: *`MyFunction`* Alias*`live`*.

Referenzierbare Eigenschaft: *`<function-LogicalId>`*`.Alias`

`AWS::Lambda::Version`

`LogicalId`: *`<function-LogicalId>`*`Version`*`<sha>`*

*`<sha>`*ist ein eindeutiger Hashwert, der bei der Erstellung des Stacks generiert wird. Zum Beispiel *`MyFunction`* Version*`926eeb5ff1`*.

Referenzierbare Eigenschaft: *<function-LogicalId>*.Version

Weitere Informationen zur AutoPublishAlias Eigenschaft finden Sie im Abschnitt Eigenschaften von AWS::Serverless::Function.

DeploymentPreference Eigenschaft ist angegeben

Wenn die DeploymentPreference Eigenschaft von an angegeben AWS::Serverless::Function ist, werden die folgenden CloudFormation Ressourcen AWS SAM generiert: AWS::CodeDeploy::Application undAWS::CodeDeploy::DeploymentGroup. Wenn die Role Eigenschaft des DeploymentPreference Objekts nicht angegeben ist, wird AWS SAM außerdem eine AWS::IAM::Role CloudFormation Ressource generiert.

AWS::CodeDeploy::Application

LogicalId: ServerlessDeploymentApplication

Referenzierbare Eigenschaft: N/A (Sie müssen den verwenden, um auf diese LogicalId Ressource zu verweisen CloudFormation)

AWS::CodeDeploy::DeploymentGroup

LogicalId: *<function-LogicalId>*DeploymentGroup

Referenzierbare Eigenschaft: N/A (Sie müssen das verwenden, um auf diese Ressource LogicalId zu verweisen) CloudFormation

AWS::IAM::Role

LogicalId: CodeDeployServiceRole

Referenzierbare Eigenschaft: N/A (Sie müssen das verwenden, um auf diese Ressource LogicalId zu verweisen) CloudFormation

FunctionUrlConfig Eigenschaft ist angegeben

Wenn die FunctionUrlConfig Eigenschaft angegeben ist, AWS SAM generiert verschiedene CloudFormation Ressourcen auf der Grundlage vonAuthType.

Wenn AuthType: NONE angegeben, werden die folgenden CloudFormation Ressourcen AWS SAM generiert:

`AWS::Lambda::Permission`(Zugriff aufrufen)

`LogicalId`: *`<function-LogicalId>`*`URLInvokeAllowPublicAccess`

Referenzierbare Eigenschaft: N/A (Sie müssen den verwenden, um auf diese Ressource `LogicalId` zu verweisen) CloudFormation

`AWS::Lambda::Permission`(Öffentlicher Zugang)

`LogicalId`: *`<function-LogicalId>`*`UrlPublicPermissions`

Referenzierbare Eigenschaft: N/A (Sie müssen das verwenden, um auf diese `LogicalId` Ressource zu verweisen CloudFormation)

`AWS::Lambda::Url`

`LogicalId`: *`<function-LogicalId>`*`Url`

Referenzierbare Eigenschaft: N/A (Sie müssen das verwenden, um auf diese Ressource `LogicalId` zu verweisen) CloudFormation

Wenn `AuthType: AWS_IAM` angegeben, AWS SAM generiert nur:

`AWS::Lambda::Url`

`LogicalId`: *`<function-LogicalId>`*`Url`

Referenzierbare Eigenschaft: N/A (Sie müssen den verwenden, um auf diese `LogicalId` Ressource zu verweisen CloudFormation)

Weitere Informationen zur `FunctionUrlConfig` Eigenschaft finden Sie unter. FunctionUrlConfig

Ereignisquellen

In den folgenden Szenarien werden CloudFormation Ressourcen auf der Grundlage von Ereignisquellen generiert:

Eine API-Ereignisquelle ist angegeben

Wenn die `Event` Eigenschaft von auf gesetzt `AWS::Serverless::Function` ist`Api`, die `RestApiId` Eigenschaft aber nicht angegeben ist, wird die `AWS::ApiGateway::RestApi` CloudFormation Ressource AWS SAM generiert.

`AWS::ApiGateway::RestApi`

`LogicalId`: `ServerlessRestApi`

Referenzierbare Eigenschaft: N/A (Sie müssen den verwenden, um auf diese `LogicalId` Ressource zu verweisen CloudFormation)

Eine HttpApi Ereignisquelle ist angegeben

Wenn die `Event` Eigenschaft von auf gesetzt `AWS::Serverless::Function` ist`HttpApi`, die `ApiId` Eigenschaft aber nicht angegeben ist, wird die `AWS::ApiGatewayV2::Api` CloudFormation Ressource AWS SAM generiert.

`AWS::ApiGatewayV2::Api`

`LogicalId`: `ServerlessHttpApi`

Referenzierbare Eigenschaft: N/A (Sie müssen den verwenden, um auf diese `LogicalId` Ressource zu verweisen CloudFormation)

Eine Quelle für Streaming-Ereignisse ist angegeben

Wenn die `Event` Eigenschaft von auf einen der Streaming-Typen gesetzt `AWS::Serverless::Function` ist, AWS SAM wird die `AWS::Lambda::EventSourceMapping` CloudFormation Ressource generiert. Dies gilt für die folgenden Typen: `DynamoDB`,`Kinesis`,`MQ`,`MSK`, und`SQS`.

`AWS::Lambda::EventSourceMapping`

`LogicalId`: *`<function-LogicalId><event-LogicalId>`*

Referenzierbare Eigenschaft: N/A (Sie müssen den verwenden, um auf diese `LogicalId` Ressource zu verweisen CloudFormation)

Eine Ereignisquelle für eine Event-Bridge (oder einen Event-Bus) wurde angegeben

Wenn die `Event` Eigenschaft von auf einen der Typen Event Bridge (oder Event Bus) gesetzt `AWS::Serverless::Function` ist, AWS SAM wird die `AWS::Events::Rule` CloudFormation Ressource generiert. Dies gilt für die folgenden Typen: `EventBridgeRuleSchedule`, und`CloudWatchEvents`.

`AWS::Events::Rule`

`LogicalId`: *`<function-LogicalId><event-LogicalId>`*

Referenzierbare Eigenschaft: N/A (Sie müssen den verwenden, um auf diese `LogicalId` Ressource zu verweisen CloudFormation)

Eine IotRule Ereignisquelle ist angegeben

Wenn die `Event` Eigenschaft von an auf Io gesetzt `AWS::Serverless::Function` istTRule, wird die `AWS::IoT::TopicRule` CloudFormation Ressource AWS SAM generiert.

`AWS::IoT::TopicRule`

`LogicalId`: *`<function-LogicalId><event-LogicalId>`*

Referenzierbare Eigenschaft: N/A (Sie müssen den verwenden, um auf diese `LogicalId` Ressource zu verweisen CloudFormation)

Konfiguration des Ereignisses

In den folgenden Szenarien werden CloudFormation Ressourcen basierend auf der Ereigniskonfiguration generiert:

OnSuccess (oder OnFailure) Die Eigenschaft ist für Amazon SNS SNS-Ereignisse angegeben

Wenn die Eigenschaft `OnSuccess` (oder`OnFailure`) der `DestinationConfig` Eigenschaft der `EventInvokeConfig` Eigenschaft von an angegeben `AWS::Serverless::Function` ist und der Zieltyp ist, SNS aber der Ziel-ARN nicht angegeben ist, werden die folgenden CloudFormation Ressourcen AWS SAM generiert: `AWS::Lambda::EventInvokeConfig` und`AWS::SNS::Topic`.

`AWS::Lambda::EventInvokeConfig`

`LogicalId`: *`<function-LogicalId>`*`EventInvokeConfig`

Referenzierbare Eigenschaft: N/A (Sie müssen den verwenden, um auf diese `LogicalId` Ressource zu verweisen CloudFormation)

`AWS::SNS::Topic`

`LogicalId`: *`<function-LogicalId>OnSuccessTopic`* (oder) *`<function-LogicalId>`*`OnFailureTopic`

Referenzierbares Eigentum: *<function-LogicalId>*`.DestinationTopic`

Wenn beide `OnSuccess` und für ein Amazon SNS SNS-Ereignis angegeben `OnFailure` sind, müssen Sie zur Unterscheidung zwischen den generierten Ressourcen den `LogicalId` verwenden.

OnSuccess (oder OnFailure) Die Eigenschaft ist für Amazon SQS SQS-Ereignisse angegeben

Wenn die Eigenschaft `OnSuccess` (oder`OnFailure`) der `DestinationConfig` Eigenschaft der `EventInvokeConfig` Eigenschaft von an angegeben `AWS::Serverless::Function` ist und der Zieltyp ist, SQS aber der Ziel-ARN nicht angegeben ist, werden die folgenden CloudFormation Ressourcen AWS SAM generiert: `AWS::Lambda::EventInvokeConfig` und`AWS::SQS::Queue`.

`AWS::Lambda::EventInvokeConfig`

LogicalId: *<function-LogicalId>*`EventInvokeConfig`

Referenzierbare Eigenschaft: N/A (Sie müssen den verwenden, um auf diese `LogicalId` Ressource zu verweisen CloudFormation)

`AWS::SQS::Queue`

LogicalId: *<function-LogicalId>OnSuccessQueue* (oder) *<function-LogicalId>*`OnFailureQueue`

Referenzierbares Eigentum: *<function-LogicalId>*`.DestinationQueue`

Wenn beide `OnSuccess` und für ein Amazon SQS SQS-Ereignis angegeben `OnFailure` sind, müssen Sie zur Unterscheidung zwischen den generierten Ressourcen den `LogicalId` verwenden.

CloudFormation Ressourcen, die generiert werdenAWS::Serverless::GraphQLApi, wenn angegeben

Wenn Sie eine `AWS::Serverless::GraphQLApi` Ressource in einer Vorlage AWS Serverless Application Model (AWS SAM) angeben, werden AWS SAM immer die folgenden AWS CloudFormation Basisressourcen erstellt.

`AWS::AppSync::DataSource`

`LogicalId`: *`<graphqlapi-LogicalId><datasource-RelativeId><datasource-Type>`*`DataSource`

Referenzierbare Eigenschaft: N/A (Sie müssen die verwenden, um auf diese `LogicalId` Ressource zu verweisen CloudFormation)

`AWS::AppSync::FunctionConfiguration`

`LogicalId`: *`<graphqlapi-LogicalId><function-RelativeId>`*

Referenzierbare Eigenschaft: N/A (Sie müssen das verwenden, um auf diese Ressource `LogicalId` zu verweisen) CloudFormation

`AWS::AppSync::GraphQLApi`

`LogicalId`: *`<graphqlapi-LogicalId>`*

Referenzierbare Eigenschaft: N/A (Sie müssen das verwenden, um auf diese Ressource `LogicalId` zu verweisen) CloudFormation

`AWS::AppSync::GraphQLSchema`

`LogicalId`: *`<graphqlapi-LogicalId>`*`Schema`

Referenzierbare Eigenschaft: N/A (Sie müssen das verwenden, um auf diese Ressource `LogicalId` zu verweisen) CloudFormation

`AWS::AppSync::Resolver`

`LogicalId`: *`<graphqlapi-LogicalId><OperationType><resolver-RelativeId>`*

Referenzierbare Eigenschaft: N/A (Sie müssen das verwenden, um auf diese Ressource `LogicalId` zu verweisen) CloudFormation

Zusätzlich zu diesen CloudFormation Ressourcen AWS SAM können, sofern `AWS::Serverless::GraphQLApi` angegeben, auch die folgenden CloudFormation Ressourcen generiert werden.

`AWS::AppSync::ApiCache`

`LogicalId`: *`<graphqlapi-LogicalId>`*`ApiCache`

Referenzierbare Eigenschaft: N/A (Sie müssen den verwenden, um auf diese `LogicalId` Ressource zu verweisen CloudFormation)

`AWS::AppSync::ApiKey`

`LogicalId`: *`<graphqlapi-LogicalId><apikey-RelativeId>`*

Referenzierbare Eigenschaft: N/A (Sie müssen das verwenden, um auf diese Ressource `LogicalId` zu verweisen) CloudFormation

`AWS::AppSync::DomainName`

`LogicalId`: *`<graphqlapi-LogicalId>`*`DomainName`

Referenzierbare Eigenschaft: N/A (Sie müssen das verwenden, um auf diese Ressource `LogicalId` zu verweisen) CloudFormation

`AWS::AppSync::DomainNameApiAssociation`

`LogicalId`: *`<graphqlapi-LogicalId>`*`DomainNameApiAssociation`

Referenzierbare Eigenschaft: N/A (Sie müssen das verwenden, um auf diese Ressource `LogicalId` zu verweisen) CloudFormation

AWS SAM kann die `AWS::Serverless::Connector` Ressource auch verwenden, um Berechtigungen bereitzustellen. Weitere Informationen finden Sie unter CloudFormation Ressourcen, die generiert werden, wenn Sie angeben AWS::Serverless::Connector.

CloudFormation Ressourcen, die generiert werden AWS::Serverless::HttpApi , wenn angegeben

Wenn an angegeben `AWS::Serverless::HttpApi` ist, generiert AWS Serverless Application Model (AWS SAM) eine `AWS::ApiGatewayV2::Api` CloudFormation Basisressource.

`AWS::ApiGatewayV2::Api`

`LogicalId`: *`<httpapi-LogicalId>`*

Referenzierbare Eigenschaft: N/A (Sie müssen den verwenden, um auf diese `LogicalId` Ressource zu verweisen CloudFormation)

Zusätzlich zu dieser CloudFormation Ressource werden, wenn `AWS::Serverless::HttpApi` angegeben, AWS SAM auch CloudFormation Ressourcen für die folgenden Szenarien generiert:

Szenarien

- StageName Eigenschaft ist angegeben
- StageName Eigenschaft ist nicht spezifiziert
- DomainName Eigenschaft ist spezifiziert

StageName Eigenschaft ist angegeben

Wenn die `StageName` Eigenschaft von an angegeben `AWS::Serverless::HttpApi` ist, wird die `AWS::ApiGatewayV2::Stage` CloudFormation Ressource AWS SAM generiert.

`AWS::ApiGatewayV2::Stage`

`LogicalId`: *`<httpapi-LogicalId><stage-name>`*`Stage`

*`<stage-name>`*ist die Zeichenfolge, auf die die `StageName` Eigenschaft gesetzt ist. Wenn Sie beispielsweise `StageName` auf einstellen`Gamma`, `LogicalId` ist dies: *`MyHttpApiGamma`* Phase.

Referenzierbare Eigenschaft: *`<httpapi-LogicalId>`*`.Stage`

StageName Eigenschaft ist nicht spezifiziert

Wenn die `StageName` Eigenschaft von an nicht angegeben `AWS::Serverless::HttpApi` ist, wird die `AWS::ApiGatewayV2::Stage` CloudFormation Ressource AWS SAM generiert.

`AWS::ApiGatewayV2::Stage`

`LogicalId`: *`<httpapi-LogicalId>`*`ApiGatewayDefaultStage`

Referenzierbare Eigenschaft: *`<httpapi-LogicalId>`*`.Stage`

DomainName Eigenschaft ist spezifiziert

Wenn die `DomainName` Eigenschaft der `Domain` Eigenschaft von an angegeben `AWS::Serverless::HttpApi` ist, wird die `AWS::ApiGatewayV2::DomainName` CloudFormation Ressource AWS SAM generiert.

`AWS::ApiGatewayV2::DomainName`

`LogicalId`: `ApiGatewayDomainNameV2`*`<sha>`*

*`<sha>`*ist ein eindeutiger Hashwert, der generiert wird, wenn der Stack erstellt wird. Beispiel, `ApiGatewayDomainNameV2`*`926eeb5ff1`*.

Referenzierbare Eigenschaft: *`<httpapi-LogicalId>`*`.DomainName`

CloudFormation Ressourcen, die generiert wurden, wenn AWS::Serverless::MicrovmImage ist angegeben

Wenn Sie eine angeben`AWS::Serverless::MicrovmImage`, generiert AWS Serverless Application Model (AWS SAM) eine `AWS::Lambda::MicrovmImage` CloudFormation Basisressource.

`AWS::Lambda::MicrovmImage`

`LogicalId`: *`<microvmimage-LogicalId>`*

Referenzierbare Eigenschaft: N/A (Sie müssen die verwenden, um auf diese CloudFormation Ressource `LogicalId` zu verweisen)

Wenn diese CloudFormation Ressource angegeben `AWS::Serverless::MicrovmImage` ist, werden zusätzlich zu dieser Ressource AWS SAM auch CloudFormation Ressourcen für die folgenden Szenarien generiert:

Szenarien

- BuildRoleArn Eigenschaft ist nicht angegeben

BuildRoleArn Eigenschaft ist nicht angegeben

Wenn Sie die `BuildRoleArn` Eigenschaft von nicht angeben`AWS::Serverless::MicrovmImage`, AWS SAM generiert eine `AWS::IAM::Role` CloudFormation Ressource mit einer Inline-Richtlinie, die die folgenden Berechtigungen gewährt:

- `s3:GetObject`— beschränkt auf den Amazon S3-Bucket von `CodeUri`
- `logs:CreateLogGroup`,`logs:CreateLogStream`, `logs:PutLogEvents` — für Build-Logs

Die Vertrauensrichtlinie für Rollen ermöglicht es dem `lambda.amazonaws.com` Dienstprinzipal, die Rolle zu übernehmen.

`AWS::IAM::Role`

`LogicalId`: *`<microvmimage-LogicalId>`*`BuildRole`

CloudFormation Ressourcen, die generiert wurden, wenn AWS::Serverless::NetworkConnector ist angegeben

Wenn Sie eine angeben`AWS::Serverless::NetworkConnector`, generiert AWS Serverless Application Model (AWS SAM) eine `AWS::Lambda::NetworkConnector` CloudFormation Basisressource.

`AWS::Lambda::NetworkConnector`

`LogicalId`: *`<networkconnector-LogicalId>`*

Referenzierbare Eigenschaft: N/A (Sie müssen die verwenden, um auf diese CloudFormation Ressource `LogicalId` zu verweisen)

Wenn diese CloudFormation Ressource angegeben `AWS::Serverless::NetworkConnector` ist, werden zusätzlich zu dieser Ressource AWS SAM auch CloudFormation Ressourcen für die folgenden Szenarien generiert:

Szenarien

- OperatorRole Eigenschaft ist nicht angegeben

OperatorRole Eigenschaft ist nicht angegeben

Wenn Sie die `OperatorRole` Eigenschaft eines nicht angeben`AWS::Serverless::NetworkConnector`, AWS SAM wird eine `AWS::IAM::Role` CloudFormation Ressource mit der angehängten `AWSLambdaNetworkConnectorOperatorPolicy` AWS verwalteten Richtlinie generiert.

Die Rollenvertrauensrichtlinie ermöglicht es dem `lambda.amazonaws.com` Dienstprinzipal, die Rolle zu übernehmen.

`AWS::IAM::Role`

`LogicalId`: *`<networkconnector-LogicalId>`*`OperatorRole`

CloudFormation Ressourcen, die generiert werden AWS::Serverless::WebSocketApi , wenn angegeben

Wenn an angegeben `AWS::Serverless::WebSocketApi` ist, generiert AWS Serverless Application Model (AWS SAM) eine `AWS::ApiGatewayV2::Api` CloudFormation Basisressource. Darüber hinaus generiert es immer eine `AWS::ApiGatewayV2::Stage` Ressource und AWS SAM generiert für jede in der `Routes` Eigenschaft definierte Route`AWS::ApiGatewayV2::Route`,`AWS::ApiGatewayV2::Integration`, und `AWS::Lambda::Permission` Ressourcen.

`AWS::ApiGatewayV2::Api`

`LogicalId`: *`<websocketapi-LogicalId>`*

Referenzierbare Eigenschaft: N/A (Sie müssen den verwenden, um auf diese `LogicalId` Ressource zu verweisen CloudFormation)

`AWS::ApiGatewayV2::Stage`

`LogicalId`: *`<websocketapi-LogicalId><stage-name>`*`Stage`

*`<stage-name>`*ist die Zeichenfolge, auf die die `StageName` Eigenschaft gesetzt ist. Wenn `StageName` nicht angegeben, ist der Standardwert`default`. Wenn Sie beispielsweise `StageName` auf einstellen`prod`, `LogicalId` ist der*`MyWebSocketApi`*`prodStage`.

Referenzierbare Eigenschaft: *`<websocketapi-LogicalId>`*`.Stage`

`AWS::ApiGatewayV2::Route`

`LogicalId`: *`<websocketapi-LogicalId><normalized-route-key>`*`Route`

*`<normalized-route-key>`*ist der Routenschlüssel ohne Sonderzeichen. Zum Beispiel ist für den Routenschlüssel `$connect` `LogicalId` der*`MyWebSocketApi`*`connectRoute`.

Referenzierbare Eigenschaft: N/A (Sie müssen den verwenden, um auf diese `LogicalId` Ressource zu verweisen CloudFormation)

`AWS::ApiGatewayV2::Integration`

`LogicalId`: *`<websocketapi-LogicalId><normalized-route-key>`*`Integration`

Referenzierbare Eigenschaft: N/A (Sie müssen das verwenden, um auf diese Ressource `LogicalId` zu verweisen) CloudFormation

`AWS::Lambda::Permission`

`LogicalId`: *`<websocketapi-LogicalId><normalized-route-key>`*`Permission`

Referenzierbare Eigenschaft: N/A (Sie müssen das verwenden, um auf diese Ressource `LogicalId` zu verweisen) CloudFormation

Wenn `AWS::Serverless::WebSocketApi` angegeben, werden zusätzlich zu diesen CloudFormation Ressourcen AWS SAM auch CloudFormation Ressourcen für die folgenden Szenarien generiert:

Szenarien

- Die Auth-Eigenschaft ist mit dem Wert AuthType CUSTOM angegeben
- DomainName Eigenschaft ist angegeben
- BasePath Eigenschaft ist angegeben
- Die Route53-Eigenschaft ist angegeben

Die Auth-Eigenschaft ist mit dem Wert AuthType CUSTOM angegeben

Wenn die `Auth` Eigenschaft von mit `AuthType` set to angegeben `AWS::Serverless::WebSocketApi` wird`CUSTOM`, wird eine `AWS::ApiGatewayV2::Authorizer` CloudFormation Ressource AWS SAM generiert. Wenn nicht angegeben, `InvokeRole` wird AWS SAM auch eine `AWS::Lambda::Permission` Ressource generiert, die es API Gateway ermöglicht, die Autorisierungsfunktion aufzurufen.

`AWS::ApiGatewayV2::Authorizer`

`LogicalId`: *`<websocketapi-LogicalId>`*`Authorizer`

Referenzierbare Eigenschaft: N/A (Sie müssen den verwenden, um auf diese Ressource `LogicalId` zu verweisen) CloudFormation

`AWS::Lambda::Permission`(wann **`InvokeRole`** ist nicht angegeben)

`LogicalId`: *`<websocketapi-LogicalId>`*`AuthorizerPermission`

Referenzierbare Eigenschaft: N/A (Sie müssen den verwenden, um auf diese `LogicalId` Ressource zu verweisen CloudFormation)

DomainName Eigenschaft ist angegeben

Wenn die `DomainName` Eigenschaft der `Domain` Eigenschaft von an angegeben `AWS::Serverless::WebSocketApi` ist, wird die `AWS::ApiGatewayV2::DomainName` CloudFormation Ressource AWS SAM generiert.

`AWS::ApiGatewayV2::DomainName`

`LogicalId`: `ApiGatewayDomainNameV2`*`<sha>`*

*`<sha>`*ist ein eindeutiger Hashwert, der bei der Erstellung des Stacks generiert wird. Beispiel: `ApiGatewayDomainNameV2`*`926eeb5ff1`*.

Referenzierbare Eigenschaft: N/A (Sie müssen den verwenden, um auf diese `LogicalId` Ressource zu verweisen CloudFormation)

BasePath Eigenschaft ist angegeben

Wenn die `BasePath` `Domain` Eigenschaft einer angegeben `AWS::Serverless::WebSocketApi` ist, werden `AWS::ApiGatewayV2::ApiMapping` CloudFormation Ressourcen AWS SAM generiert, eine für jeden angegebenen Basispfad.

`AWS::ApiGatewayV2::ApiMapping`

`LogicalId`: *`<websocketapi-LogicalId><basepath>`*`ApiMapping`

*`<basepath>`*ist der Basispfadwert. Wenn Sie beispielsweise `v1` als Basispfad angeben, `LogicalId` ist der*`MyWebSocketApi`*`v1ApiMapping`.

Referenzierbare Eigenschaft: N/A (Sie müssen den verwenden, um auf diese `LogicalId` Ressource zu verweisen CloudFormation)

Die Route53-Eigenschaft ist angegeben

Wenn die `Route53` Eigenschaft einer angegeben `AWS::Serverless::WebSocketApi` ist, wird eine `AWS::Route53::RecordSetGroup` CloudFormation Ressource AWS SAM generiert. `Domain`

`AWS::Route53::RecordSetGroup`

`LogicalId`: `RecordSetGroup`*`<sha>`*

*`<sha>`*ist ein eindeutiger Hashwert, der bei der Erstellung des Stacks generiert wird. Beispiel: `RecordSetGroup`*`926eeb5ff1`*.

Referenzierbare Eigenschaft: N/A (Sie müssen den verwenden, um auf diese `LogicalId` Ressource zu verweisen CloudFormation)

CloudFormation Ressourcen, die generiert werdenAWS::Serverless::LayerVersion, wenn angegeben

Wenn an angegeben `AWS::Serverless::LayerVersion` ist, generiert AWS Serverless Application Model (AWS SAM) eine `AWS::Lambda::LayerVersion` CloudFormation Basisressource.

`AWS::Lambda::LayerVersion`

`LogicalId`: *`<layerversion-LogicalId>`*

Referenzierbare Eigenschaft: N/A (Sie müssen den verwenden, um auf diese `LogicalId` Ressource zu verweisen CloudFormation)

CloudFormation Ressourcen, die generiert werdenAWS::Serverless::SimpleTable, wenn angegeben

Wenn an angegeben `AWS::Serverless::SimpleTable` ist, generiert AWS Serverless Application Model (AWS SAM) eine `AWS::DynamoDB::Table` CloudFormation Basisressource.

`AWS::DynamoDB::Table`

`LogicalId`: *`<simpletable-LogicalId>`*

Referenzierbare Eigenschaft: N/A (Sie müssen den verwenden, um auf diese `LogicalId` Ressource zu verweisen CloudFormation)

CloudFormation Ressourcen, die generiert werdenAWS::Serverless::StateMachine, wenn angegeben

Wenn an angegeben `AWS::Serverless::StateMachine` ist, generiert AWS Serverless Application Model (AWS SAM) eine `AWS::StepFunctions::StateMachine` CloudFormation Basisressource.

`AWS::StepFunctions::StateMachine`

`LogicalId`: *`<statemachine-LogicalId>`*

Referenzierbare Eigenschaft: N/A (Sie müssen den verwenden, um auf diese `LogicalId` Ressource zu verweisen CloudFormation)

Zusätzlich zu dieser CloudFormation Ressource werden, wenn `AWS::Serverless::StateMachine` angegeben, AWS SAM auch CloudFormation Ressourcen für die folgenden Szenarien generiert:

Szenarien

- Die Rolleneigenschaft ist nicht angegeben
- Eine API-Ereignisquelle ist angegeben
- Eine Ereignisquelle für eine Event-Bridge (oder einen Event-Bus) wurde angegeben

Die Rolleneigenschaft ist nicht angegeben

Wenn die `Role` Eigenschaft von nicht angegeben `AWS::Serverless::StateMachine` ist, wird eine `AWS::IAM::Role` CloudFormation Ressource AWS SAM generiert.

`AWS::IAM::Role`

`LogicalId`: *`<statemachine-LogicalId>`*`Role`

Referenzierbare Eigenschaft: N/A (Sie müssen den verwenden, um auf diese `LogicalId` Ressource zu verweisen CloudFormation)

Eine API-Ereignisquelle ist angegeben

Wenn die `Event` Eigenschaft von auf gesetzt `AWS::Serverless::StateMachine` ist`Api`, die `RestApiId` Eigenschaft aber nicht angegeben ist, wird die `AWS::ApiGateway::RestApi` CloudFormation Ressource AWS SAM generiert.

`AWS::ApiGateway::RestApi`

`LogicalId`: `ServerlessRestApi`

Referenzierbare Eigenschaft: N/A (Sie müssen den verwenden, um auf diese `LogicalId` Ressource zu verweisen CloudFormation)

Eine Ereignisquelle für eine Event-Bridge (oder einen Event-Bus) wurde angegeben

Wenn die `Event` Eigenschaft von auf einen der Typen Event Bridge (oder Event Bus) gesetzt `AWS::Serverless::StateMachine` ist, AWS SAM wird die `AWS::Events::Rule` CloudFormation Ressource generiert. Dies gilt für die folgenden Typen: `EventBridgeRuleSchedule`, und`CloudWatchEvents`.

`AWS::Events::Rule`

`LogicalId`: *`<statemachine-LogicalId><event-LogicalId>`*

Referenzierbare Eigenschaft: N/A (Sie müssen den verwenden, um auf diese `LogicalId` Ressource zu verweisen CloudFormation)

Ressourcenattribute, unterstützt von AWS SAM

Ressourcenattribute sind Attribute, die Sie hinzufügen können, AWS SAM und CloudFormation Ressourcen, um zusätzliche Verhaltensweisen und Beziehungen zu steuern. Weitere Informationen zu Ressourcenattributen finden Sie unter Referenz zu Ressourcenattributen im AWS CloudFormation Benutzerhandbuch.

AWS SAM unterstützt eine Teilmenge von Ressourcenattributen, die von CloudFormation definiert sind. Einige der unterstützten Ressourcenattribute werden nur auf die generierte CloudFormation Basisressource der entsprechenden AWS SAM Ressource kopiert, andere wiederum auf alle generierten CloudFormation Ressourcen, die sich aus der entsprechenden AWS SAM Ressource

ergeben. Weitere Hinweise zu CloudFormation Ressourcen, die aus entsprechenden AWS SAM Ressourcen generiert wurden, finden Sie unterGeneriert CloudFormation Ressourcen für AWS SAM.

In der folgenden Tabelle wird die Unterstützung von Ressourcenattributen zusammengefasst AWS SAM, abhängig von den unten Ausnahmen aufgeführten Kriterien.

Ressourcenattribute	Vom Ziel generierte Ressource (n)
DependsOn Metadaten [1, 2]	Nur von der Basis CloudFormation generierte Ressource. Informationen zur Zuordnung zwischen AWS SAM Ressourcen und CloudFormation Basisressourcen finden Sie unterGeneriert CloudFormation Ressourcenszenarien.
Zustand DeletionPolicy UpdateReplacePolicy	Alle generierten CloudFormation Ressourcen aus der entsprechenden AWS SAM Ressource. Hinweise zu Szenarien für generierte CloudFormation Ressourcen finden Sie unterGeneriert CloudFormation Ressourcenszenarien.

Hinweise:

1. Weitere Hinweise zur Verwendung des `Metadata` Ressourcenattributs mit dem `AWS::Serverless::Function` Ressourcentyp finden Sie unterErstellen von Lambda-Funktionen mit benutzerdefinierten Laufzeiten in AWS SAM.
2. Weitere Hinweise zur Verwendung des `Metadata` Ressourcenattributs mit dem `AWS::Serverless::LayerVersion` Ressourcentyp finden Sie unterAufbau von Lambda-Schichten in AWS SAM.

Ausnahmen

Es gibt eine Reihe von Ausnahmen zu den zuvor beschriebenen Regeln für Ressourcenattribute:

- Für `RetentionPolicy` legt `AWS::Lambda::LayerVersion` das benutzerdefinierte Feld AWS SAM-only das `DeletionPolicy` für die generierten CloudFormation Ressourcen fest. Dies hat eine höhere Priorität als `DeletionPolicy` es selbst. Wenn keiner von beiden festgelegt ist, `DeletionPolicy` ist standardmäßig auf `Retain` gesetzt.

- Wenn nicht angegeben, `DeletionPolicy` ist die Standardeinstellung`Retain`.
  `AWS::Lambda::Version`
- In dem Szenario, in dem dies für eine serverlose Funktion angegeben `DeploymentPreferences` ist, werden Ressourcenattribute nicht auf die folgenden generierten CloudFormation Ressourcen kopiert:
  - `AWS::CodeDeploy::Application`
  - `AWS::CodeDeploy::DeploymentGroup`
  - Der `AWS::IAM::Role` Name`CodeDeployServiceRole`, der für dieses Szenario erstellt wurde
- Wenn Ihre AWS SAM Vorlage mehrere Funktionen mit implizit erstellten API-Ereignisquellen enthält, teilen sich die Funktionen die generierte `AWS::ApiGateway::RestApi` Ressource. Wenn die Funktionen in diesem Szenario unterschiedliche Ressourcenattribute haben, werden die `AWS::ApiGateway::RestApi` Ressourcenattribute für die generierte Ressource gemäß den folgenden priorisierten Listen AWS SAM kopiert:
  - `UpdateReplacePolicy`:
    1. `Retain`
    2. `Snapshot`
    3. `Delete`
  - `DeletionPolicy`:
    1. `Retain`
    2. `Delete`

API-Gateway-Erweiterungen für AWS SAM

API Gateway Gateway-Erweiterungen, die speziell für entwickelt wurden AWS, bieten zusätzliche Anpassungen und Funktionen für Design und Verwaltung APIs. API Gateway Gateway-Erweiterungen sind Erweiterungen der OpenAPI-Spezifikation, die die AWS-spezifische Autorisierung und API Gateway Gateway-spezifische API-Integrationen unterstützen. Weitere Informationen zu API-Gateway-Erweiterungen finden Sie unter API-Gateway-Erweiterungen für OpenAPI.

AWS SAM unterstützt eine Teilmenge von API Gateway Gateway-Erweiterungen. In der folgenden Tabelle können Sie nachlesen AWS SAM, welche API-Gateway-Erweiterungen unterstützt werden.

API-Gateway-Erweiterung	Unterstützt von AWS SAM

x-amazon-apigateway-any-method Objekt	Ja
x-amazon-apigateway-api-key-source-Eigenschaft	Nein
x-amazon-apigateway-auth Objekt	Ja
x-amazon-apigateway-authorizer Objekt	Ja
x-amazon-apigateway-authtype Eigenschaft	Ja
x-amazon-apigateway-binary-media-types-Eigenschaft	Ja
x-amazon-apigateway-documentation Objekt	Nein
x-amazon-apigateway-endpoint-configuration-Objekt	Nein
x-amazon-apigateway-gatewayObjekt -responses	Ja
x-amazon-apigateway-gateway-Responses.GatewayResponse-Objekt	Ja
x-amazon-apigateway-gateway-Responses.ResponseParameters-Objekt	Ja
x-amazon-apigateway-gateway-Responses.ResponseTemplates-Objekt	Ja
x-amazon-apigateway-integration Objekt	Ja
x-amazon-apigateway-integration.requestTemplates-Objekt	Ja
x-amazon-apigateway-integration.RequestParameter-Objekt	Nein
x-amazon-apigateway-integration.responses-Objekt	Ja
x-amazon-apigateway-integration.response-Objekt	Ja
x-amazon-apigateway-integration.responseTemplates-Objekt	Ja
x-amazon-apigateway-integration.responseParameters-Objekt	Ja
x-amazon-apigateway-request-validator-Eigenschaft	Nein
x-amazon-apigateway-request-validators-Objekt	Nein

x-amazon-apigateway-request-Validators.RequestValidator-Objekt	Nein

Intrinsische Funktionen für AWS SAM

Systeminterne Funktionen sind integrierte Funktionen, mit denen Sie Eigenschaften Werte zuweisen können, die nur zur Laufzeit verfügbar sind. AWS SAM unterstützt bestimmte intrinsische Funktionseigenschaften nur begrenzt und ist daher nicht in der Lage, einige systeminterne Funktionen aufzulösen. Daher empfehlen wir, die `AWS::LanguageExtensions` Transformation hinzuzufügen, um dieses Problem zu beheben. Das `AWS::LanguageExtensions` ist ein von gehostetes Makro CloudFormation , mit dem Sie systeminterne Funktionen und andere Funktionen verwenden können, die standardmäßig nicht in enthalten sind. CloudFormation

```
Transform:
  - AWS::LanguageExtensions
  - AWS::Serverless-2016-10-31
```

> **Note**
>
> Wenn Sie systeminterne Funktionen in einer CodeUri Eigenschaft verwenden, können AWS SAM die Werte nicht korrekt analysiert werden. Erwägen Sie stattdessen die Verwendung von Transform`AWS::LanguageExtensions`.
> Weitere Informationen finden Sie im Abschnitt Eigenschaften von AWS::Serverless: :Function.

Einzelheiten darüber, wie der mit der `AWS::LanguageExtensions` Transformation AWS SAMCLI umgeht, einschließlich der Verwendung systemeigener Funktionen in `CodeUri` und anderer paketierbarer Eigenschaften, finden Sie unter. CloudFormation Unterstützung für Spracherweiterungen

Weitere Informationen zu systeminternen Funktionen finden Sie im Benutzerhandbuch unter Intrinsic Function Reference.AWS CloudFormation

CloudFormation Unterstützung für Spracherweiterungen

Das AWS SAMCLI unterstützt Vorlagen, die die `AWS::LanguageExtensions` Transformation verwenden`Fn::ForEach`, einschließlich`Fn::Length`,`Fn::ToJsonString`, und `Fn::FindInMap`

with`DefaultValue`. Vollständige Referenzinformationen zu diesen Konstrukten finden Sie unter AWS::LanguageExtensions Transform im AWS CloudFormation Benutzerhandbuch.

Informationen zur Verwendung von Spracherweiterungen in einer AWS SAM Vorlage finden Sie `AWS::LanguageExtensions` im vorherigen *`Transform`* `AWS::Serverless-2016-10-31` Abschnitt:

```
Transform:
  - AWS::LanguageExtensions
  - AWS::Serverless-2016-10-31
```

Wenn der `AWS::LanguageExtensions` im `Transform` Abschnitt einer Vorlage AWS SAMCLI erkannt wird und Sie sich für die lokale Verarbeitung entschieden haben, werden die Konstrukte der Spracherweiterungen lokal erweitert, bevor AWS SAM Transformationen ausgeführt werden. Dadurch kann`sam build`,`sam package`,`sam deploy`,`sam sync`, `sam validate`sam local invoke, und mit Vorlagen arbeiten `sam local start-api`sam local start-lambda, die diese Konstrukte verwenden.

Die lokale Verarbeitung ist standardmäßig deaktiviert. Wenn diese Option deaktiviert ist, AWS SAMCLI leitet der die Vorlage unverändert weiter und CloudFormation verarbeitet die `AWS::LanguageExtensions` Transformation serverseitig bei der Bereitstellung. Informationen zu Aktivierungsmethoden finden Sie unterSpracherweiterungen werden aktiviert.

Die Erweiterung erfolgt in zwei Phasen, wenn sie aktiviert ist:

1. Phase 1 (Spracherweiterungen) — `Fn::ForEach` Schleifen werden erweitert, systeminterne Funktionen werden soweit möglich aufgelöst und die Vorlage wird in den Standard umgewandelt. CloudFormation
2. Phase 2 (AWS SAM Transformation) — Die erweiterte Vorlage wird vom AWS SAM Übersetzer wie gewohnt verarbeitet.

Die ursprüngliche Vorlage (mit `Fn::ForEach` intakter Vorlage) wird für die CloudFormation Bereitstellung beibehalten, da die `AWS::LanguageExtensions` Transformation serverseitig CloudFormation verarbeitet wird.

Spracherweiterungen werden aktiviert

Die lokale Verarbeitung von `AWS::LanguageExtensions` erfolgt per Opt-In pro Befehl. Es gibt drei äquivalente Aktivierungsmethoden, die in der Reihenfolge ihrer Priorität aufgeführt sind:

1. CLIflag — einen einzigen Aufruf `--language-extensions` weiterleiten:

```
sam build --language-extensions
sam package --language-extensions ...
sam deploy --language-extensions ...
```

   `--no-language-extensions`deaktiviert explizit und überschreibt beide `samconfig.toml` sowie die unten beschriebene Umgebungsvariable.

2. **`samconfig.toml`**— Behält die Auswahl pro Projekt bei:

```
[default.build.parameters]
language_extensions = true

[default.package.parameters]
language_extensions = true

[default.deploy.parameters]
language_extensions = true

[default.sync.parameters]
language_extensions = true

[default.local_invoke.parameters]
language_extensions = true

[default.local_start_api.parameters]
language_extensions = true

[default.local_start_lambda.parameters]
language_extensions = true

[default.validate.parameters]
language_extensions = true
```

   Ein `samconfig.toml` Eintrag wird standardmäßig für den Befehl geladen, sodass er so wirkt, als ob das Flag übergeben worden wäre — und hat somit Vorrang vor der Umgebungsvariablen.

3. Umgebungsvariable — so gesetzt`SAM_CLI_ENABLE_LANGUAGE_EXTENSIONS=1`, dass sie für die aktuelle Shell aktiviert wird:

```
export SAM_CLI_ENABLE_LANGUAGE_EXTENSIONS=1
```

```
sam build
sam local invoke MyFunction
```

Wahrheitsgemäße Werte (ohne Berücksichtigung von Groß- und Kleinschreibung) sind`1`, und`true`. `yes` Alles andere, einschließlich der leeren Zeichenfolge, wird als ausgeschaltet behandelt. Die Umgebungsvariable wird nur dann konsultiert, wenn weder das CLI Flag noch einen Wert `samconfig.toml` festlegt.

> **Important**
>
> Jeder Befehl benötigt seine eigene Aktivierung. Die Übergabe `--language-extensions` an `sam build` wird nicht auf einen späteren Zeitpunkt übertragen `sam local invoke` — die lokale Verarbeitung wird pro Befehlsaufruf entschieden. Verwenden Sie die Umgebungsvariable oder einen `samconfig.toml` Eintrag, um Across-Befehle zu aktivieren, ohne das Flag zu wiederholen.

Fn:: ForEach

`Fn::ForEach`generiert mehrere Ressourcen, Bedingungen oder Ausgaben aus einer einzigen Vorlagendefinition:

```
Transform:
  - AWS::LanguageExtensions
  - AWS::Serverless-2016-10-31

Parameters:
  ServiceNames:
    Type: CommaDelimitedList
    Default: "Users,Orders,Products"

Resources:
  Fn::ForEach::Services:
    - Name
    - !Ref ServiceNames
    - ${Name}Function:
        Type: AWS::Serverless::Function
        Properties:
          Handler: index.handler
          Runtime: python3.12
```

```
            CodeUri: ./services/${Name}
```

Beim Ausführen wird dies auf`UsersFunction`, und`OrdersFunction`, `sam build` erweitert`ProductsFunction`, die jeweils aus ihrem jeweiligen Quellverzeichnis erstellt wurden.

Dynamische Artefakteigenschaften

Wenn eine paketierbare Eigenschaft eine Schleifenvariable verwendet (z. B.`./services/${Name}`), AWS SAMCLI generiert sie einen CloudFormation `Mappings` Abschnitt, der jeden Sammlungswert seinem Amazon S3 zuordnet. URI Der `Fn::ForEach` Text wird so umgeschrieben, dass er verwendet werden CloudFormation kann, `Fn::FindInMap` damit er bei der Bereitstellung das richtige Artefakt auflösen kann.

Bei den erkannten Artefakteigenschaften handelt es sich um dieselben, die heute neu geschrieben wurden. `sam package` Das beinhaltet:

Dynamische Artefakteigenschaften, erkannt von AWS SAM CLI

Ressourcentyp	Eigenschaft
`AWS::Serverless::Function`	`CodeUri`, `ImageUri`
`AWS::Serverless::LayerVersion`	`ContentUri`
`AWS::Serverless::Api`	`DefinitionUri`
`AWS::Serverless::HttpApi`	`DefinitionUri`
`AWS::Serverless::StateMachine`	`DefinitionUri`
`AWS::Serverless::GraphQLApi`	`SchemaUri` , `CodeUri`
`AWS::Serverless::Application`	`Location`
`AWS::Lambda::Function`	`Code`, `Code.ImageUri`
`AWS::Lambda::LayerVersion`	`Content`
`AWS::ApiGateway::RestApi`	`BodyS3Location`

Ressourcentyp	Eigenschaft
AWS::ApiGatewayV2::Api	BodyS3Location
AWS::AppSync::GraphQLSchema	DefinitionS3Location
AWS::AppSync::Resolver	RequestMappingTemplateS3Location , ResponseMappingTemplateS3Location , CodeS3Location
AWS::AppSync::FunctionConfiguration	RequestMappingTemplateS3Location , ResponseMappingTemplateS3Location , CodeS3Location
AWS::StepFunctions::StateMachine	DefinitionS3Location
AWS::ElasticBeanstalk::ApplicationVersion	SourceBundle
AWS::Glue::Job	Command.ScriptLocation
AWS::CloudFormation::Stack	TemplateURL
AWS::CloudFormation::StackSet	TemplateURL
AWS::CloudFormation::ModuleVersion	ModulePackage
AWS::CloudFormation::ResourceVersion	SchemaHandlerPackage

Bei Eigenschaften mit Punkten (z. B. `Command.ScriptLocation` on `AWS::Glue::Job` oder `Code.ImageUri` on`AWS::Lambda::Function`) wird der Wert an der verschachtelten Position auf der Ressource gelesen und geschrieben. Der generierte Mapping-Name verwendet nur das Blattsegment des Eigenschaftenpfads.

Wenn es sich bei der Eigenschaft um eine Loop-Vorlage handelt, lautet der Mapping-Name `SAM<LeafProperty><LoopName>` (zum Beispiel oder). `SAMCodeUriServices` `SAMScriptLocationJobs`

> **Important**
>
> Customer-authored Zuordnungen sollten nicht mit diesen `SAM*` Präfixen beginnen — sie sind reserviert für. AWS SAMCLI Siehe Einschränkungen.

Zum Beispiel nach: `sam package`

```
Mappings:
  SAMCodeUriServices:
    Users:
      CodeUri: s3://my-bucket/abc123
    Orders:
      CodeUri: s3://my-bucket/def456
    Products:
      CodeUri: s3://my-bucket/ghi789

Resources:
  Fn::ForEach::Services:
    - Name
    - !Ref ServiceNames
    - ${Name}Function:
        Type: AWS::Serverless::Function
        Properties:
          Handler: index.handler
          Runtime: python3.12
          CodeUri: !FindInMap [SAMCodeUriServices, !Ref Name, CodeUri]
```

Mehrere Ressourcen pro ForEach Körper

Ein einzelner `Fn::ForEach` Körper kann mehr als eine Ressource pro Iteration emittieren. Jede Ressource wird für jeden Sammlungswert generiert:

```
Resources:
  Fn::ForEach::Tables:
    - TableName
    - [Users, Orders, Products]
```

```
    - ${TableName}Table:
        Type: AWS::DynamoDB::Table
        Properties:
          TableName: !Sub "${AWS::StackName}-${TableName}"
          # ...

      ${TableName}StreamProcessor:
        Type: AWS::Serverless::Function
        Properties:
          CodeUri: stream-processors/${TableName}/
          Events:
            DDBStream:
              Type: DynamoDB
              Properties:
                Stream: !GetAtt
                  - !Sub "${TableName}Table"
                  - StreamArn
```

Zuordnung, Name, Kollision, Auflösung

Wenn zwei Ressourcen im selben `Fn::ForEach` Hauptteil dieselbe dynamische Artefakteigenschaft deklarieren (z. B. sowohl an als auch als `StateMachine` use`DefinitionUri`), wird ein `Api` Suffix AWS SAMCLI angehängt, das aus dem statischen Teil der Vorlage für die logische ID der Ressource übernommen wurde, um die Zuordnungsnamen eindeutig zu halten:

Beispiel für die Zuordnung von Namen mit Kollisionssuffixen

Vorlage für Ressourcen	Eigenschaft	Name der Zuordnung
`${Svc}Api`	`DefinitionUri`	`SAMDefinitionUriServicesApi`
`${Svc}StateMachine`	`DefinitionUri`	`SAMDefinitionUriServicesStateMachine`

Liegt keine Kollision vor, wird der Basisname (z. B.`SAMDefinitionUriServices`) verwendet.

Parameter-based Sammlungen

Wenn es sich bei der `Fn::ForEach` Sammlung um eine Parameterreferenz handelt (z. B.`!Ref ServiceNames`) und der Schleifenkörper eine dynamische Artefakteigenschaft verwendet (z. B.`CodeUri: ./services/${Name}`), AWS SAMCLI benötigt er die Sammlungswerte, um die

unter beschriebenen `SAM*` Zuordnungen zu generieren. Dynamische Artefakteigenschaften Sie werden bei der Verarbeitung der Vorlage wie folgt aufgelöst:

1. `--parameter-overrides`an den AWS SAMCLI Befehl übergeben.
2. Der `Default` Wert des Parameters in der Vorlage.

> **Important**
>
> Da die `SAM*` Mappings zum Zeitpunkt des Paketes integriert werden, müssen Sie jedes Mal, wenn Sie den Parameterwert ändern (z. B. wenn Sie einen neuen Dienst hinzufügen), neu verpacken, damit die Mappings Einträge für die neuen Werte enthalten. Dies gilt nur, wenn der Parameter eine dynamische Artefaktschleife steuert. Andere Parameterüberschreibungen können wie gewohnt bei der Bereitstellung geändert werden.

```
# Package with the values you intend to deploy with
sam package --language-extensions --parameter-overrides
 ServiceNames="Users,Orders,Products"

# Deploy with the same values
sam deploy --language-extensions --parameter-overrides
 ServiceNames="Users,Orders,Products"
```

Verschachtelte Stacks

`Fn::ForEach`in Nested Stack wird Templates (`AWS::CloudFormation::Stack`) unterstützt. Der AWS SAMCLI übergibt die `Parameters` Eigenschaft des übergeordneten Stacks an die Erweiterung der untergeordneten Vorlage, sodass untergeordnete `Fn::ForEach` Sammlungen, die auf vom Elternteil bereitgestellte Parameter verweisen, korrekt aufgelöst werden.

```
# parent.yaml
Resources:
  ChildStack:
    Type: AWS::CloudFormation::Stack
    Properties:
      TemplateURL: ./child.yaml
      Parameters:
        ServiceNames: "Users,Orders,Products"
```

Verschachteltes Fn: ForEach

Es werden bis zu 5 Verschachtelungsebenen unterstützt, die dem CloudFormation Limit entsprechen:

```
Resources:
  Fn::ForEach::Envs:
    - Env
    - [Dev, Staging, Prod]
    - Fn::ForEach::Services:
        - Svc
        - [Users, Orders]
        - ${Env}${Svc}Function:
            Type: AWS::Serverless::Function
            Properties:
              CodeUri: ./services/${Svc}
              Environment:
                Variables:
                  STAGE: ${Env}
```

ForEach in den Ausgängen

`Fn::ForEach`Blöcke werden auch innerhalb des `Outputs` Abschnitts erweitert, sodass Sie pro Sammlungswert eine Ausgabe ausgeben können:

```
Outputs:
  Fn::ForEach::FunctionArns:
    - Name
    - [alpha, beta]
    - ${Name}FunctionArn:
        Value: !GetAtt
          - !Sub "${Name}Function"
          - Arn
```

Bedingungen und DependsOn

Ressourcen, die von `Fn::ForEach` can emittiert werden, tragen `Condition` und `DependsOn` wie jede andere Ressource. Die Bedingung oder Abhängigkeit wird auf jede generierte Ressource repliziert:

```
Conditions:
  IsProd: !Equals [!Ref Environment, prod]
```

```
Resources:
  SharedTable:
    Type: AWS::DynamoDB::Table
    # ...

  Fn::ForEach::Functions:
    - Name
    - [api, worker]
    - ${Name}Function:
        Type: AWS::Serverless::Function
        Condition: IsProd
        DependsOn: SharedTable
        Properties:
          Handler: main.handler
          CodeUri: functions/${Name}/
```

&{Identifier} -Syntax

Die `&{identifier}` Syntax entfernt nicht-alphanumerische Zeichen aus dem ersetzten Wert, was nützlich ist, um gültige logische IDs aus Werten wie IP-Adressen zu generieren:

```
Fn::ForEach::Hosts:
  - IP
  - ["10.0.0.1", "10.0.0.2"]
  - Host&{IP}:
      Type: AWS::EC2::Instance
      # Expands to Host10001, Host10002
```

Unterstützte systeminterne Funktionen

Die folgenden systemeigenen Funktionen werden während der Erweiterung lokal gelöst:

Systeminterne Funktionen wurden bei der Erweiterung der Spracherweiterungen lokal behoben

Funktion	Description
`Fn::ForEach`	Loop-Erweiterung.
`Fn::Length`	Gibt die Anzahl der Listenelemente zurück.
`Fn::ToJsonString`	Konvertiert einen Wert in eine JSON Zeichenfolge.

Funktion	Description
`Fn::FindInMap`	Kartensuche, einschließlich der optionalen`DefaultValue` .
`Fn::If`	Bedingte Auswahl von Werten.
`Fn::Sub`	Ersetzung von Zeichenketten.
`Fn::Join`	Verkettung von Zeichenketten.
`Fn::Split`	Aufteilen von Zeichenketten.
`Fn::Select`	Elementauswahl auflisten.
`Fn::Base64`	Base64-Kodierung.
`Fn::Equals` , `Fn::And`, `Fn::Or`, `Fn::Not`	Bewertung des Zustands.
`Ref`	Parameter- und Pseudo-Parameter-Referenzen.

Funktionen, die bereitgestellte Ressourcen (`Fn::GetAtt`,,`Fn::GetAZs`) benötigen`Fn::ImportValue`, werden beibehalten, CloudFormation damit sie bei der Bereitstellung behoben werden können.

Validierungsfehler

Die folgenden Vorlagenprobleme werden lokal erkannt, bevor die AWS SAM Transformation ausgeführt wird:

Bei der Erweiterung der Spracherweiterungen sind Validierungsfehler aufgetreten

Ursache	Fehlermeldung
Der `Fn::ForEach` Wert ist falsch formatiert — es handelt sich nicht um eine Liste, er hat nicht genau 3 Elemente oder er hat einen Schleifen bezeichner, der keine Zeichenfolge ist.	`Fn::ForEach::<key> layout is incorrect` (ausgelöst als`InvalidTemplateException`).

Ursache	Fehlermeldung
Mehr als 5 Ebenen von `Fn::ForEach` sind verschachtelt.	`Fn::ForEach nesting depth of <N> exceeds the maximum allowed depth of 5. CloudFormation supports up to 5 nested Fn::ForEach loops.`
Die Sammlung wird in eine leere Liste aufgelöst (z. B. ein `CommaDelimitedList` Parameter mit`Default: ""`).	Kein Fehler — die Schleife wird stillschweigend übersprungen und es werden keine Ressourcen ausgegeben.
Die Punkte `!Ref` in der Sammlung befinden sich auf einem Parameter, der nicht in der Vorlage deklariert ist.	Kein Fehler — die ungelöste Referenz wird in der Vorlage beibehalten. CloudFormation Wird das Problem bei der Bereitstellung serverseitig lösen.

Einschränkungen

- Sammlungen müssen zu einem bestimmten Zeitpunkt build/package auflösbar sein. `Fn::ForEach`Sammlungen, `Fn::GetAttFn::ImportValue`, oder dynamische SSM/ Secrets Manager-Verweise verwenden, können nicht lokal erweitert werden. Verwenden Sie `--parameter-overrides` stattdessen einen Parameter mit.
- Dynamische Artefaktzuordnungen werden zur Paketzeit behoben. Wenn es sich bei einer `Fn::ForEach` Sammlung um eine Parameterreferenz handelt und der Loop-Body eine dynamische Artefakteigenschaft verwendet (z. B.`CodeUri: ./services/${Name}`), enthalten die generierten `SAM*` Mappings nur Einträge für die Parameterwerte, die zur Paketzeit aufgelöst wurden. Wenn Sie diesen Parameter bei der Bereitstellung ändern`--parameter-overrides`, ohne das Paket neu zu verpacken, enthalten die neuen Werte keine Mapping-Einträge und die Bereitstellung schlägt fehl. Dies gilt nicht für Parameter, die nicht zur Steuerung eines dynamischen `Fn::ForEach` Artefakts verwendet werden.
- **`DeletionPolicy`**und **`UpdateReplacePolicy`** werden während der Erweiterung validiert und behoben. Sie unterstützen zwei Parameter`Ref`, aber keine anderen systemeigenen Funktionen.
- Verschachtelungslimit. `Fn::ForEach`Es können bis zu 5 Ebenen verschachtelt werden, was dem CloudFormation serverseitigen Limit entspricht.

- Reservierte Zuordnungsnamen. Mapping-Namen, die mit einem der folgenden Zeichen beginnen, sind für die reserviert AWS SAMCLI — erstellen Sie keine eigenen Mappings mit diesen Präfixen:
  - `SAMCodeUri`,`SAMImageUri`,,`SAMContentUri`,`SAMDefinitionUri`,`SAMSchemaUri`,, `SAMBodyS3Location` `SAMDefinitionS3Location` `SAMTemplateURLSAMCode`, `SAMContent` — wird von `sam package` für dynamische Artefakteigenschaften ausgegeben. Sehen Sie sich die Dynamische Artefakteigenschaften Tabelle an.
  - `SAMLayers`— wird ausgegeben`sam build`, wenn eine `Fn::ForEach` -generierte Funktion automatisch generierte Referenzen auf Abhängigkeitsebenen aufnimmt (Lambda überlagert die Builds in einen verschachtelten Stapel). AWS SAMCLI Dieses Präfix hat keine entsprechende vom Benutzer verfasste Eigenschaft; es wird automatisch hinzugefügt.

Telemetrie

Das AWS SAMCLI gibt ein *`CFNLanguageExtensions`* Telemetrieereignis aus, wenn ein Befehl mit *`--language-extensions`* (oder einer entsprechenden Umgebungsvariablen) aufgerufen wird und die Vorlage die Transformation deklariert. `AWS::LanguageExtensions` Das Ereignis wird einmal pro Aufruf ausgelöst und es wird kein Vorlageninhalt übertragen. Wenn die lokale Verarbeitung ausgeschaltet ist (Standardeinstellung), wird kein Ereignis ausgelöst.

Beispiel für serverlose Anwendungen für AWS SAM

Dieser Abschnitt enthält zwei Beispielanwendungen: eine, die DynamoDB-Ereignisse verarbeitet, und eine andere, die Amazon S3 S3-Ereignisse verarbeitet. Jedes Beispiel führt Sie durch den step-by-step Prozess der Erstellung einer Anwendung. Darüber hinaus enthalten beide Informationen zur Konfiguration von Ereignisquellen und AWS Ressourcen. In beiden Fällen wird zunächst festgelegt, was getan werden muss, bevor Sie beginnen, und anschließend die Schritte zum Initialisieren, Testen, Verpacken und Bereitstellen Ihrer Anwendung beschrieben.

Themen

- DynamoDB-Ereignisse verarbeiten mit AWS SAM
- Verarbeiten Sie Amazon S3 S3-Ereignisse mit AWS SAM

DynamoDB-Ereignisse verarbeiten mit AWS SAM

Mit dieser Beispielanwendung bauen Sie auf dem auf, was Sie in der Übersicht und der Schnellstartanleitung gelernt haben, und installieren eine weitere Beispielanwendung. Diese Anwendung besteht aus einer Lambda-Funktion, die von einer DynamoDB-Tabellenereignisquelle aufgerufen wird. Die Lambda-Funktion ist sehr einfach: Sie protokolliert Daten, die über die Ereignisquellnachricht weitergeleitet wurden.

In dieser Übung erfahren Sie, wie Sie Nachrichten aus der Ereignisquelle nachahmen, die an Lambda-Funktionen übergeben werden, wenn sie aufgerufen werden.

Bevor Sie beginnen

Stellen Sie sicher, dass Sie das erforderliche Setup in der Installieren Sie das AWS SAMCLI abgeschlossen haben.

Schritt 1: Initialisieren Sie die Anwendung

In diesem Abschnitt laden Sie das Anwendungspaket herunter, das aus einer AWS SAM Vorlage und einem Anwendungscode besteht.

So initialisieren Sie die Anwendung

1. Führen Sie den folgenden Befehl an einer AWS SAMCLI Eingabeaufforderung aus.

```
sam init \
--location gh:aws-samples/cookiecutter-aws-sam-dynamodb-python \
--no-input
```

Beachten Sie, dass `gh:` der obige Befehl auf die GitHub URL erweitert wird`https://github.com/`.

2. Überprüfen Sie den Inhalt des Verzeichnisses, das der Befehl erstellt hat (`dynamodb_event_reader/`):

   - `template.yaml`— Definiert zwei AWS Ressourcen, die die Read DynamoDB-Anwendung benötigt: eine Lambda-Funktion und eine DynamoDB-Tabelle. Die Vorlage definiert auch die Zuweisung zwischen den beiden Ressourcen.
   - `read_dynamodb_event/`directory — Enthält den DynamoDB-Anwendungscode.

Schritt 2: Testen Sie die Anwendung lokal

Verwenden Sie für lokale Tests das, AWS SAMCLI um ein DynamoDB-Beispielereignis zu generieren und die Lambda-Funktion aufzurufen:

```
sam local generate-event dynamodb update | sam local invoke --event - ReadDynamoDBEvent
```

Der `generate-event` Befehl erstellt eine Quellnachricht für das Testereignis, ähnlich den Nachrichten, die erstellt werden, wenn alle Komponenten in der Cloud bereitgestellt werden. AWS Diese Ereignisquellnachricht wird an die Lambda-Funktion weitergeleitet. ReadDynamo DBEvent

Stellen Sie sicher, dass die erwarteten Meldungen auf der Grundlage des Quellcodes in der Konsole ausgegeben werden. `app.py`

Schritt 3: Verpacken Sie die Anwendung

Nachdem Sie Ihre Anwendung lokal getestet haben, verwenden Sie das, AWS SAMCLI um ein Bereitstellungspaket zu erstellen, mit dem Sie die Anwendung in der AWS Cloud bereitstellen.

So erstellen Sie ein Lambda-Bereitstellungspaket

1. Erstellen Sie einen S3-Bucket an dem Speicherort, an dem Sie den verpackten Code speichern möchten. Wenn Sie einen vorhandenen S3-Bucket verwenden möchten, überspringen Sie diesen Schritt.

```
aws s3 mb s3://bucketname
```

2. Erstellen Sie das Bereitstellungspaket, indem Sie den folgenden `package` CLI-Befehl an der Befehlszeile ausführen.

```
sam package \
    --template-file template.yaml \
    --output-template-file packaged.yaml \
    --s3-bucket bucketname
```

Sie geben die neue Vorlagendatei an`packaged.yaml`, wenn Sie die Anwendung im nächsten Schritt bereitstellen.

Schritt 4: Stellen Sie die Anwendung bereit

Nachdem Sie das Bereitstellungspaket erstellt haben, verwenden Sie es, um die Anwendung in der AWS Cloud bereitzustellen. Anschließend testen Sie die Anwendung.

Um die serverlose Anwendung in der AWS Cloud bereitzustellen

- Verwenden Sie in der AWS SAMCLI den `deploy` CLI-Befehl, um alle Ressourcen bereitzustellen, die Sie in der Vorlage definiert haben.

```
sam deploy \
    --template-file packaged.yaml \
    --stack-name sam-app \
    --capabilities CAPABILITY_IAM \
    --region us-east-1
```

Im Befehl ermöglicht der `--capabilities` Parameter AWS CloudFormation die Erstellung einer IAM-Rolle.

CloudFormation erstellt die AWS Ressourcen, die in der Vorlage definiert sind. Sie können in der CloudFormation Konsole auf die Namen dieser Ressourcen zugreifen.

Um die serverlose Anwendung in der AWS Cloud zu testen

1. Öffnen Sie die DynamoDB-Konsole.
2. Fügen Sie einen Datensatz in die Tabelle ein, die Sie gerade erstellt haben.
3. Gehen Sie zur Registerkarte Metriken der Tabelle und wählen Sie Alle CloudWatch Metriken anzeigen aus. Wählen Sie in der CloudWatch Konsole Logs aus, um die Protokollausgabe anzeigen zu können.

Nächste Schritte

Weitere Informationen AWS SAM finden Sie in den folgenden Ressourcen:

- Serverless Land — Website, die serverlose Muster, Codebeispiele und Lernressourcen speziell für AWS SAM-Anwendungen zusammenfasst.

Verarbeiten Sie Amazon S3 S3-Ereignisse mit AWS SAM

Mit dieser Beispielanwendung bauen Sie auf dem auf, was Sie in den vorherigen Beispielen gelernt haben, und installieren eine komplexere Anwendung. Diese Anwendung besteht aus einer Lambda-Funktion, die von einer Amazon S3 S3-Objekt-Upload-Ereignisquelle aufgerufen wird. In dieser Übung erfahren Sie, wie Sie über eine Lambda-Funktion auf AWS Ressourcen zugreifen und AWS Serviceanrufe tätigen.

Diese serverlose Beispielanwendung verarbeitet Ereignisse zur Objekterstellung in Amazon S3. Für jedes Bild, das in einen Bucket hochgeladen wird, erkennt Amazon S3 das vom Objekt erstellte Ereignis und ruft eine Lambda-Funktion auf. Die Lambda-Funktion ruft Amazon Rekognition auf, um Text im Bild zu erkennen. Anschließend werden die von Amazon Rekognition zurückgegebenen Ergebnisse in einer DynamoDB-Tabelle gespeichert.

> **Note**
>
> Mit dieser Beispielanwendung führen Sie die Schritte in einer etwas anderen Reihenfolge aus als in den vorherigen Beispielen. Der Grund dafür ist, dass in diesem Beispiel AWS Ressourcen erstellt und IAM-Berechtigungen konfiguriert werden müssen, bevor Sie die Lambda-Funktion lokal testen können. Wir werden sie nutzen, um die Ressourcen CloudFormation zu erstellen und die Berechtigungen für Sie zu konfigurieren. Andernfalls müssten Sie dies manuell tun, bevor Sie die Lambda-Funktion lokal testen können.

Da dieses Beispiel komplizierter ist, sollten Sie sicherstellen, dass Sie mit der Installation der vorherigen Beispielanwendungen vertraut sind, bevor Sie diese ausführen.

Bevor Sie beginnen

Stellen Sie sicher, dass Sie das erforderliche Setup in der abgeschlossen habenInstallieren Sie das AWS SAMCLI.

Schritt 1: Initialisieren Sie die Anwendung

In diesem Abschnitt laden Sie die Beispielanwendung herunter, die aus einer AWS SAM Vorlage und einem Anwendungscode besteht.

So initialisieren Sie die Anwendung

1. Führen Sie den folgenden Befehl an einer AWS SAMCLI Eingabeaufforderung aus.

```
sam init \
--location https://github.com/aws-samples/cookiecutter-aws-sam-s3-rekognition-
dynamodb-python \
--no-input
```

2. Überprüfen Sie den Inhalt des Verzeichnisses, das der Befehl erstellt hat (`aws_sam_ocr/`):

   - `template.yaml`— Definiert drei AWS Ressourcen, die die Amazon S3 S3-Anwendung benötigt: eine Lambda-Funktion, einen Amazon S3 S3-Bucket und eine DynamoDB-Tabelle. Die Vorlage definiert auch die Zuordnungen und Berechtigungen zwischen diesen Ressourcen.
   - `src/`Verzeichnis — Enthält den Amazon S3 S3-Anwendungscode.
   - `SampleEvent.json`— Die Beispiel-Eventquelle, die für lokale Tests verwendet wird.

Schritt 2: Verpacken Sie die Anwendung

Bevor Sie diese Anwendung lokal testen können, müssen Sie das verwenden, AWS SAMCLI um ein Bereitstellungspaket zu erstellen, mit dem Sie die Anwendung in der AWS Cloud bereitstellen. Diese Bereitstellung erstellt die erforderlichen AWS Ressourcen und Berechtigungen, die zum lokalen Testen der Anwendung erforderlich sind.

So erstellen Sie ein Lambda-Bereitstellungspaket

1. Erstellen Sie einen S3-Bucket an dem Speicherort, an dem Sie den verpackten Code speichern möchten. Wenn Sie einen vorhandenen S3-Bucket verwenden möchten, überspringen Sie diesen Schritt.

   ```
   aws s3 mb s3://bucketname
   ```

2. Erstellen Sie das Bereitstellungspaket, indem Sie den folgenden `package` CLI-Befehl an der Befehlszeile ausführen.

   ```
   sam package \
       --template-file template.yaml \
       --output-template-file packaged.yaml \
       --s3-bucket bucketname
   ```

   Sie geben die neue Vorlagendatei an`packaged.yaml`, wenn Sie die Anwendung im nächsten Schritt bereitstellen.

Schritt 3: Stellen Sie die Anwendung bereit

Nachdem Sie das Bereitstellungspaket erstellt haben, verwenden Sie es, um die Anwendung in der AWS Cloud bereitzustellen. Anschließend testen Sie die Anwendung, indem Sie sie in der AWS Cloud aufrufen.

Um die serverlose Anwendung in der Cloud bereitzustellen AWS

- Verwenden Sie in der den `deploy` Befehl AWS SAMCLI, um alle Ressourcen bereitzustellen, die Sie in der Vorlage definiert haben.

  ```
  sam deploy \
      --template-file packaged.yaml \
      --stack-name aws-sam-ocr \
      --capabilities CAPABILITY_IAM \
      --region us-east-1
  ```

  Im Befehl ermöglicht der `--capabilities` Parameter CloudFormation das Erstellen einer IAM-Rolle.

CloudFormation erstellt die AWS Ressourcen, die in der Vorlage definiert sind. Sie können in der CloudFormation Konsole auf die Namen dieser Ressourcen zugreifen.

Um die serverlose Anwendung in der AWS Cloud zu testen

1. Laden Sie ein Bild in den Amazon S3 S3-Bucket hoch, den Sie für diese Beispielanwendung erstellt haben.
2. Öffnen Sie die DynamoDB-Konsole und suchen Sie die Tabelle, die erstellt wurde. In der Tabelle finden Sie die von Amazon Rekognition zurückgegebenen Ergebnisse.
3. Stellen Sie sicher, dass die DynamoDB-Tabelle neue Datensätze enthält, die Text enthalten, den Amazon Rekognition im hochgeladenen Bild gefunden hat.

Schritt 4: Testen Sie die Anwendung lokal

Bevor Sie die Anwendung lokal testen können, müssen Sie zunächst die Namen der AWS Ressourcen abrufen, die von erstellt wurden CloudFormation.

- Rufen Sie den Amazon S3 S3-Schlüsselnamen und den Bucket-Namen von ab CloudFormation. Ändern Sie die `SampleEvent.json` Datei, indem Sie die Werte für den Objektschlüssel, den Bucket-Namen und den Bucket-ARN ersetzen.
- Rufen Sie den DynamoDB-Tabellennamen ab. Dieser Name wird für den folgenden `sam local invoke` Befehl verwendet.

Verwenden Sie das AWS SAMCLI, um ein Amazon S3 S3-Beispielereignis zu generieren und die Lambda-Funktion aufzurufen:

```
TABLE_NAME=Table name obtained from CloudFormation console sam local invoke --event
 SampleEvent.json
```

Der `TABLE_NAME=` Teil legt den DynamoDB-Tabellennamen fest. Der `--event` Parameter gibt die Datei an, die die Testereignisnachricht enthält, die an die Lambda-Funktion übergeben werden soll.

Sie können jetzt überprüfen, ob die erwarteten DynamoDB-Datensätze auf der Grundlage der von Amazon Rekognition zurückgegebenen Ergebnisse erstellt wurden.

Nächste Schritte

Weitere Informationen finden Sie in den AWS SAM folgenden Ressourcen:

- Serverless Land — Website, die serverlose Muster, Codebeispiele und Lernressourcen speziell für AWS SAM-Anwendungen zusammenfasst.

Entwickeln Sie Ihre serverlose Anwendung mit AWS SAM

Dieser Abschnitt enthält Themen zur Validierung Ihrer AWS SAM Vorlage und zum Erstellen Ihrer Anwendung mit Abhängigkeiten. Es enthält auch Themen AWS SAM zur Verwendung für bestimmte Anwendungsfälle, z. B. die Arbeit mit Lambda-Schichten, die Verwendung verschachtelter Anwendungen, die Steuerung des Zugriffs auf API Gateway APIs, die Orchestrierung von AWS Ressourcen mit Step Functions und die Codesignatur Ihrer Anwendungen. Die drei wichtigsten Meilensteine, die Sie für die Entwicklung Ihrer Anwendung erreichen müssen, sind unten aufgeführt.

Themen

- Erstellen Sie Ihre Bewerbung in AWS SAM
- Definieren Sie Ihre Infrastruktur mit AWS SAM
- Erstellen Sie Ihre Anwendung mit AWS SAM

Erstellen Sie Ihre Bewerbung in AWS SAM

Nachdem Sie die Lektüre „Erste Schritte“ abgeschlossen habenWie benutzt man AWS Serverless Application Model (AWS SAM), sind Sie bereit, ein AWS SAM Projekt in Ihrer Entwicklerumgebung zu erstellen. Ihr AWS SAM Projekt dient als Ausgangspunkt für das Schreiben Ihrer serverlosen Anwendung. Eine Liste der AWS SAMCLI `sam init` Befehlsoptionen finden Sie untersam init.

Der AWS Serverless Application Model Befehl Command Line Interface (AWS SAMCLI) bietet Optionen zur Initialisierung einer neuen serverlosen Anwendung, die aus `sam init` folgenden Komponenten besteht:

- Eine AWS SAM Vorlage zur Definition Ihres Infrastrukturcodes.
- Eine Ordnerstruktur, die Ihre Anwendung organisiert.
- Konfiguration für Ihre AWS Lambda Funktionen.

Informationen zum Erstellen eines AWS SAM Projekts finden Sie in den Themen in diesen Abschnitten.

> **Note**
>
> `sam init`enthält Projektvorlagen für dauerhafte Funktionen in unterstützten Laufzeiten (TypeScript, Python, Java). Diese Vorlagen bieten Startcode und Konfiguration für die Erstellung von statusbehafteten serverlosen Anwendungen.

Topics

- Initialisieren Sie eine neue serverlose Anwendung
- Optionen für Sam Init
- Fehlerbehebung
- Beispiele
- Weitere Informationen
- Nächste Schritte

Initialisieren Sie eine neue serverlose Anwendung

Um eine neue serverlose Anwendung mit dem zu initialisieren AWS SAMCLI

1. `cd`in ein Startverzeichnis.
2. Führen Sie in der Befehlszeile Folgendes aus:

```
$ sam init
```

3. Der AWS SAMCLI führt Sie durch einen interaktiven Ablauf zur Erstellung einer neuen serverlosen Anwendung.

> **Note**
>
> Wie unter beschriebenTutorial: Stellen Sie eine Hello World-Anwendung bereit mit AWS SAM, initialisiert dieser Befehl Ihre serverlose Anwendung und erstellt Ihr Projektverzeichnis. Dieses Verzeichnis wird mehrere Dateien und Ordner enthalten. Die wichtigste Datei ist`template.yaml`. Das ist deine AWS SAM Vorlage. Ihre Version von Python muss mit der Version von Python übereinstimmen, die in der `template.yaml` Datei aufgeführt ist, die der sam init Befehl erstellt hat.

Wählen Sie eine Startvorlage

Eine Vorlage besteht aus den folgenden Komponenten:

1. Eine AWS SAM Vorlage für Ihren Infrastrukturcode.
2. Ein Startprojektverzeichnis, das Ihre Projektdateien organisiert. Dies kann beispielsweise Folgendes beinhalten:
   a. Eine Struktur für Ihren Lambda-Funktionscode und seine Abhängigkeiten.
   b. Ein `events` Ordner, der Testereignisse für lokale Tests enthält.
   c. Ein `tests` Ordner zur Unterstützung von Komponententests.
   d. Eine `samconfig.toml` Datei zur Konfiguration der Projekteinstellungen.
   e. Eine `ReadMe` Datei und andere grundlegende Startprojektdateien.

   Im Folgenden finden Sie ein Beispiel für ein Startprojektverzeichnis:

```
sam-app
### README.md
### __init__.py
### events
#   ### event.json
### hello_world
#   ### __init__.py
#   ### app.py
#   ### requirements.txt
### samconfig.toml
### template.yaml
### tests
    ### __init__.py
    ### integration
    #   ### __init__.py
    #   ### test_api_gateway.py
    ### requirements.txt
    ### unit
        ### __init__.py
        ### test_handler.py
```

Sie können aus einer Liste verfügbarer AWS Schnellstartvorlagen auswählen oder Ihren eigenen Speicherort für benutzerdefinierte Vorlagen angeben.

Um eine AWS Schnellstart-Vorlage auszuwählen

1. Wenn Sie dazu aufgefordert werden, wählen Sie AWS Schnellstartvorlagen aus.
2. Wählen Sie zunächst eine AWS Schnellstart-Vorlage aus. Im Folgenden wird ein Beispiel gezeigt:

```
Which template source would you like to use?
    1 - AWS Quick Start Templates
    2 - Custom Template Location
Choice: 1

Choose an AWS Quick Start application template
    1 - Hello World Example
    2 - Multi-step workflow
    3 - Serverless API
    4 - Scheduled task
    5 - Standalone function
    6 - Data processing
    7 - Hello World Example With Powertools
    8 - Infrastructure event management
    9 - Serverless Connector Hello World Example
    10 - Multi-step workflow with Connectors
    11 - Lambda EFS example
    12 - DynamoDB Example
    13 - Machine Learning
Template: 4
```

Um Ihren eigenen Speicherort für benutzerdefinierte Vorlagen auszuwählen

1. Wenn Sie dazu aufgefordert werden, wählen Sie den Speicherort für benutzerdefinierte Vorlagen aus.

```
Which template source would you like to use?
    1 - AWS Quick Start Templates
    2 - Custom Template Location
Choice: 2
```

2. Sie AWS SAMCLI werden aufgefordert, einen Speicherort für die Vorlage anzugeben.

```
Template location (git, mercurial, http(s), zip, path):
```

Geben Sie einen der folgenden Speicherorte für Ihr ZIP-Vorlagenarchiv an:

- GitHubRepository — Der Pfad zur .zip-Datei in Ihrem GitHub Repository. Die Datei muss sich im Stammverzeichnis Ihres Repositorys befinden.
- MercurialRepository — Der Pfad zur .zip-Datei in Ihrem Mercurial Repository. Die Datei muss sich im Stammverzeichnis Ihres Repositorys befinden.
- .zip-Pfad — Ein HTTPS-Pfad oder ein lokaler Pfad zu Ihrer .zip-Datei.

3. Dadurch AWS SAMCLI wird Ihre serverlose Anwendung mithilfe Ihrer benutzerdefinierten Vorlage initialisiert.

Wählen Sie eine Laufzeit

Wenn Sie eine AWS Schnellstartvorlage auswählen, werden Sie AWS SAMCLI aufgefordert, eine Laufzeit für Ihre Lambda-Funktionen auszuwählen. Die von der angezeigte Liste der Optionen entspricht den AWS SAMCLI Laufzeiten, die von Lambda nativ unterstützt werden.

- Die Laufzeit stellt eine sprachspezifische Umgebung bereit, die in der Ausführungsumgebung ausgeführt wird.
- Bei der Bereitstellung auf dem AWS Cloud ruft der Lambda-Service Ihre Funktion in einer Ausführungsumgebung auf.

Sie können jede andere Programmiersprache mit einer benutzerdefinierten Laufzeit verwenden. Dazu müssen Sie Ihre Startanwendungsstruktur manuell erstellen. Anschließend können Sie Ihre Anwendung `sam init` schnell initialisieren, indem Sie einen benutzerdefinierten Speicherort für Vorlagen konfigurieren.

Aus Ihrer Auswahl AWS SAMCLI erstellt der das Startverzeichnis für Ihren Lambda-Funktionscode und die Abhängigkeiten.

Wenn Lambda mehrere Abhängigkeitsmanager für Ihre Laufzeit unterstützt, werden Sie aufgefordert, Ihren bevorzugten Abhängigkeitsmanager auszuwählen.

Wählen Sie einen Pakettyp

Wenn Sie eine AWS Schnellstartvorlage und eine Runtime auswählen, werden Sie AWS SAMCLI aufgefordert, einen Pakettyp auszuwählen. Der Pakettyp bestimmt, wie Ihre Lambda-Funktionen für

die Verwendung mit dem Lambda-Service bereitgestellt werden. Die beiden unterstützten Pakettypen sind:

1. Container-Image — Enthält das Basisbetriebssystem, die Laufzeit, Lambda-Erweiterungen, Ihren Anwendungscode und dessen Abhängigkeiten.
2. ZIP-Dateiarchiv — Enthält Ihren Anwendungscode und seine Abhängigkeiten.

Weitere Informationen zu Bereitstellungspakettypen finden Sie unter Lambda-Bereitstellungspakete im AWS Lambda Entwicklerhandbuch.

Im Folgenden finden Sie ein Beispiel für die Verzeichnisstruktur einer Anwendung mit einer Lambda-Funktion, die als Container-Image verpackt ist. Das AWS SAMCLI lädt das Bild herunter und erstellt `Dockerfile` im Verzeichnis der Funktion ein, um das Bild zu spezifizieren.

```
sam-app
### README.md
### __init__.py
### events
#   ### event.json
### hello_world
#   ### Dockerfile
#   ### __init__.py
#   ### app.py
#   ### requirements.txt
### samconfig.toml
### template.yaml
### tests
    ### __init__.py
    ### unit
        ### __init__.py
        ### test_handler.py
```

Im Folgenden finden Sie ein Beispiel für die Verzeichnisstruktur einer Anwendung mit einer Funktion, die als ZIP-Dateiarchiv verpackt ist.

```
sam-app
### README.md
### __init__.py
### events
#   ### event.json
### hello_world
```

```
#   ### __init__.py
#   ### app.py
#   ### requirements.txt
### samconfig.toml
### template.yaml
### tests
    ### __init__.py
    ### integration
    #   ### __init__.py
    #   ### test_api_gateway.py
    ### requirements.txt
    ### unit
        ### __init__.py
        ### test_handler.py
```

Konfigurieren Sie die Ablaufverfolgung AWS X-Ray

Sie können wählen, ob Sie die AWS X-Ray Ablaufverfolgung aktivieren möchten. Weitere Informationen finden Sie unter Was ist AWS X-Ray? im AWS X-Ray Entwicklerhandbuch.

Wenn Sie es aktivieren, AWS SAMCLI konfiguriert das Ihre AWS SAM Vorlage. Im Folgenden wird ein Beispiel gezeigt:

```
Globals:
  Function:
    ...
    Tracing: Active
  Api:
    TracingEnabled: True
```

Konfigurieren Sie die Überwachung mit Amazon CloudWatch Application Insights

Sie können wählen, ob Sie die Überwachung mit Amazon CloudWatch Application Insights aktivieren möchten. Weitere Informationen finden Sie unter Amazon CloudWatch Application Insights im CloudWatch Amazon-Benutzerhandbuch.

Wenn Sie aktivieren, AWS SAMCLI konfiguriert der Ihre AWS SAM Vorlage. Im Folgenden wird ein Beispiel gezeigt:

```
Resources:
  ApplicationResourceGroup:
    Type: AWS::ResourceGroups::Group
```

```
    Properties:
      Name:
        Fn::Join:
        - ''
        - - ApplicationInsights-SAM-
          - Ref: AWS::StackName
      ResourceQuery:
        Type: CLOUDFORMATION_STACK_1_0
  ApplicationInsightsMonitoring:
    Type: AWS::ApplicationInsights::Application
    Properties:
      ResourceGroupName:
        Fn::Join:
        - ''
        - - ApplicationInsights-SAM-
          - Ref: AWS::StackName
      AutoConfigurationEnabled: 'true'
    DependsOn: ApplicationResourceGroup
```

Benennen Sie Ihre Anwendung

Geben Sie einen Namen für Ihre Anwendung ein. Der AWS SAMCLI erstellt unter diesem Namen einen Ordner auf oberster Ebene für Ihre Anwendung.

Optionen für Sam Init

Im Folgenden sind einige der wichtigsten Optionen aufgeführt, die Sie mit dem `sam init` Befehl verwenden können. Eine Liste aller Optionen finden Sie untersam init.

Initialisieren Sie eine Anwendung mithilfe eines benutzerdefinierten Vorlagenspeicherorts

Verwenden Sie die `--location` Option und geben Sie einen unterstützten Speicherort für benutzerdefinierte Vorlagen an. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam init --location https://github.com/aws-samples/sessions-with-aws-sam/raw/master/
starter-templates/web-app.zip
```

Initialisieren Sie eine Anwendung ohne den interaktiven Ablauf

Verwenden Sie die `--no-interactive` Option und geben Sie Ihre Konfigurationsoptionen in der Befehlszeile an, um den interaktiven Ablauf zu überspringen. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam init --no-interactive --runtime go1.x --name go-demo --dependency-manager mod --
app-template hello-world
```

Fehlerbehebung

Informationen zur Behebung von Problemen finden Sie unterAWS SAMCLIProblembehandlung. AWS SAMCLI

Beispiele

Initialisieren Sie eine neue serverlose Anwendung mithilfe der Hello World AWS Starter-Vorlage

Dieses Beispiel finden Sie Schritt 1: Initialisieren Sie die Hello World-Beispielanwendung im Tutorial: Bereitstellen einer Hello World-Anwendung.

Initialisieren Sie eine neue serverlose Anwendung mit einem benutzerdefinierten Speicherort für Vorlagen

Im Folgenden finden Sie Beispiele für die Bereitstellung eines GitHub Speicherorts für Ihre benutzerdefinierte Vorlage:

```
$ sam init --location gh:aws-samples/cookiecutter-aws-sam-python
$ sam init --location git+sh://git@github.com/aws-samples/cookiecutter-aws-sam-
python.git
$ sam init --location hg+ssh://hg@bitbucket.org/repo/template-name
```

Im Folgenden finden Sie ein Beispiel für einen lokalen Dateipfad:

```
$ sam init --location /path/to/template.zip
```

Im Folgenden finden Sie ein Beispiel für einen Pfad, der über HTTPS erreichbar ist:

```
$ sam init --location https://github.com/aws-samples/sessions-with-aws-sam/raw/master/
starter-templates/web-app.zip
```

Weitere Informationen

Weitere Informationen zur Verwendung des `sam init` Befehls finden Sie in den folgenden Abschnitten:

- Lernen AWS SAM: sam init — Reihe Serverless Land „Lernen AWS SAM“ amYouTube.
- Strukturierung serverloser Anwendungen für die Verwendung mit den AWS SAMCLI (Sessions with SAM S2E7) — Sessions mit der Serie On. AWS SAM YouTube

Nächste Schritte

Nachdem Sie Ihr AWS SAM Projekt erstellt haben, können Sie mit der Erstellung Ihrer Anwendung beginnen. Definieren Sie Ihre Infrastruktur mit AWS SAMAusführliche Anweisungen zu den Aufgaben, die Sie dazu erledigen müssen, finden Sie unter.

Definieren Sie Ihre Infrastruktur mit AWS SAM

Nachdem Sie Ihr Projekt erstellt haben, sind Sie bereit, Ihre Anwendungsinfrastruktur mit zu definieren AWS SAM. Konfigurieren Sie dazu Ihre AWS SAM Vorlage so, dass sie die Ressourcen und Eigenschaften Ihrer Anwendung definiert. Dabei handelt es sich um die `template.yaml` Datei in Ihrem AWS SAM Projekt.

Die Themen in diesem Abschnitt enthalten Inhalte zur Definition Ihrer Infrastruktur in Ihrer AWS SAM Vorlage (Ihrer `template.yaml` Datei). Es enthält auch Themen zur Definition von Ressourcen für bestimmte Anwendungsfälle, z. B. zur Arbeit mit Lambda-Layern, zur Verwendung verschachtelter Anwendungen, zur Steuerung des Zugriffs auf API Gateway APIs, zur Orchestrierung von AWS Ressourcen mit Step Functions, zur Codesignatur Ihrer Anwendungen und zur Validierung Ihrer Vorlage. AWS SAM

Themen

- Definieren Sie Anwendungsressourcen in Ihrer AWS SAM Vorlage
- Richten Sie den Ressourcenzugriff in Ihrer AWS SAM Vorlage ein und verwalten Sie ihn
- Kontrollieren Sie den API-Zugriff mit Ihrer AWS SAM Vorlage
- Steigern Sie die Effizienz mithilfe von Lambda-Schichten mit AWS SAM
- Verwenden Sie Code und Ressourcen mithilfe verschachtelter Anwendungen in AWS SAM
- Verwalten Sie zeitbasierte Ereignisse mit dem EventBridge Scheduler in AWS SAM

- Orchestrierung von AWS SAM Ressourcen mit AWS Step Functions
- Richten Sie die Codesignatur für Ihre AWS SAM Anwendung ein
- AWS SAM Vorlagendateien validieren

Definieren Sie Anwendungsressourcen in Ihrer AWS SAM Vorlage

Sie definieren die AWS Ressourcen, die Ihre serverlose Anwendung verwendet, im `Resources` Abschnitt Ihrer AWS SAM Vorlage. Wenn Sie eine Ressource definieren, legen Sie fest, was die Ressource ist, wie sie mit anderen Ressourcen interagiert und wie auf sie zugegriffen werden kann (d. h. die Berechtigungen der Ressource).

Der `Resources` Abschnitt Ihrer AWS SAM Vorlage kann eine Kombination aus CloudFormation Ressourcen und AWS SAM Ressourcen enthalten. Darüber hinaus können Sie die Kurzsyntax für die folgenden Ressourcen verwenden AWS SAM:

AWS SAM kurze Syntax	Was macht es mit einer verwandten Ressource AWS
AWS::Serverless::Api	Erstellt eine Sammlung von API-Gateway-Ressourcen und -Methoden, die über HTTPS-Endpunkte aufgerufen werden können.
AWS::Serverless::Application	Bettet eine serverlose Anwendung aus dem AWS Serverless Application Repository oder aus einem Amazon S3 S3-Bucket als verschachtelte Anwendung ein.
AWS::Serverless::Connector	Konfiguriert Berechtigungen zwischen zwei Ressourcen. Eine Einführung in Konnektoren finden Sie unterVerwaltung von Ressourcenberechtigungen mit AWS SAM Steckverbinder.
AWS::Serverless: :Funktion	Erstellt eine AWS Lambda Funktion, eine AWS Identity and Access Management (IAM-) Ausführungsrolle und Zuordnungen von Ereignisquellen, die die Funktion auslösen.

AWS SAM kurze Syntax	Was macht es mit einer verwandten Ressource AWS
AWS::Serverless::GraphQLApi	erstellt und konfiguriert eine AWS AppSync GraphQL API für Ihre serverlose Anwendung.
AWS::Serverless::HttpApi	Erstellt eine Amazon API Gateway Gateway-HTTP-API, mit der Sie RESTful APIs mit geringerer Latenz und geringeren Kosten als REST erstellen können APIs.
AWS::Serverless::LayerVersion	Erzeugt ein Lambda LayerVersion , das Bibliotheks- oder Laufzeitcode enthält, der von einer Lambda-Funktion benötigt wird.
AWS::Serverless::SimpleTable	Erstellt eine DynamoDB-Tabelle mit einem einzelnen Attribut-Primärschlüssel.
AWS::Serverless::StateMachine	Erstellt eine AWS Step Functions Zustandsmaschine, mit der Sie AWS Lambda Funktionen und andere AWS Ressourcen orchestrieren können, um komplexe und robuste Workflows zu bilden.

Die oben genannten Ressourcen sind auch unter aufgeführtAWS SAM Ressourcen und Immobilien.

Referenzinformationen zu allen AWS Ressourcen- und Eigenschaftstypen CloudFormation sowie deren AWS SAM Unterstützung finden Sie unter Referenz zu AWS Ressourcen- und Eigenschaftstypen im AWS CloudFormation Benutzerhandbuch.

Richten Sie den Ressourcenzugriff in Ihrer AWS SAM Vorlage ein und verwalten Sie ihn

Damit Ihre AWS Ressourcen miteinander interagieren können, müssen der richtige Zugriff und die entsprechenden Berechtigungen zwischen Ihren Ressourcen konfiguriert werden. Dazu ist die Konfiguration von AWS Identity and Access Management (IAM-) Benutzern, Rollen und Richtlinien erforderlich, um Ihre Interaktion auf sichere Weise durchzuführen.

Die Themen in diesem Abschnitt beziehen sich alle auf die Einrichtung des Zugriffs auf die in Ihrer Vorlage definierten Ressourcen. Dieser Abschnitt beginnt mit allgemeinen bewährten Methoden. In den nächsten beiden Themen werden zwei Optionen behandelt, mit denen Sie Zugriff und Berechtigungen für die Ressourcen einrichten können, auf die in Ihrer serverlosen Anwendung verwiesen wird: AWS SAM Konnektoren und AWS SAM Richtlinienvorlagen. Das letzte Thema enthält Einzelheiten zur Verwaltung des Benutzerzugriffs mithilfe derselben Mechanik CloudFormation wie bei der Benutzerverwaltung.

Weitere Informationen finden Sie unter Steuern des Zugriffs mit AWS Identity and Access Management im AWS CloudFormation Benutzerhandbuch.

AWS Serverless Application Model (AWS SAM) bietet zwei Optionen, die die Zugriffs- und Berechtigungsverwaltung für Ihre serverlosen Anwendungen vereinfachen.

1. AWS SAM Anschlüsse
2. AWS SAM Richtlinienvorlagen

AWS SAM Anschlüsse

Konnektoren sind eine Möglichkeit, Berechtigungen zwischen zwei Ressourcen bereitzustellen. Dazu beschreiben Sie in Ihrer AWS SAM Vorlage, wie sie miteinander interagieren sollen. Sie können entweder mit dem `Connectors` Ressourcenattribut oder dem `AWS::Serverless::Connector` Ressourcentyp definiert werden. Konnektoren unterstützen die Bereitstellung von `Read` und `Write` den Zugriff auf Daten und Ereignisse zwischen einer Kombination von AWS Ressourcen. Weitere Informationen zu AWS SAM Konnektoren finden Sie unterVerwaltung von Ressourcenberechtigungen mit AWS SAM Steckverbinder.

AWS SAM Richtlinienvorlagen

AWS SAM Richtlinienvorlagen sind vordefinierte Berechtigungssätze, die Sie Ihren AWS SAM Vorlagen hinzufügen können, um den Zugriff und die Berechtigungen zwischen Ihren AWS Lambda Funktionen, AWS Step Functions Zustandsmaschinen und den Ressourcen, mit denen sie interagieren, zu verwalten. Weitere Informationen zu AWS SAM Richtlinienvorlagen finden Sie unterAWS SAM Richtlinienvorlagen.

AWS CloudFormation Mechanismen

CloudFormation Zu den Mechanismen gehört die Konfiguration von IAM-Benutzern, -Rollen und Richtlinien zur Verwaltung von Berechtigungen zwischen Ihren AWS Ressourcen. Weitere

Informationen hierzu finden Sie unter Verwaltung von AWS SAM Berechtigungen mit CloudFormation Mechanismen.

Best Practices

In Ihren serverlosen Anwendungen können Sie mehrere Methoden verwenden, um Berechtigungen zwischen Ihren Ressourcen zu konfigurieren. Daher können Sie für jedes Szenario die beste Option auswählen und in Ihren Anwendungen mehrere Optionen zusammen verwenden. Hier sind einige Dinge, die Sie bei der Auswahl der für Sie besten Option beachten sollten:

- AWS SAM Sowohl Konnektoren als auch Richtlinienvorlagen reduzieren den IAM-Bedarf an Fachwissen, das erforderlich ist, um sichere Interaktionen zwischen Ihren AWS Ressourcen zu ermöglichen. Verwenden Sie Konnektoren und Richtlinienvorlagen, sofern diese unterstützt werden.
- AWS SAM Konnektoren bieten eine einfache und intuitive Kurzsyntax zur Definition von Berechtigungen in Ihren AWS SAM Vorlagen und erfordern das geringste Maß an IAM-Fachwissen. Wenn sowohl AWS SAM Konnektoren als auch Richtlinienvorlagen unterstützt werden, verwenden Sie AWS SAM Konnektoren.
- AWS SAM Konnektoren können Daten `Read` und Ereignisse zwischen unterstützten AWS SAM Quell- und Zielressourcen bereitstellen und `Write` darauf zugreifen. Eine Liste der unterstützten Ressourcen finden Sie unterAWS SAM Steckverbinderreferenz. Verwenden Sie AWS SAM Konnektoren, sofern dies unterstützt wird.
- Während AWS SAM Richtlinienvorlagen auf Berechtigungen zwischen Ihren Lambda-Funktionen, Step Functions Functions-Zustandsmaschinen und den AWS Ressourcen, mit denen sie interagieren, beschränkt sind, unterstützen Richtlinienvorlagen alle CRUD-Operationen. Verwenden AWS SAM Sie Richtlinienvorlagen, sofern sie unterstützt werden und wenn eine AWS SAM Richtlinienvorlage für Ihr Szenario verfügbar ist. Eine Liste der verfügbaren Richtlinienvorlagen finden Sie unterAWS SAM Richtlinienvorlagen.
- Verwenden Sie für alle anderen Szenarien oder wenn Granularität erforderlich ist, CloudFormation Mechanismen.

Verwaltung von Ressourcenberechtigungen mit AWS SAM Steckverbinder

Konnektoren sind ein abstrakter Ressourcentyp AWS Serverless Application Model (AWS SAM)`AWS::Serverless::Connector`, der einfache und gut abgegrenzte Berechtigungen zwischen Ihren serverlosen Anwendungsressourcen bereitstellt.

Vorteile von AWS SAM Steckverbinder

Durch die automatische Erstellung der entsprechenden Zugriffsrichtlinien zwischen Ressourcen können Sie mithilfe von Konnektoren Ihre serverlosen Anwendungen erstellen und sich auf Ihre Anwendungsarchitektur konzentrieren, ohne dass Sie Kenntnisse in den Bereichen AWS Autorisierungsfunktionen, Richtliniensprache und dienstspezifische Sicherheitseinstellungen benötigen. Daher sind Konnektoren ein großer Vorteil für Entwickler, die mit der serverlosen Entwicklung noch nicht vertraut sind, oder für erfahrene Entwickler, die ihre Entwicklungsgeschwindigkeit erhöhen möchten.

Verwenden AWS SAM Steckverbinder

Verwenden Sie das `Connectors` Ressourcenattribut, indem Sie es in eine Quellressource einbetten. Definieren Sie anschließend Ihre Zielressource und beschreiben Sie, wie Daten oder Ereignisse zwischen diesen Ressourcen fließen sollen. AWS SAM erstellt dann die Zugriffsrichtlinien, die zur Erleichterung der erforderlichen Interaktionen erforderlich sind.

Im Folgenden wird beschrieben, wie dieses Ressourcenattribut geschrieben wird:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  <source-resource-logical-id>:
    Type: <resource-type>
    ...
    Connectors:
      <connector-name>:
        Properties:
          Destination:
            <properties-that-identify-destination-resource>
          Permissions:
            <permission-types-to-provision>
  ...
```

Wie funktionieren Konnektoren

> **Note**
>
> In diesem Abschnitt wird erklärt, wie Konnektoren die erforderlichen Ressourcen hinter den Kulissen bereitstellen. Dies geschieht automatisch für Sie, wenn Sie Konnektoren verwenden.

Zunächst wird das eingebettete `Connectors` Ressourcenattribut in einen `AWS::Serverless::Connector` Ressourcentyp umgewandelt. Seine logische ID wird automatisch als erstellt*<source-resource-logical-id><embedded-connector-logical-id>*.

Hier ist zum Beispiel ein eingebetteter Konnektor:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Lambda::Function
    Connectors:
      MyConn:
        Properties:
          Destination:
            Id: MyTable
          Permissions:
            - Read
            - Write
  MyTable:
    Type: AWS::DynamoDB::Table
```

Dadurch wird die folgende `AWS::Serverless::Connector` Ressource generiert:

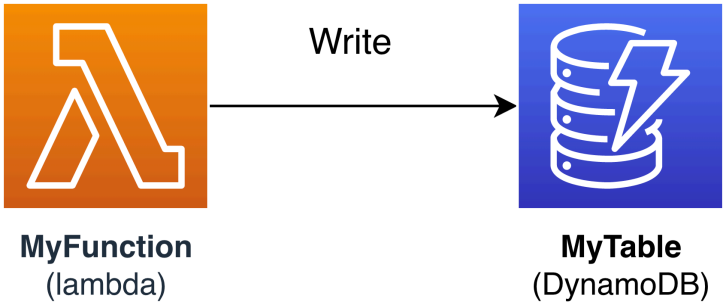
```
Transform: AWS::Serverless-2016-10-31
Resources:
  ...
  MyFunctionMyConn:
    Type: AWS::Serverless::Connector
    Properties:
      Source:
        Id: MyFunction
      Destination:
        Id: MyTable
      Permissions:
        - Read
        - Write
```

> **Note**
>
> Mithilfe dieser Syntax können Sie in Ihrer AWS SAM Vorlage auch Konnektoren definieren. Dies wird empfohlen, wenn Ihre Quellressource in einer anderen Vorlage als Ihr Connector definiert ist.

Als Nächstes werden die erforderlichen Zugriffsrichtlinien für diese Verbindung automatisch erstellt. Weitere Informationen zu den von Konnektoren generierten Ressourcen finden Sie unterCloudFormation Ressourcen, die generiert werden, wenn Sie angeben AWS::Serverless::Connector.

Beispiel für Konnektoren

Das folgende Beispiel zeigt, wie Sie Konnektoren verwenden können, um Daten aus einer AWS Lambda Funktion in eine Amazon DynamoDB-Tabelle zu schreiben.




```
Transform: AWS::Serverless-2016-10-31
Resources:
  MyTable:
    Type: AWS::Serverless::SimpleTable
  MyFunction:
    Type: AWS::Serverless::Function
    Connectors:
      MyConn:
        Properties:
          Destination:
            Id: MyTable
          Permissions:
            - Write
    Properties:
      Runtime: nodejs16.x
      Handler: index.handler
      InlineCode: |
```

```
        const AWS = require("aws-sdk");
        const docClient = new AWS.DynamoDB.DocumentClient();
        exports.handler = async (event, context) => {
          await docClient.put({
            TableName: process.env.TABLE_NAME,
            Item: {
              id: context.awsRequestId,
              event: JSON.stringify(event)
            }
          }).promise();
        }
      Environment:
        Variables:
          TABLE_NAME: !Ref MyTable
```

Das `Connectors` Ressourcenattribut ist in die Quellressource der Lambda-Funktion eingebettet. Die DynamoDB-Tabelle wird mithilfe der Eigenschaft als Zielressource definiert. `Id` Konnektoren stellen `Write` Berechtigungen zwischen diesen beiden Ressourcen bereit.

Wenn Sie Ihre AWS SAM Vorlage auf bereitstellen CloudFormation, AWS SAM werden automatisch die erforderlichen Zugriffsrichtlinien erstellt, die für das Funktionieren dieser Verbindung erforderlich sind.

Unterstützte Verbindungen zwischen Quell- und Zielressourcen

Konnektoren unterstützen `Write` Daten `Read` - und Ereignisberechtigungstypen zwischen einer ausgewählten Kombination von Quell- und Zielressourcenverbindungen. Konnektoren unterstützen beispielsweise eine `Write` Verbindung zwischen einer `AWS::ApiGateway::RestApi` Quellressource und einer `AWS::Lambda::Function` Zielressource.

Quell- und Zielressourcen können mithilfe einer Kombination unterstützter Eigenschaften definiert werden. Die Eigenschaftsanforderungen hängen von der Verbindung ab, die Sie herstellen und davon, wo die Ressourcen definiert sind.

> **Note**
>
> Connectoren können Berechtigungen zwischen unterstützten serverlosen und nicht serverlosen Ressourcentypen bereitstellen.

Eine Liste der unterstützten Ressourcenverbindungen und ihrer Eigenschaftsanforderungen finden Sie unter. Unterstützte Quell- und Zielressourcentypen für Konnektoren

Definieren Sie Lese- und Schreibberechtigungen in AWS SAM

In AWS SAM`Read` und `Write` Berechtigungen können innerhalb eines einzigen Connectors bereitgestellt werden:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Lambda::Function
    Connectors:
      MyTableConn:
        Properties:
          Destination:
            Id: MyTable
          Permissions:
            - Read
            - Write
  MyTable:
    Type: AWS::DynamoDB::Table
```

Weitere Informationen zur Verwendung von Konnektoren finden Sie unter. AWS SAM Steckverbinderreferenz

Definieren Sie Ressourcen mithilfe anderer unterstützter Eigenschaften in AWS SAM

Verwenden Sie die `Id` Eigenschaft sowohl für Quell- als auch für Zielressourcen, sofern sie in derselben Vorlage definiert sind. Optional `Qualifier` kann eine hinzugefügt werden, um den Umfang Ihrer definierten Ressource einzuschränken. Wenn sich die Ressource nicht in derselben Vorlage befindet, verwenden Sie eine Kombination unterstützter Eigenschaften.

- Eine Liste der unterstützten Eigenschaftskombinationen für Quell- und Zielressourcen finden Sie unterUnterstützte Quell- und Zielressourcentypen für Konnektoren.
- Eine Beschreibung der Eigenschaften, die Sie mit Konnektoren verwenden können, finden Sie unterAWS::Serverless::Connector.

Wenn Sie eine Quellressource mit einer anderen Eigenschaft als definieren`Id`, verwenden Sie die `SourceReference` Eigenschaft.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  <source-resource-logical-id>:
    Type: <resource-type>
    ...
    Connectors:
      <connector-name>:
        Properties:
          SourceReference:
            Qualifier: <optional-qualifier>
            <other-supported-properties>
          Destination:
            <properties-that-identify-destination-resource>
          Permissions:
            <permission-types-to-provision>
```

Hier ist ein Beispiel, bei dem a verwendet wird, `Qualifier` um den Umfang einer Amazon API Gateway Gateway-Ressource einzuschränken:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyApi:
    Type: AWS::Serverless::Api
    Connectors:
      ApiToLambdaConn:
        Properties:
          SourceReference:
            Qualifier: Prod/GET/foobar
          Destination:
            Id: MyFunction
          Permissions:
            - Write
  ...
```

Hier ist ein Beispiel, bei dem eine unterstützte Kombination von `Arn` und verwendet wird`Type`, um eine Zielressource aus einer anderen Vorlage zu definieren:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Connectors:
      TableConn:
        Properties:
          Destination:
            Type: AWS::DynamoDB::Table
            Arn: !GetAtt MyTable.Arn
  ...
```

Weitere Informationen zur Verwendung von Konnektoren finden Sie unterAWS SAM Steckverbinderreferenz.

Erstellen Sie mehrere Konnektoren aus einer einzigen Quelle in AWS SAM

Innerhalb einer Quellressource können Sie mehrere Konnektoren mit jeweils einer anderen Zielressource definieren.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Connectors:
      BucketConn:
        Properties:
          Destination:
            Id: amzn-s3-demo-bucket
          Permissions:
            - Read
            - Write
      SQSConn:
        Properties:
          Destination:
            Id: MyQueue
```

```
          Permissions:
            - Read
            - Write
      TableConn:
        Properties:
          Destination:
            Id: MyTable
          Permissions:
            - Read
            - Write
      TableConnWithTableArn:
        Properties:
          Destination:
            Type: AWS::DynamoDB::Table
            Arn: !GetAtt MyTable.Arn
          Permissions:
            - Read
            - Write
...
```

Weitere Informationen zur Verwendung von Konnektoren finden Sie unterAWS SAM Steckverbinderreferenz.

Erstellen Sie Konnektoren mit mehreren Zielen in AWS SAM

Innerhalb einer Quellressource können Sie einen einzelnen Konnektor mit mehreren Zielressourcen definieren. Hier ist ein Beispiel für eine Lambda-Funktionsquellenressource, die eine Verbindung zu einem Amazon Simple Storage Service (Amazon S3) -Bucket und einer DynamoDB-Tabelle herstellt:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Connectors:
      WriteAccessConn:
        Properties:
          Destination:
            - Id: OutputBucket
            - Id: CredentialTable
          Permissions:
            - Write
```

```
  ...
  OutputBucket:
    Type: AWS::S3::Bucket
  CredentialTable:
    Type: AWS::DynamoDB::Table
```

Weitere Informationen zur Verwendung von Konnektoren finden Sie unter. AWS SAM Steckverbinderreferenz

Definieren Sie Ressourcenattribute mit Konnektoren in AWS SAM

Ressourcenattribute können für Ressourcen definiert werden, um zusätzliche Verhaltensweisen und Beziehungen zu spezifizieren. Weitere Informationen zu Ressourcenattributen finden Sie unter Referenz zu Ressourcenattributen im AWS CloudFormation Benutzerhandbuch.

Sie können Ihrem eingebetteten Konnektor Ressourcenattribute hinzufügen, indem Sie sie auf derselben Ebene wie Ihre Konnektoreigenschaften definieren. Wenn Ihre AWS SAM Vorlage bei der Bereitstellung transformiert wird, werden die Attribute an die generierten Ressourcen weitergegeben.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Connectors:
      MyConn:
        DeletionPolicy: Retain
        DependsOn: AnotherFunction
        Properties:
          ...
```

Weitere Informationen zur Verwendung von Konnektoren finden Sie unterAWS SAM Steckverbinderreferenz.

Weitere Informationen

Weitere Informationen zur Verwendung von AWS SAM Konnektoren finden Sie in den folgenden Themen:

- AWS::Serverless::Connector
- Definieren Sie Lese- und Schreibberechtigungen in AWS SAM

- Definieren Sie Ressourcen mithilfe anderer unterstützter Eigenschaften in AWS SAM
- Erstellen Sie mehrere Konnektoren aus einer einzigen Quelle in AWS SAM
- Erstellen Sie Konnektoren mit mehreren Zielen in AWS SAM
- Definieren Sie Lese- und Schreibberechtigungen in AWS SAM
- Definieren Sie Ressourcenattribute mit Konnektoren in AWS SAM

Bereitstellen von Feedback

Wenn Sie Feedback zu Konnektoren geben möchten, reichen Sie ein neues Problem im AWS GitHubServerless-Application-Model-Repository ein.

AWS SAM Richtlinienvorlagen

Mit AWS Serverless Application Model (AWS SAM) können Sie aus einer Liste von Richtlinienvorlagen auswählen, um die Berechtigungen Ihrer Lambda-Funktionen und AWS Step Functions Zustandsmaschinen auf die Ressourcen zu beschränken, die von Ihrer Anwendung verwendet werden.

AWS SAM Für Anwendungen in der AWS Serverless Application Repository , die Richtlinienvorlagen verwenden, ist keine besondere Bestätigung durch den Kunden erforderlich, um die Anwendung von zu implementieren. AWS Serverless Application Repository

Wenn Sie das Hinzufügen einer neuen Richtlinienvorlage anfordern möchten, führen Sie die folgenden Schritte aus:

1. Reichen Sie eine Pull-Anfrage für die Quelldatei policy_templates.json im Branch des Projekts ein. `develop` AWS SAM GitHub Sie finden die Quelldatei in policy_templates.json auf der Website. GitHub
2. Reichen Sie im AWS SAM GitHub Projekt ein Problem ein, das die Gründe für Ihre Pull-Anfrage und einen Link zur Anfrage enthält. Verwenden Sie diesen Link, um ein neues Problem einzureichen AWS Serverless Application Model: Probleme.

Syntax

Für jede Richtlinienvorlage, die Sie in Ihrer AWS SAM Vorlagendatei angeben, müssen Sie immer ein Objekt angeben, das die Platzhalterwerte der Richtlinienvorlage enthält. Wenn für eine Richtlinienvorlage keine Platzhalterwerte erforderlich sind, müssen Sie ein leeres Objekt angeben.

YAML

```
MyFunction:
  Type: AWS::Serverless::Function
  Properties:
    Policies:
      - PolicyTemplateName1:        # Policy template with placeholder value
          Key1: Value1
      - PolicyTemplateName2: {}     # Policy template with no placeholder value
```

> **Note**
>
> Wenn Sie eine reguläre IAM-Richtlinie eingerichtet oder Richtlinien über Lambda verwaltet haben, kann die Richtlinienvorlage eingerichtet werden, ohne ein leeres Objekt zu verwenden.

Beispiele

Beispiel 1: Richtlinienvorlage mit Platzhalterwerten

Das folgende Beispiel zeigt, dass die SQSPollerPolicy-Richtlinienvorlage als Ressource einen `QueueName` erwartet. Die AWS SAM Vorlage ruft den Namen der "`MyQueue`" Amazon SQS SQS-Warteschlange ab, die Sie in derselben Anwendung erstellen oder als Parameter für die Anwendung anfordern können.

```
MyFunction:
  Type: 'AWS::Serverless::Function'
  Properties:
    CodeUri: ${codeuri}
    Handler: hello.handler
    Runtime: python2.7
    Policies:
      - SQSPollerPolicy:
          QueueName:
            !GetAtt MyQueue.QueueName
```

Beispiel 2: Richtlinienvorlage ohne Platzhalterwerten

Das folgende Beispiel enthält die CloudWatchPutMetricPolicy-Richtlinienvorlage ohne Platzhalterwerte.

> **Note**
>
> Auch wenn es keine Platzhalterwerte gibt, müssen Sie ein leeres Objekt angeben, da andernfalls ein Fehler auftritt.

```
MyFunction:
  Type: 'AWS::Serverless::Function'
  Properties:
    CodeUri: ${codeuri}
    Handler: hello.handler
    Runtime: python2.7
    Policies:
      - CloudWatchPutMetricPolicy: {}
```

Beispiel 3: Richtlinienvorlage mit Platzhalterwerten und einer regulären IAM-Richtlinie

Das folgende Beispiel enthält die Amazon SQSFull Access-Richtlinie und die DynamoDBCrudPolicy Richtlinienvorlage. Die Amazon SQSFull Access-Richtlinie ist eine IAM-Richtlinie und keine AWS SAM Richtlinie. Sie müssen also kein leeres Objekt angeben, da die Richtlinie direkt an die CloudFormationübergeben würde.

```
MyFunction:
  Type: 'AWS::Serverless::Function'
  Properties:
    CodeUri: ${codeuri}
    Handler: hello.handler
    Runtime: python2.7
    Policies:
      - AmazonSQSFullAccess // IAM policy could be set without passing an empty object
      - DynamoDBCrudPolicy: // SAM specific policy, has a defined structure
          TableName:
            !Ref SampleTable
```

Tabelle mit Richtlinienvorlagen

Die folgende Tabelle enthält die verfügbaren Richtlinienvorlagen.

Richtlinienvorlage	Description
AcmGetCertificatePolicy	Erteilt die Berechtigung zum Lesen eines Zertifikats von AWS Certificate Manager.
AMIDescribePolicy	Erteilt die Erlaubnis, Amazon Machine Images (AMIs) zu beschreiben.
AthenaQueryPolicy	Erteilt Berechtigungen zur Ausführung von Athena-Abfragen.
AWSSecretsManagerGetSecretValuePolicy	Erteilt die Erlaubnis, den geheimen Wert für das angegebene AWS Secrets Manager Geheimnis abzurufen.
AWSSecretsManagerRotationPolicy	Erteilt die Erlaubnis, ein Geheimnis abwechselnd einzugeben AWS Secrets Manager.
CloudFormationDescribeStacksPolicy	Erteilt die Erlaubnis, CloudFormation Stapel zu beschreiben.
CloudWatchDashboardPolicy	Erteilt die Erlaubnis, Metriken auf CloudWatch Dashboards zu platzieren.
CloudWatchDescribeAlarmHistoryPolicy	Erlaubt die Beschreibung des CloudWatch Alarmverlaufs.

Richtlini envorlage	Description		
CloudWatchPutMetricPolicy	Erteilt die Erlaubnis, Messwerte an zu senden CloudWatch.		
CodeCommitCrudPolicy	Erteilt Berechtigungen für create/read/update/delete Objekte in einem bestimmten CodeCommit Repository.		
CodeCommitReadPolicy	Erteilt Berechtigungen zum Lesen von Objekten in einem bestimmten CodeCommit Repository.		
CodePipelineLambdaExecutionPolicy	Erteilt die Erlaubnis für eine Lambda-Funktion, die von aufgerufen wird CodePipeline , um den Status des Jobs zu melden.		
CodePipelineReadOnlyPolicy	Erteilt die Leseberechtigung zum Abrufen von Details zu einer CodePipeline Pipeline.		
ComprehendBasicAccessPolicy	Erteilt die Berechtigung zum Erkennen von Entitäten, Schlüsselbegriffen, Sprachen und Stimmungen.		
CostExplorerReadOnlyPolicy	Erteilt dem Cost Explorer APIs mit Lesezugriff für die Abrechnungshistorie die Berechtigung zum Lesen.		
DynamoDBBackupFullAccessPolicy	Erteilt Lese- und Schreibberechtigungen für DynamoDB-Backups auf Abruf für eine Tabelle.		
DynamoDBCrudPolicy	Erteilt Erstellungs-, Lese-, Aktualisierungs- und Löschberechtigungen für eine Amazon DynamoDB-Tabelle.		

Richtlini envorlage	Description		
DynamoDBR eadPolicy	Erteilt einer DynamoDB-Tabelle nur Leseberechtigungen.		
DynamoDBR econfigur ePolicy	Erteilt die Erlaubnis, eine DynamoDB-Tabelle neu zu konfigurieren.		
DynamoDBR estoreFro mBackupPolicy	Erteilt die Erlaubnis, eine DynamoDB-Tabelle aus einem Backup wiederher zustellen.		
DynamoDBS treamRead Policy	Erlaubt das Beschreiben und Lesen von DynamoDB-Streams und -Datensätzen.		
DynamoDBW ritePolicy	Erteilt nur Schreibberechtigungen für eine DynamoDB-Tabelle.		
EC2CopyIm agePolicy	Erlaubt das Kopieren von Amazon EC2 EC2-Images.		
EC2Descri bePolicy	Erteilt die Erlaubnis, Amazon Elastic Compute Cloud (Amazon EC2) -Instances zu beschreiben.		
EcsRunTas kPolicy	Erteilt die Erlaubnis, eine neue Aufgabe für eine Aufgabendefinition zu starten.		
EFSWriteA ccessPolicy	Erteilt die Erlaubnis, ein Amazon EFS-Dateisystem mit Schreibzugriff zu mounten.		
EKSDescri bePolicy	Erteilt die Erlaubnis, Amazon EKS-Cluster zu beschreiben oder aufzulisten.		

Richtlinienvorlage	Description		
ElasticMapReduceAddJobFlowStepsPolicy	Erteilt die Erlaubnis, einem laufenden Cluster neue Schritte hinzuzufügen.		
ElasticMapReduceCancelStepsPolicy	Erteilt die Erlaubnis, einen oder mehrere ausstehende Schritte in einem laufenden Cluster abzubrechen.		
ElasticMapReduceModifyInstanceFleetPolicy	Erteilt die Erlaubnis, Details für Instance-Flotten innerhalb eines Clusters aufzulisten und Kapazitäten zu ändern.		
ElasticMapReduceModifyInstanceGroupsPolicy	Erteilt die Erlaubnis, Details aufzulisten und Einstellungen für Instanzgruppen innerhalb eines Clusters zu ändern.		
ElasticMapReduceSetTerminationProtectionPolicy	Erteilt die Erlaubnis, den Kündigungsschutz für einen Cluster festzulegen.		
ElasticMapReduceTerminateJobFlowsPolicy	Erteilt die Erlaubnis, einen Cluster herunterzufahren.		
ElasticsearchHttpPostPolicy	Erteilt Amazon OpenSearch Service die POST-Berechtigung.		

Richtlinienvorlage	Description		
EventBridgePutEventsPolicy	Erteilt Berechtigungen zum Senden von Ereignissen an EventBridge.		
FilterLogEventsPolicy	Erteilt die Berechtigung, CloudWatch Log-Ereignisse aus einer bestimmten Protokollgruppe zu filtern.		
FirehoseCrudPolicy	Erteilt die Erlaubnis, einen Firehose-Lieferstream zu erstellen, zu schreiben, zu aktualisieren und zu löschen.		
FirehoseWritePolicy	Erteilt die Erlaubnis, in einen Firehose-Lieferstream zu schreiben.		
KinesisCrudPolicy	Erteilt die Erlaubnis, einen Amazon Kinesis Kinesis-Stream zu erstellen, zu veröffentlichen und zu löschen.		
KinesisStreamReadPolicy	Erteilt die Erlaubnis, einen Amazon Kinesis Kinesis-Stream aufzulisten und zu lesen.		
KMSDecryptPolicy	Erteilt die Erlaubnis zum Entschlüsseln mit einem AWS Key Management Service (AWS KMS) -Schlüssel.		
KMSEncryptPolicy	Erteilt die Erlaubnis, mit einem Schlüssel AWS Key Management Service (AWS KMS) zu verschlüsseln.		
LambdaInvokePolicy	Erteilt die Erlaubnis, eine AWS Lambda Funktion, einen Alias oder eine Version aufzurufen.		

Richtlinienvorlage	Description		
MobileAnalyticsWriteOnlyAccessPolicy	Erteilt Schreibzugriff zum Speichern von Ereignisdaten für alle Anwendungsressourcen.		
OrganizationsListAccountsPolicy	Erteilt die Leseberechtigung zum Auflisten der Kontonamen von Kindern und. IDs		
PinpointEndpointAccessPolicy	Erteilt die Erlaubnis, Endpunkte für eine Amazon Pinpoint Pinpoint-Anwendung abzurufen und zu aktualisieren.		
PollyFullAccessPolicy	Gewährt vollen Zugriff auf Amazon Polly Polly-Lexikonressourcen.		
RekognitionDetectOnlyPolicy	Erteilt die Erlaubnis, Gesichter, Beschriftungen und Text zu erkennen.		
RekognitionFacesManagementPolicy	Erlaubt das Hinzufügen, Löschen und Suchen von Gesichtern in einer Amazon Rekognition Rekognition-Sammlung.		
RekognitionFacesPolicy	Erlaubt das Vergleichen und Erkennen von Gesichtern und Labels.		
RekognitionLabelsPolicy	Erteilt die Erlaubnis, Objekt- und Moderationsbezeichnungen zu erkennen.		
RekognitionNoDataAccessPolicy	Erlaubt das Vergleichen und Erkennen von Gesichtern und Labels.		

Richtlini envorlage	Description		
RekognitionReadPolicy	Erlaubt das Auflisten und Suchen von Gesichtern.		
RekognitionWriteOnlyAccessPolicy	Erteilt die Erlaubnis, Gesichter in einer Sammlung zu erstellen und zu indexieren.		
Route53ChangeResourceRecordSetsPolicy	Erteilt die Erlaubnis, Ressourcendatensätze in Route 53 zu ändern.		
S3CrudPolicy	Erteilt Erstellungs-, Lese-, Aktualisierungs- und Löschberechtigungen, um auf die Objekte in einem Amazon S3 S3-Bucket zu reagieren.		
S3FullAccessPolicy	Erteilt die volle Zugriffsberechtigung, um auf die Objekte in einem Amazon S3 S3-Bucket zu reagieren.		
S3ReadPolicy	Erteilt nur Leseberechtigungen zum Lesen von Objekten in einem Amazon Simple Storage Service (Amazon S3) -Bucket.		
S3WritePolicy	Erteilt Schreibberechtigungen zum Schreiben von Objekten in einen Amazon S3 S3-Bucket.		
SageMakerCreateEndpointConfigPolicy	Erteilt die Erlaubnis, eine Endpunktkonfiguration in SageMaker AI zu erstellen.		

Richtlini envorlage	Description		
SageMaker CreateEnd pointPolicy	Erteilt die Erlaubnis, einen Endpunkt in SageMaker AI zu erstellen.		
Serverles sRepoRead WriteAcce ssPolicy	Erteilt die Erlaubnis, Anwendungen im AWS Serverless Application Repository Dienst zu erstellen und aufzulisten.		
SESBulkTe mplatedCr udPolicy	Erteilt die Erlaubnis, E-Mails, E-Mails mit Vorlagen und Massen-E-Mails mit Vorlagen zu senden und die Identität zu überprüfen.		
SESBulkTe mplatedCr udPolicy_v2	Erteilt die Erlaubnis, Amazon SES SES-E-Mails, E-Mail-Vorlagen und Massen-E-Mails mit Vorlagen zu senden und die Identität zu überprüfen.		
SESCrudPolicy	Erteilt die Erlaubnis, E-Mails zu senden und die Identität zu überprüfen.		
SESEmailT emplateCr udPolicy	Erteilt die Erlaubnis, Amazon SES SES-E-Mail-Vorlagen zu erstellen, abzurufen , aufzulisten, zu aktualisieren und zu löschen.		
SESSendBo uncePolicy	SendBounce Erteilt die Erlaubnis für eine Amazon Simple Email Service (Amazon SES) -Identität.		
SNSCrudPo licy	Erteilt die Erlaubnis, Amazon SNS SNS-Themen zu erstellen, zu veröffentlichen und zu abonnieren.		

Richtlini envorlage	Description		
SNSPublishMessageP olicy	Erteilt die Erlaubnis, eine Nachricht zu einem Amazon Simple Notification Service (Amazon SNS) -Thema zu veröffentlichen.		
SQSPoller Policy	Erteilt die Erlaubnis, eine Amazon Simple Queue Service (Amazon SQS) -Wartesch lange abzufragen.		
SQSSendMe ssagePolicy	Erteilt die Erlaubnis, Nachrichten an eine Amazon SQS SQS-Warteschlange zu senden.		
SSMParame terReadPolicy	Erteilt die Erlaubnis, auf einen Parameter aus einem Amazon EC2 Systems Manager (SSM) -Parameterspeicher zuzugreifen, um Geheimnisse in dieses Konto zu laden. Wird verwendet, wenn der Parametername kein Schrägstrich-Präfix hat.		
SSMParame terWithSl ashPrefix ReadPolicy	Erteilt die Erlaubnis, auf einen Parameter aus einem Amazon EC2 Systems Manager (SSM) -Parameterspeicher zuzugreifen, um Geheimnisse in dieses Konto zu laden. Wird verwendet, wenn der Parametername ein Schrägstrich-Präfix hat.		
StepFunct ionsExecu tionPolicy	Erteilt die Erlaubnis, die Ausführung einer Step Functions Functions-Zustands maschine zu starten.		
TextractD etectAnal yzePolicy	Ermöglicht den Zugriff auf die Erkennung und Analyse von Dokumenten mit Amazon Textract.		

Richtlini envorlage	Description		
TextractGetResultPolicy	Ermöglicht den Zugriff auf erkannte und analysierte Dokumente von Amazon Textract.		
TextractPolicy	Ermöglicht vollen Zugriff auf Amazon Textract.		
VPCAccessPolicy	Ermöglicht das Erstellen, Löschen, Beschreiben und Trennen elastischer Netzwerkschnittstellen.		

Fehlerbehebung

SAM-CLI-Fehler: „Gültige Parameterwerte für die Richtlinienvorlage '< policy-template-name >' müssen angegeben werden“

Bei der Ausführung von `sam build` wird der folgende Fehler angezeigt:

```
"Must specify valid parameter values for policy template '<policy-template-name>'"
```

Das bedeutet, dass Sie bei der Deklaration einer Richtlinienvorlage, die keine Platzhalterwerte enthält, kein leeres Objekt übergeben haben.

Um dieses Problem zu beheben, deklarieren Sie die Richtlinie wie im folgenden Beispiel für. CloudWatchPutMetricPolicy

```
MyFunction:
  Policies:
    - CloudWatchPutMetricPolicy: {}
```

AWS SAM Liste der Richtlinienvorlagen

Im Folgenden sind die verfügbaren Richtlinienvorlagen zusammen mit den Berechtigungen aufgeführt, die auf die einzelnen Vorlagen angewendet werden. AWS Serverless Application Model

(AWS SAM) füllt die Platzhalterelemente (wie AWS Region und Konto-ID) automatisch mit den entsprechenden Informationen aus.

Themen

- AcmGetCertificatePolicy
- AMIDescribePolicy
- AthenaQueryPolicy
- AWSSecretsManagerGetSecretValuePolicy
- AWSSecretsManagerRotationPolicy
- CloudFormationDescribeStacksPolicy
- CloudWatchDashboardPolicy
- CloudWatchDescribeAlarmHistoryPolicy
- CloudWatchPutMetricPolicy
- CodePipelineLambdaExecutionPolicy
- CodePipelineReadOnlyPolicy
- CodeCommitCrudPolicy
- CodeCommitReadPolicy
- ComprehendBasicAccessPolicy
- CostExplorerReadOnlyPolicy
- DynamoDBBackupFullAccessPolicy
- DynamoDBCrudPolicy
- DynamoDBReadPolicy
- DynamoDBReconfigurePolicy
- DynamoDBRestoreFromBackupPolicy
- DynamoDBStreamReadPolicy
- DynamoDBWritePolicy
- EC2CopyImagePolicy
- EC2DescribePolicy
- EcsRunTaskPolicy
- EFSWriteAccessPolicy
- EKSDescribePolicy

- ElasticMapReduceAddJobFlowStepsPolicy
- ElasticMapReduceCancelStepsPolicy
- ElasticMapReduceModifyInstanceFleetPolicy
- ElasticMapReduceModifyInstanceGroupsPolicy
- ElasticMapReduceSetTerminationProtectionPolicy
- ElasticMapReduceTerminateJobFlowsPolicy
- ElasticsearchHttpPostPolicy
- EventBridgePutEventsPolicy
- FilterLogEventsPolicy
- FirehoseCrudPolicy
- FirehoseWritePolicy
- KinesisCrudPolicy
- KinesisStreamReadPolicy
- KMSDecryptPolicy
- KMSEncryptPolicy
- LambdaInvokePolicy
- MobileAnalyticsWriteOnlyAccessPolicy
- OrganizationsListAccountsPolicy
- PinpointEndpointAccessPolicy
- PollyFullAccessPolicy
- RekognitionDetectOnlyPolicy
- RekognitionFacesManagementPolicy
- RekognitionFacesPolicy
- RekognitionLabelsPolicy
- RekognitionNoDataAccessPolicy
- RekognitionReadPolicy
- RekognitionWriteOnlyAccessPolicy
- Route53ChangeResourceRecordSetsPolicy
- S3CrudPolicy
- S3FullAccessPolicy

- S3ReadPolicy
- S3WritePolicy
- SageMakerCreateEndpointConfigPolicy
- SageMakerCreateEndpointPolicy
- ServerlessRepoReadWriteAccessPolicy
- SESBulkTemplatedCrudPolicy
- SESBulkTemplatedCrudPolicy_v2
- SESCrudPolicy
- SESEmailTemplateCrudPolicy
- SESSendBouncePolicy
- SNSCrudPolicy
- SNSPublishMessagePolicy
- SQSPollerPolicy
- SQSSendMessagePolicy
- SSMParameterReadPolicy
- SSMParameterWithSlashPrefixReadPolicy
- StepFunctionsExecutionPolicy
- TextractDetectAnalyzePolicy
- TextractGetResultPolicy
- TextractPolicy
- VPCAccessPolicy

AcmGetCertificatePolicy

Erteilt die Berechtigung zum Lesen eines Zertifikats von. AWS Certificate Manager

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "acm:GetCertificate"
    ],
    "Resource": {
      "Fn::Sub": [
```

```
        "${certificateArn}",
        {
          "certificateArn": {
            "Ref": "CertificateArn"
          }
        }
      ]
    }
  }
]
```

AMIDescribePolicy

Erteilt die Erlaubnis, Amazon Machine Images (AMIs) zu beschreiben.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ec2:DescribeImages"
    ],
    "Resource": "*"
  }
]
```

AthenaQueryPolicy

Erteilt Berechtigungen zur Ausführung von Athena-Abfragen.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "athena:ListWorkGroups",
      "athena:GetExecutionEngine",
      "athena:GetExecutionEngines",
      "athena:GetNamespace",
      "athena:GetCatalogs",
      "athena:GetNamespaces",
      "athena:GetTables",
      "athena:GetTable"
    ],
    "Resource": "*"
```

```
  },
  {
    "Effect": "Allow",
    "Action": [
      "athena:StartQueryExecution",
      "athena:GetQueryResults",
      "athena:DeleteNamedQuery",
      "athena:GetNamedQuery",
      "athena:ListQueryExecutions",
      "athena:StopQueryExecution",
      "athena:GetQueryResultsStream",
      "athena:ListNamedQueries",
      "athena:CreateNamedQuery",
      "athena:GetQueryExecution",
      "athena:BatchGetNamedQuery",
      "athena:BatchGetQueryExecution",
      "athena:GetWorkGroup"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:athena:${AWS::Region}:${AWS::AccountId}:workgroup/
${workgroupName}",
        {
          "workgroupName": {
            "Ref": "WorkGroupName"
          }
        }
      ]
    }
  }
]
```

AWSSecretsManagerGetSecretValuePolicy

Erteilt die Erlaubnis, den geheimen Wert für das angegebene AWS Secrets Manager Geheimnis abzurufen.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "secretsmanager:GetSecretValue"
    ],
```

```
    "Resource": {
      "Fn::Sub": [
        "${secretArn}",
        {
          "secretArn": {
            "Ref": "SecretArn"
          }
        }
      ]
    }
  }
]
```

AWSSecretsManagerRotationPolicy

Erteilt die Erlaubnis, ein Geheimnis zu wechseln AWS Secrets Manager.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "secretsmanager:DescribeSecret",
      "secretsmanager:GetSecretValue",
      "secretsmanager:PutSecretValue",
      "secretsmanager:UpdateSecretVersionStage"
    ],
    "Resource": {
      "Fn::Sub": "arn:${AWS::Partition}:secretsmanager:${AWS::Region}:
${AWS::AccountId}:secret:*"
    },
    "Condition": {
      "StringEquals": {
        "secretsmanager:resource/AllowRotationLambdaArn": {
          "Fn::Sub": [
            "arn:${AWS::Partition}:lambda:${AWS::Region}:${AWS::AccountId}:function:
${functionName}",
            {
              "functionName": {
                "Ref": "FunctionName"
              }
            }
          ]
        }
      }
```

```
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "secretsmanager:GetRandomPassword"
    ],
    "Resource": "*"
  }
]
```

CloudFormationDescribeStacksPolicy

Erteilt die Erlaubnis, CloudFormation Stapel zu beschreiben.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "cloudformation:DescribeStacks"
    ],
    "Resource": {
      "Fn::Sub": "arn:${AWS::Partition}:cloudformation:${AWS::Region}:
${AWS::AccountId}:stack/*"
    }
  }
]
```

CloudWatchDashboardPolicy

Erteilt die Erlaubnis, Metriken auf CloudWatch Dashboards zu platzieren.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "cloudwatch:GetDashboard",
      "cloudwatch:ListDashboards",
      "cloudwatch:PutDashboard",
      "cloudwatch:ListMetrics"
    ],
    "Resource": "*"
  }
```

```
]
```

CloudWatchDescribeAlarmHistoryPolicy

Erlaubt die Beschreibung des CloudWatch Amazon-Alarmverlaufs.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "cloudwatch:DescribeAlarmHistory"
    ],
    "Resource": "*"
  }
]
```

CloudWatchPutMetricPolicy

Erteilt die Erlaubnis, Metriken an zu senden CloudWatch.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "cloudwatch:PutMetricData"
    ],
    "Resource": "*"
  }
]
```

CodePipelineLambdaExecutionPolicy

Erteilt die Erlaubnis für eine Lambda-Funktion, die von aufgerufen wird AWS CodePipeline , um den Status des Jobs zu melden.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "codepipeline:PutJobSuccessResult",
      "codepipeline:PutJobFailureResult"
    ],
    "Resource": "*"
```

```
  }
]
```

CodePipelineReadOnlyPolicy

Erteilt die Leseberechtigung zum Abrufen von Details zu einer CodePipeline Pipeline.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "codepipeline:ListPipelineExecutions"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:codepipeline:${AWS::Region}:${AWS::AccountId}:
${pipelinename}",
        {
          "pipelinename": {
            "Ref": "PipelineName"
          }
        }
      ]
    }
  }
]
```

CodeCommitCrudPolicy

Erteilt Berechtigungen zum Erstellen, Lesen, Aktualisieren und Löschen von Objekten innerhalb eines bestimmten CodeCommit Repositorys.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "codecommit:GitPull",
      "codecommit:GitPush",
      "codecommit:CreateBranch",
      "codecommit:DeleteBranch",
      "codecommit:GetBranch",
      "codecommit:ListBranches",
      "codecommit:MergeBranchesByFastForward",
```

```
        "codecommit:MergeBranchesBySquash",
        "codecommit:MergeBranchesByThreeWay",
        "codecommit:UpdateDefaultBranch",
        "codecommit:BatchDescribeMergeConflicts",
        "codecommit:CreateUnreferencedMergeCommit",
        "codecommit:DescribeMergeConflicts",
        "codecommit:GetMergeCommit",
        "codecommit:GetMergeOptions",
        "codecommit:BatchGetPullRequests",
        "codecommit:CreatePullRequest",
        "codecommit:DescribePullRequestEvents",
        "codecommit:GetCommentsForPullRequest",
        "codecommit:GetCommitsFromMergeBase",
        "codecommit:GetMergeConflicts",
        "codecommit:GetPullRequest",
        "codecommit:ListPullRequests",
        "codecommit:MergePullRequestByFastForward",
        "codecommit:MergePullRequestBySquash",
        "codecommit:MergePullRequestByThreeWay",
        "codecommit:PostCommentForPullRequest",
        "codecommit:UpdatePullRequestDescription",
        "codecommit:UpdatePullRequestStatus",
        "codecommit:UpdatePullRequestTitle",
        "codecommit:DeleteFile",
        "codecommit:GetBlob",
        "codecommit:GetFile",
        "codecommit:GetFolder",
        "codecommit:PutFile",
        "codecommit:DeleteCommentContent",
        "codecommit:GetComment",
        "codecommit:GetCommentsForComparedCommit",
        "codecommit:PostCommentForComparedCommit",
        "codecommit:PostCommentReply",
        "codecommit:UpdateComment",
        "codecommit:BatchGetCommits",
        "codecommit:CreateCommit",
        "codecommit:GetCommit",
        "codecommit:GetCommitHistory",
        "codecommit:GetDifferences",
        "codecommit:GetObjectIdentifier",
        "codecommit:GetReferences",
        "codecommit:GetTree",
        "codecommit:GetRepository",
        "codecommit:UpdateRepositoryDescription",
```

```
      "codecommit:ListTagsForResource",
      "codecommit:TagResource",
      "codecommit:UntagResource",
      "codecommit:GetRepositoryTriggers",
      "codecommit:PutRepositoryTriggers",
      "codecommit:TestRepositoryTriggers",
      "codecommit:GetBranch",
      "codecommit:GetCommit",
      "codecommit:UploadArchive",
      "codecommit:GetUploadArchiveStatus",
      "codecommit:CancelUploadArchive"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:codecommit:${AWS::Region}:${AWS::AccountId}:
${repositoryName}",
        {
          "repositoryName": {
            "Ref": "RepositoryName"
          }
        }
      ]
    }
  }
]
```

CodeCommitReadPolicy

Erteilt Berechtigungen zum Lesen von Objekten innerhalb eines bestimmten CodeCommit Repositorys.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "codecommit:GitPull",
      "codecommit:GetBranch",
      "codecommit:ListBranches",
      "codecommit:BatchDescribeMergeConflicts",
      "codecommit:DescribeMergeConflicts",
      "codecommit:GetMergeCommit",
      "codecommit:GetMergeOptions",
      "codecommit:BatchGetPullRequests",
```

```
      "codecommit:DescribePullRequestEvents",
      "codecommit:GetCommentsForPullRequest",
      "codecommit:GetCommitsFromMergeBase",
      "codecommit:GetMergeConflicts",
      "codecommit:GetPullRequest",
      "codecommit:ListPullRequests",
      "codecommit:GetBlob",
      "codecommit:GetFile",
      "codecommit:GetFolder",
      "codecommit:GetComment",
      "codecommit:GetCommentsForComparedCommit",
      "codecommit:BatchGetCommits",
      "codecommit:GetCommit",
      "codecommit:GetCommitHistory",
      "codecommit:GetDifferences",
      "codecommit:GetObjectIdentifier",
      "codecommit:GetReferences",
      "codecommit:GetTree",
      "codecommit:GetRepository",
      "codecommit:ListTagsForResource",
      "codecommit:GetRepositoryTriggers",
      "codecommit:TestRepositoryTriggers",
      "codecommit:GetBranch",
      "codecommit:GetCommit",
      "codecommit:GetUploadArchiveStatus"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:codecommit:${AWS::Region}:${AWS::AccountId}:
${repositoryName}",
        {
          "repositoryName": {
            "Ref": "RepositoryName"
          }
        }
      ]
    }
  }
]
```

ComprehendBasicAccessPolicy

Erteilt die Erlaubnis zum Erkennen von Entitäten, Schlüsselbegriffen, Sprachen und Stimmungen.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "comprehend:BatchDetectKeyPhrases",
      "comprehend:DetectDominantLanguage",
      "comprehend:DetectEntities",
      "comprehend:BatchDetectEntities",
      "comprehend:DetectKeyPhrases",
      "comprehend:DetectSentiment",
      "comprehend:BatchDetectDominantLanguage",
      "comprehend:BatchDetectSentiment"
    ],
    "Resource": "*"
  }
]
```

CostExplorerReadOnlyPolicy

Erlaubt dem Nur-Lese-Zugriff AWS Cost Explorer (Cost Explorer) APIs für die Abrechnungshistorie.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ce:GetCostAndUsage",
      "ce:GetDimensionValues",
      "ce:GetReservationCoverage",
      "ce:GetReservationPurchaseRecommendation",
      "ce:GetReservationUtilization",
      "ce:GetTags"
    ],
    "Resource": "*"
  }
]
```

DynamoDBBackupFullAccessPolicy

Erteilt Lese- und Schreibberechtigungen für DynamoDB-Backups auf Abruf für eine Tabelle.

```
"Statement": [
  {
    "Effect": "Allow",
```

```
    "Action": [
      "dynamodb:CreateBackup",
      "dynamodb:DescribeContinuousBackups"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
${tableName}",
        {
          "tableName": {
            "Ref": "TableName"
          }
        }
      ]
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "dynamodb:DeleteBackup",
      "dynamodb:DescribeBackup",
      "dynamodb:ListBackups"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
${tableName}/backup/*",
        {
          "tableName": {
            "Ref": "TableName"
          }
        }
      ]
    }
  }
]
```

DynamoDBCrudPolicy

Erteilt Erstellungs-, Lese-, Aktualisierungs- und Löschberechtigungen für eine Amazon DynamoDB-Tabelle.

```
"Statement": [
```

```
  {
    "Effect": "Allow",
    "Action": [
      "dynamodb:GetItem",
      "dynamodb:DeleteItem",
      "dynamodb:PutItem",
      "dynamodb:Scan",
      "dynamodb:Query",
      "dynamodb:UpdateItem",
      "dynamodb:BatchWriteItem",
      "dynamodb:BatchGetItem",
      "dynamodb:DescribeTable",
      "dynamodb:ConditionCheckItem"
    ],
    "Resource": [
      {
        "Fn::Sub": [
          "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
${tableName}",
          {
            "tableName": {
              "Ref": "TableName"
            }
          }
        ]
      },
      {
        "Fn::Sub": [
          "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
${tableName}/index/*",
          {
            "tableName": {
              "Ref": "TableName"
            }
          }
        ]
      }
    ]
  }
]
```

DynamoDBReadPolicy

Erteilt einer DynamoDB-Tabelle nur Leseberechtigungen.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "dynamodb:GetItem",
      "dynamodb:Scan",
      "dynamodb:Query",
      "dynamodb:BatchGetItem",
      "dynamodb:DescribeTable"
    ],
    "Resource": [
      {
        "Fn::Sub": [
          "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
${tableName}",
          {
            "tableName": {
              "Ref": "TableName"
            }
          }
        ]
      },
      {
        "Fn::Sub": [
          "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
${tableName}/index/*",
          {
            "tableName": {
              "Ref": "TableName"
            }
          }
        ]
      }
    ]
  }
]
```

DynamoDBReconfigurePolicy

Erteilt die Erlaubnis, eine DynamoDB-Tabelle neu zu konfigurieren.

```
"Statement": [
  {
```

```
    "Effect": "Allow",
    "Action": [
      "dynamodb:UpdateTable"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
${tableName}",
        {
          "tableName": {
            "Ref": "TableName"
          }
        }
      ]
    }
  }
]
```

DynamoDBRestoreFromBackupPolicy

Erteilt die Erlaubnis, eine DynamoDB-Tabelle aus einem Backup wiederherzustellen.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "dynamodb:RestoreTableFromBackup"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
${tableName}/backup/*",
        {
          "tableName": {
            "Ref": "TableName"
          }
        }
      ]
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "dynamodb:PutItem",
```

```
      "dynamodb:UpdateItem",
      "dynamodb:DeleteItem",
      "dynamodb:GetItem",
      "dynamodb:Query",
      "dynamodb:Scan",
      "dynamodb:BatchWriteItem"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
${tableName}",
        {
          "tableName": {
            "Ref": "TableName"
          }
        }
      ]
    }
  }
]
```

DynamoDBStreamReadPolicy

Erlaubt das Beschreiben und Lesen von DynamoDB-Streams und -Datensätzen.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "dynamodb:DescribeStream",
      "dynamodb:GetRecords",
      "dynamodb:GetShardIterator"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
${tableName}/stream/${streamName}",
        {
          "tableName": {
            "Ref": "TableName"
          },
          "streamName": {
            "Ref": "StreamName"
          }
```

```
        }
      ]
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "dynamodb:ListStreams"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
${tableName}/stream/*",
        {
          "tableName": {
            "Ref": "TableName"
          }
        }
      ]
    }
  }
]
```

DynamoDBWritePolicy

Erteilt nur Schreibberechtigungen für eine DynamoDB-Tabelle.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "dynamodb:PutItem",
      "dynamodb:UpdateItem",
      "dynamodb:BatchWriteItem"
    ],
    "Resource": [
      {
        "Fn::Sub": [
          "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
${tableName}",
          {
            "tableName": {
              "Ref": "TableName"
            }
```

```
          }
        ]
      },
      {
        "Fn::Sub": [
          "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
${tableName}/index/*",
          {
            "tableName": {
              "Ref": "TableName"
            }
          }
        ]
      }
    ]
  }
]
```

EC2CopyImagePolicy

Erteilt die Erlaubnis, EC2 Amazon-Bilder zu kopieren.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ec2:CopyImage"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:ec2:${AWS::Region}:${AWS::AccountId}:image/${imageId}",
        {
          "imageId": {
            "Ref": "ImageId"
          }
        }
      ]
    }
  }
]
```

EC2DescribePolicy

Erteilt die Erlaubnis, Amazon Elastic Compute Cloud (Amazon EC2) -Instances zu beschreiben.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ec2:DescribeRegions",
      "ec2:DescribeInstances"
    ],
    "Resource": "*"
  }
]
```

EcsRunTaskPolicy

Erteilt die Erlaubnis, eine neue Aufgabe für eine Aufgabendefinition zu starten.

```
"Statement": [
  {
    "Action": [
      "ecs:RunTask"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:ecs:${AWS::Region}:${AWS::AccountId}:task-definition/
${taskDefinition}",
        {
          "taskDefinition": {
            "Ref": "TaskDefinition"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

EFSWriteAccessPolicy

Erteilt die Erlaubnis, ein Amazon EFS-Dateisystem mit Schreibzugriff zu mounten.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "elasticfilesystem:ClientMount",
      "elasticfilesystem:ClientWrite"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:elasticfilesystem:${AWS::Region}:${AWS::AccountId}:file-
system/${FileSystem}",
        {
          "FileSystem": {
            "Ref": "FileSystem"
          }
        }
      ]
    },
    "Condition": {
      "StringEquals": {
        "elasticfilesystem:AccessPointArn": {
          "Fn::Sub": [
            "arn:${AWS::Partition}:elasticfilesystem:${AWS::Region}:
${AWS::AccountId}:access-point/${AccessPoint}",
            {
              "AccessPoint": {
                "Ref": "AccessPoint"
              }
            }
          ]
        }
      }
    }
  }
]
```

EKSDescribePolicy

Erteilt die Erlaubnis, Amazon Elastic Kubernetes Service (Amazon EKS) -Cluster zu beschreiben oder aufzulisten.

```
"Statement": [
  {
```

```
    "Effect": "Allow",
    "Action": [
      "eks:DescribeCluster",
      "eks:ListClusters"
    ],
    "Resource": "*"
  }
]
```

ElasticMapReduceAddJobFlowStepsPolicy

Erteilt die Erlaubnis, einem laufenden Cluster neue Schritte hinzuzufügen.

```
"Statement": [
  {
    "Action": "elasticmapreduce:AddJobFlowSteps",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:
${AWS::AccountId}:cluster/${clusterId}",
        {
          "clusterId": {
            "Ref": "ClusterId"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

ElasticMapReduceCancelStepsPolicy

Erteilt die Erlaubnis, einen oder mehrere ausstehende Schritte in einem laufenden Cluster abzubrechen.

```
"Statement": [
  {
    "Action": "elasticmapreduce:CancelSteps",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:
${AWS::AccountId}:cluster/${clusterId}",
```

```
        {
          "clusterId": {
            "Ref": "ClusterId"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

ElasticMapReduceModifyInstanceFleetPolicy

Erteilt die Erlaubnis, Details für Instance-Flotten innerhalb eines Clusters aufzulisten und Kapazitäten zu ändern.

```
"Statement": [
  {
    "Action": [
      "elasticmapreduce:ModifyInstanceFleet",
      "elasticmapreduce:ListInstanceFleets"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:
${AWS::AccountId}:cluster/${clusterId}",
        {
          "clusterId": {
            "Ref": "ClusterId"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

ElasticMapReduceModifyInstanceGroupsPolicy

Erteilt die Erlaubnis, Details aufzulisten und Einstellungen für Instanzgruppen innerhalb eines Clusters zu ändern.

```
"Statement": [
```

```
  {
    "Action": [
      "elasticmapreduce:ModifyInstanceGroups",
      "elasticmapreduce:ListInstanceGroups"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:
${AWS::AccountId}:cluster/${clusterId}",
        {
          "clusterId": {
            "Ref": "ClusterId"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

ElasticMapReduceSetTerminationProtectionPolicy

Erteilt die Erlaubnis, den Kündigungsschutz für einen Cluster festzulegen.

```
"Statement": [
  {
    "Action": "elasticmapreduce:SetTerminationProtection",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:
${AWS::AccountId}:cluster/${clusterId}",
        {
          "clusterId": {
            "Ref": "ClusterId"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

ElasticMapReduceTerminateJobFlowsPolicy

Erteilt die Erlaubnis, einen Cluster herunterzufahren.

```
"Statement": [
  {
    "Action": "elasticmapreduce:TerminateJobFlows",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:
${AWS::AccountId}:cluster/${clusterId}",
        {
          "clusterId": {
            "Ref": "ClusterId"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

ElasticsearchHttpPostPolicy

Erteilt Amazon OpenSearch Service die POST- und PUT-Berechtigungen.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "es:ESHttpPost",
      "es:ESHttpPut"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:es:${AWS::Region}:${AWS::AccountId}:domain/
${domainName}/*",
        {
          "domainName": {
            "Ref": "DomainName"
          }
        }
      ]
```

```
    }
  }
]
```

EventBridgePutEventsPolicy

Erteilt die Erlaubnis, Ereignisse an Amazon zu senden EventBridge.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": "events:PutEvents",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:events:${AWS::Region}:${AWS::AccountId}:event-bus/
${eventBusName}",
        {
          "eventBusName": {
            "Ref": "EventBusName"
          }
        }
      ]
    }
  }
]
```

FilterLogEventsPolicy

Erteilt die Erlaubnis, CloudWatch Log-Ereignisse aus einer bestimmten Protokollgruppe zu filtern.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "logs:FilterLogEvents"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:logs:${AWS::Region}:${AWS::AccountId}:log-group:
${logGroupName}:log-stream:*",
        {
          "logGroupName": {
            "Ref": "LogGroupName"
```

```
          }
        }
      ]
    }
  }
]
```

FirehoseCrudPolicy

Erteilt die Erlaubnis, einen Firehose-Lieferstream zu erstellen, zu schreiben, zu aktualisieren und zu löschen.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "firehose:CreateDeliveryStream",
      "firehose:DeleteDeliveryStream",
      "firehose:DescribeDeliveryStream",
      "firehose:PutRecord",
      "firehose:PutRecordBatch",
      "firehose:UpdateDestination"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:firehose:${AWS::Region}:
${AWS::AccountId}:deliverystream/${deliveryStreamName}",
        {
          "deliveryStreamName": {
            "Ref": "DeliveryStreamName"
          }
        }
      ]
    }
  }
]
```

FirehoseWritePolicy

Erteilt die Erlaubnis, in einen Firehose-Lieferstream zu schreiben.

```
"Statement": [
  {
```

```
    "Effect": "Allow",
    "Action": [
      "firehose:PutRecord",
      "firehose:PutRecordBatch"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:firehose:${AWS::Region}:
${AWS::AccountId}:deliverystream/${deliveryStreamName}",
        {
          "deliveryStreamName": {
            "Ref": "DeliveryStreamName"
          }
        }
      ]
    }
  }
]
```

KinesisCrudPolicy

Erteilt die Erlaubnis, einen Amazon Kinesis Kinesis-Stream zu erstellen, zu veröffentlichen und zu löschen.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "kinesis:AddTagsToStream",
      "kinesis:CreateStream",
      "kinesis:DecreaseStreamRetentionPeriod",
      "kinesis:DeleteStream",
      "kinesis:DescribeStream",
      "kinesis:DescribeStreamSummary",
      "kinesis:GetShardIterator",
      "kinesis:IncreaseStreamRetentionPeriod",
      "kinesis:ListTagsForStream",
      "kinesis:MergeShards",
      "kinesis:PutRecord",
      "kinesis:PutRecords",
      "kinesis:SplitShard",
      "kinesis:RemoveTagsFromStream"
    ],
```

```
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:kinesis:${AWS::Region}:${AWS::AccountId}:stream/
${streamName}",
        {
          "streamName": {
            "Ref": "StreamName"
          }
        }
      ]
    }
  }
]
```

KinesisStreamReadPolicy

Erteilt die Erlaubnis, einen Amazon Kinesis Kinesis-Stream aufzulisten und zu lesen.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "kinesis:ListStreams",
      "kinesis:DescribeLimits"
    ],
    "Resource": {
      "Fn::Sub": "arn:${AWS::Partition}:kinesis:${AWS::Region}:
${AWS::AccountId}:stream/*"
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "kinesis:DescribeStream",
      "kinesis:DescribeStreamSummary",
      "kinesis:GetRecords",
      "kinesis:GetShardIterator"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:kinesis:${AWS::Region}:${AWS::AccountId}:stream/
${streamName}",
        {
          "streamName": {
```

```
            "Ref": "StreamName"
          }
        }
      ]
    }
  }
]
```

KMSDecryptPolicy

Erteilt die Erlaubnis zum Entschlüsseln mit einem AWS Key Management Service (AWS KMS) -Schlüssel. Beachten Sie, dass es sich um eine AWS KMS Schlüssel-ID und nicht um einen Schlüsselalias handeln `keyId` muss.

```
"Statement": [
  {
    "Action": "kms:Decrypt",
    "Effect": "Allow",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:kms:${AWS::Region}:${AWS::AccountId}:key/${keyId}",
        {
          "keyId": {
            "Ref": "KeyId"
          }
        }
      ]
    }
  }
]
```

KMSEncryptPolicy

Erteilt die Erlaubnis, mit einem AWS KMS Schlüssel zu verschlüsseln. Beachten Sie, dass KeyID eine AWS KMS Schlüssel-ID und kein Schlüsselalias sein muss.

```
"Statement": [
  {
    "Action": "kms:Encrypt",
    "Effect": "Allow",
    "Resource": {
      "Fn::Sub": [
```

```
        "arn:${AWS::Partition}:kms:${AWS::Region}:${AWS::AccountId}:key/${keyId}",
        {
          "keyId": {
            "Ref": "KeyId"
          }
        }
      ]
    }
  }
]
```

LambdaInvokePolicy

Erteilt die Erlaubnis, eine AWS Lambda Funktion, einen Alias oder eine Version aufzurufen.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "lambda:InvokeFunction"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:lambda:${AWS::Region}:${AWS::AccountId}:function:
${functionName}*",
        {
          "functionName": {
            "Ref": "FunctionName"
          }
        }
      ]
    }
  }
]
```

MobileAnalyticsWriteOnlyAccessPolicy

Erteilt Schreibzugriff zum Speichern von Ereignisdaten für alle Anwendungsressourcen.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
```

```
      "mobileanalytics:PutEvents"
    ],
    "Resource": "*"
  }
]
```

OrganizationsListAccountsPolicy

Erlaubt die Nur-Lese-Berechtigung zum Auflisten der Kontonamen von Kindern und. IDs

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "organizations:ListAccounts"
    ],
    "Resource": "*"
  }
]
```

PinpointEndpointAccessPolicy

Erteilt die Erlaubnis, Endpunkte für eine Amazon Pinpoint Pinpoint-Anwendung abzurufen und zu aktualisieren.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "mobiletargeting:GetEndpoint",
      "mobiletargeting:UpdateEndpoint",
      "mobiletargeting:UpdateEndpointsBatch"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:mobiletargeting:${AWS::Region}:${AWS::AccountId}:apps/
${pinpointApplicationId}/endpoints/*",
        {
          "pinpointApplicationId": {
            "Ref": "PinpointApplicationId"
          }
        }
      ]
```

```
    }
  }
]
```

PollyFullAccessPolicy

Gewährt vollen Zugriff auf Amazon Polly Polly-Lexikonressourcen.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "polly:GetLexicon",
      "polly:DeleteLexicon"
    ],
    "Resource": [
      {
        "Fn::Sub": [
          "arn:${AWS::Partition}:polly:${AWS::Region}:${AWS::AccountId}:lexicon/
${lexiconName}",
          {
            "lexiconName": {
              "Ref": "LexiconName"
            }
          }
        ]
      }
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "polly:DescribeVoices",
      "polly:ListLexicons",
      "polly:PutLexicon",
      "polly:SynthesizeSpeech"
    ],
    "Resource": [
      {
        "Fn::Sub": "arn:${AWS::Partition}:polly:${AWS::Region}:
${AWS::AccountId}:lexicon/*"
      }
    ]
  }
```

```
]
```

RekognitionDetectOnlyPolicy

Erteilt die Erlaubnis, Gesichter, Beschriftungen und Text zu erkennen.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "rekognition:DetectFaces",
      "rekognition:DetectLabels",
      "rekognition:DetectModerationLabels",
      "rekognition:DetectText"
    ],
    "Resource": "*"
  }
]
```

RekognitionFacesManagementPolicy

Erlaubt das Hinzufügen, Löschen und Suchen von Gesichtern in einer Amazon Rekognition Rekognition-Sammlung.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "rekognition:IndexFaces",
      "rekognition:DeleteFaces",
      "rekognition:SearchFaces",
      "rekognition:SearchFacesByImage",
      "rekognition:ListFaces"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:rekognition:${AWS::Region}:${AWS::AccountId}:collection/
${collectionId}",
        {
          "collectionId": {
            "Ref": "CollectionId"
          }
        }
```

```
      ]
    }
  }
]
```

RekognitionFacesPolicy

Erlaubt das Vergleichen und Erkennen von Gesichtern und Beschriftungen.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "rekognition:CompareFaces",
      "rekognition:DetectFaces"
    ],
    "Resource": "*"
  }
]
```

RekognitionLabelsPolicy

Erteilt die Erlaubnis, Objekt- und Moderationsbezeichnungen zu erkennen.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "rekognition:DetectLabels",
      "rekognition:DetectModerationLabels"
    ],
    "Resource": "*"
  }
]
```

RekognitionNoDataAccessPolicy

Erlaubt das Vergleichen und Erkennen von Gesichtern und Labels.

```
"Statement": [
  {
    "Effect": "Allow",
```

```
    "Action": [
      "rekognition:CompareFaces",
      "rekognition:DetectFaces",
      "rekognition:DetectLabels",
      "rekognition:DetectModerationLabels"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:rekognition:${AWS::Region}:${AWS::AccountId}:collection/
${collectionId}",
        {
          "collectionId": {
            "Ref": "CollectionId"
          }
        }
      ]
    }
  }
]
```

RekognitionReadPolicy

Erlaubt das Auflisten und Suchen von Gesichtern.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "rekognition:ListCollections",
      "rekognition:ListFaces",
      "rekognition:SearchFaces",
      "rekognition:SearchFacesByImage"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:rekognition:${AWS::Region}:${AWS::AccountId}:collection/
${collectionId}",
        {
          "collectionId": {
            "Ref": "CollectionId"
          }
        }
      ]
    }
```

```
  }
]
```

RekognitionWriteOnlyAccessPolicy

Erteilt die Erlaubnis, Gesichter in einer Sammlung zu erstellen und zu indexieren.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "rekognition:CreateCollection",
      "rekognition:IndexFaces"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:rekognition:${AWS::Region}:${AWS::AccountId}:collection/
${collectionId}",
        {
          "collectionId": {
            "Ref": "CollectionId"
          }
        }
      ]
    }
  }
]
```

Route53ChangeResourceRecordSetsPolicy

Erteilt die Erlaubnis, Ressourcendatensätze in Route 53 zu ändern.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "route53:ChangeResourceRecordSets"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:route53:::hostedzone/${HostedZoneId}",
        {
          "HostedZoneId": {
```

```
            "Ref": "HostedZoneId"
          }
        }
      ]
    }
  }
]
```

S3CrudPolicy

Erteilt Erstellungs-, Lese-, Aktualisierungs- und Löschberechtigungen, um auf die Objekte in einem Amazon S3 S3-Bucket zu reagieren.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
      "s3:ListBucket",
      "s3:GetBucketLocation",
      "s3:GetObjectVersion",
      "s3:PutObject",
      "s3:PutObjectAcl",
      "s3:GetLifecycleConfiguration",
      "s3:PutLifecycleConfiguration",
      "s3:DeleteObject"
    ],
    "Resource": [
      {
        "Fn::Sub": [
          "arn:${AWS::Partition}:s3:::${bucketName}",
          {
            "bucketName": {
              "Ref": "BucketName"
            }
          }
        ]
      },
      {
        "Fn::Sub": [
          "arn:${AWS::Partition}:s3:::${bucketName}/*",
          {
            "bucketName": {
```

```
              "Ref": "BucketName"
            }
          }
        ]
      }
    ]
  }
]
```

S3FullAccessPolicy

Erteilt die volle Zugriffsberechtigung, um auf die Objekte in einem Amazon S3 S3-Bucket zu reagieren.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
      "s3:GetObjectAcl",
      "s3:GetObjectVersion",
      "s3:PutObject",
      "s3:PutObjectAcl",
      "s3:DeleteObject",
      "s3:DeleteObjectTagging",
      "s3:DeleteObjectVersionTagging",
      "s3:GetObjectTagging",
      "s3:GetObjectVersionTagging",
      "s3:PutObjectTagging",
      "s3:PutObjectVersionTagging"
    ],
    "Resource": [
      {
        "Fn::Sub": [
          "arn:${AWS::Partition}:s3:::${bucketName}/*",
          {
            "bucketName": {
              "Ref": "BucketName"
            }
          }
        ]
      }
    ]
```

```
  },
  {
    "Effect": "Allow",
    "Action": [
      "s3:ListBucket",
      "s3:GetBucketLocation",
      "s3:GetLifecycleConfiguration",
      "s3:PutLifecycleConfiguration"
    ],
    "Resource": [
      {
        "Fn::Sub": [
          "arn:${AWS::Partition}:s3:::${bucketName}",
          {
            "bucketName": {
              "Ref": "BucketName"
            }
          }
        ]
      }
    ]
  }
]
```

S3ReadPolicy

Erteilt nur Leseberechtigungen zum Lesen von Objekten in einem Amazon Simple Storage Service (Amazon S3) -Bucket.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
      "s3:ListBucket",
      "s3:GetBucketLocation",
      "s3:GetObjectVersion",
      "s3:GetLifecycleConfiguration"
    ],
    "Resource": [
      {
        "Fn::Sub": [
          "arn:${AWS::Partition}:s3:::${bucketName}",
```

```
          {
            "bucketName": {
              "Ref": "BucketName"
            }
          }
        ]
      },
      {
        "Fn::Sub": [
          "arn:${AWS::Partition}:s3:::${bucketName}/*",
          {
            "bucketName": {
              "Ref": "BucketName"
            }
          }
        ]
      }
    ]
  }
]
```

S3WritePolicy

Erteilt Schreibberechtigungen zum Schreiben von Objekten in einen Amazon S3 S3-Bucket.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "s3:PutObject",
      "s3:PutObjectAcl",
      "s3:PutLifecycleConfiguration"
    ],
    "Resource": [
      {
        "Fn::Sub": [
          "arn:${AWS::Partition}:s3:::${bucketName}",
          {
            "bucketName": {
              "Ref": "BucketName"
            }
          }
        ]
      },
```

```
      {
        "Fn::Sub": [
          "arn:${AWS::Partition}:s3:::${bucketName}/*",
          {
            "bucketName": {
              "Ref": "BucketName"
            }
          }
        ]
      }
    ]
  }
]
```

SageMakerCreateEndpointConfigPolicy

Erteilt die Erlaubnis, eine Endpunktkonfiguration in SageMaker AI zu erstellen.

```
"Statement": [
  {
    "Action": [
      "sagemaker:CreateEndpointConfig"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:sagemaker:${AWS::Region}:${AWS::AccountId}:endpoint-
config/${endpointConfigName}",
        {
          "endpointConfigName": {
            "Ref": "EndpointConfigName"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

SageMakerCreateEndpointPolicy

Erteilt die Erlaubnis, einen Endpunkt in SageMaker KI zu erstellen.

```
"Statement": [
```

```
  {
    "Action": [
      "sagemaker:CreateEndpoint"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:sagemaker:${AWS::Region}:${AWS::AccountId}:endpoint/
${endpointName}",
        {
          "endpointName": {
            "Ref": "EndpointName"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

ServerlessRepoReadWriteAccessPolicy

Erteilt die Erlaubnis, Anwendungen im Dienst AWS Serverless Application Repository (AWS SAM) zu erstellen und aufzulisten.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "serverlessrepo:CreateApplication",
      "serverlessrepo:CreateApplicationVersion",
      "serverlessrepo:GetApplication",
      "serverlessrepo:ListApplications",
      "serverlessrepo:ListApplicationVersions"
    ],
    "Resource": [
      {
        "Fn::Sub": "arn:${AWS::Partition}:serverlessrepo:${AWS::Region}:
${AWS::AccountId}:applications/*"
      }
    ]
  }
]
```

SESBulkTemplatedCrudPolicy

Erteilt die Erlaubnis, Amazon SES SES-E-Mails, E-Mail-Vorlagen und Massen-E-Mails mit Vorlagen zu senden und die Identität zu überprüfen.

> **Note**
>
> Für die `ses:SendTemplatedEmail` Aktion ist ein Template-ARN erforderlich. Verwenden Sie stattdessen `SESBulkTemplatedCrudPolicy_v2`.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ses:GetIdentityVerificationAttributes",
      "ses:SendEmail",
      "ses:SendRawEmail",
      "ses:SendTemplatedEmail",
      "ses:SendBulkTemplatedEmail",
      "ses:VerifyEmailIdentity"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:ses:${AWS::Region}:${AWS::AccountId}:identity/
${identityName}",
        {
          "identityName": {
            "Ref": "IdentityName"
          }
        }
      ]
    }
  }
]
```

SESBulkTemplatedCrudPolicy_v2

Erteilt die Erlaubnis, Amazon SES SES-E-Mails, E-Mail-Vorlagen und Massen-E-Mails mit Vorlagen zu senden und die Identität zu überprüfen.

```
"Statement": [
```

```
  {
    "Action": [
      "ses:SendEmail",
      "ses:SendRawEmail",
      "ses:SendTemplatedEmail",
      "ses:SendBulkTemplatedEmail"
    ],
    "Effect": "Allow",
    "Resource": [
      {
        "Fn::Sub": [
          "arn:${AWS::Partition}:ses:${AWS::Region}:${AWS::AccountId}:identity/
${identityName}",
          {
            "identityName": {
              "Ref": "IdentityName"
            }
          }
        ]
      },
      {
        "Fn::Sub": [
          "arn:${AWS::Partition}:ses:${AWS::Region}:${AWS::AccountId}:template/
${templateName}",
          {
            "templateName": {
              "Ref": "TemplateName"
            }
          }
        ]
      }
    ]
  },
  {
    "Action": [
      "ses:GetIdentityVerificationAttributes",
      "ses:VerifyEmailIdentity"
    ],
    "Effect": "Allow",
    "Resource": "*"
  }
]
```

SESCrudPolicy

Erteilt die Erlaubnis, E-Mails zu senden und die Identität zu überprüfen.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ses:GetIdentityVerificationAttributes",
      "ses:SendEmail",
      "ses:SendRawEmail",
      "ses:VerifyEmailIdentity"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:ses:${AWS::Region}:${AWS::AccountId}:identity/
${identityName}",
        {
          "identityName": {
            "Ref": "IdentityName"
          }
        }
      ]
    }
  }
]
```

SESEmailTemplateCrudPolicy

Erteilt die Erlaubnis, Amazon SES SES-E-Mail-Vorlagen zu erstellen, abzurufen, aufzulisten, zu aktualisieren und zu löschen.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ses:CreateTemplate",
      "ses:GetTemplate",
      "ses:ListTemplates",
      "ses:UpdateTemplate",
      "ses:DeleteTemplate",
      "ses:TestRenderTemplate"
    ],
```

```
    "Resource": "*"
  }
]
```

SESSendBouncePolicy

SendBounce Erteilt die Erlaubnis für eine Amazon Simple Email Service (Amazon SES) -Identität.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ses:SendBounce"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:ses:${AWS::Region}:${AWS::AccountId}:identity/
${identityName}",
        {
          "identityName": {
            "Ref": "IdentityName"
          }
        }
      ]
    }
  }
]
```

SNSCrudPolicy

Erteilt die Erlaubnis, Amazon SNS SNS-Themen zu erstellen, zu veröffentlichen und zu abonnieren.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "sns:ListSubscriptionsByTopic",
      "sns:CreateTopic",
      "sns:SetTopicAttributes",
      "sns:Subscribe",
      "sns:Publish"
    ],
    "Resource": {
```

```
      "Fn::Sub": [
        "arn:${AWS::Partition}:sns:${AWS::Region}:${AWS::AccountId}:${topicName}*",
        {
          "topicName": {
            "Ref": "TopicName"
          }
        }
      ]
    }
  }
]
```

SNSPublishMessagePolicy

Erteilt die Erlaubnis, eine Nachricht zu einem Amazon Simple Notification Service (Amazon SNS) -Thema zu veröffentlichen.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "sns:Publish"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:sns:${AWS::Region}:${AWS::AccountId}:${topicName}",
        {
          "topicName": {
            "Ref": "TopicName"
          }
        }
      ]
    }
  }
]
```

SQSPollerPolicy

Erteilt die Erlaubnis, eine Amazon Simple Queue Service (Amazon SQS) -Warteschlange abzufragen.

```
"Statement": [
  {
```

```
    "Effect": "Allow",
    "Action": [
      "sqs:ChangeMessageVisibility",
      "sqs:ChangeMessageVisibilityBatch",
      "sqs:DeleteMessage",
      "sqs:DeleteMessageBatch",
      "sqs:GetQueueAttributes",
      "sqs:ReceiveMessage"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:sqs:${AWS::Region}:${AWS::AccountId}:${queueName}",
        {
          "queueName": {
            "Ref": "QueueName"
          }
        }
      ]
    }
  }
]
```

SQSSendMessagePolicy

Erteilt die Erlaubnis, Nachrichten an eine Amazon SQS SQS-Warteschlange zu senden.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "sqs:SendMessage*"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:sqs:${AWS::Region}:${AWS::AccountId}:${queueName}",
        {
          "queueName": {
            "Ref": "QueueName"
          }
        }
      ]
    }
  }
```

```
]
```

SSMParameterReadPolicy

Erteilt die Erlaubnis, auf einen Parameter aus einem Amazon EC2 Systems Manager (SSM) - Parameterspeicher zuzugreifen, um Geheimnisse in dieses Konto zu laden. Wird verwendet, wenn der Parametername kein Schrägstrich-Präfix hat.

> **Note**
>
> Wenn Sie keinen Standardschlüssel verwenden, benötigen Sie auch die `KMSDecryptPolicy` Richtlinie.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ssm:DescribeParameters"
    ],
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "ssm:GetParameters",
      "ssm:GetParameter",
      "ssm:GetParametersByPath"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:ssm:${AWS::Region}:${AWS::AccountId}:parameter/
${parameterName}",
        {
          "parameterName": {
            "Ref": "ParameterName"
          }
        }
      ]
    }
  }
```

```
]
```

SSMParameterWithSlashPrefixReadPolicy

Erteilt die Erlaubnis, auf einen Parameter aus einem Amazon EC2 Systems Manager (SSM) - Parameterspeicher zuzugreifen, um Geheimnisse in dieses Konto zu laden. Wird verwendet, wenn der Parametername ein Schrägstrich-Präfix hat.

> **Note**
>
> Wenn Sie keinen Standardschlüssel verwenden, benötigen Sie auch die `KMSDecryptPolicy` Richtlinie.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ssm:DescribeParameters"
    ],
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "ssm:GetParameters",
      "ssm:GetParameter",
      "ssm:GetParametersByPath"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:ssm:${AWS::Region}:${AWS::AccountId}:parameter
${parameterName}",
        {
          "parameterName": {
            "Ref": "ParameterName"
          }
        }
      ]
    }
  }
```

```
]
```

StepFunctionsExecutionPolicy

Erteilt die Erlaubnis, die Ausführung einer Step Functions Functions-Zustandsmaschine zu starten.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "states:StartExecution"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:states:${AWS::Region}:${AWS::AccountId}:stateMachine:
${stateMachineName}",
        {
          "stateMachineName": {
            "Ref": "StateMachineName"
          }
        }
      ]
    }
  }
]
```

TextractDetectAnalyzePolicy

Ermöglicht den Zugriff auf die Erkennung und Analyse von Dokumenten mit Amazon Textract.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "textract:DetectDocumentText",
      "textract:StartDocumentTextDetection",
      "textract:StartDocumentAnalysis",
      "textract:AnalyzeDocument"
    ],
    "Resource": "*"
  }
]
```

TextractGetResultPolicy

Ermöglicht den Zugriff auf erkannte und analysierte Dokumente von Amazon Textract.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "textract:GetDocumentTextDetection",
      "textract:GetDocumentAnalysis"
    ],
    "Resource": "*"
  }
]
```

TextractPolicy

Gewährt vollen Zugriff auf Amazon Textract.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "textract:*"
    ],
    "Resource": "*"
  }
]
```

VPCAccessPolicy

Ermöglicht das Erstellen, Löschen, Beschreiben und Trennen von Elastic Network-Schnittstellen.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ec2:CreateNetworkInterface",
      "ec2:DeleteNetworkInterface",
      "ec2:DescribeNetworkInterfaces",
      "ec2:DetachNetworkInterface"
    ],
    "Resource": "*"
```

```
  }
]
```

Verwaltung von AWS SAM Berechtigungen mit CloudFormation Mechanismen

Um den Zugriff auf AWS Ressourcen zu kontrollieren, kann AWS Serverless Application Model (AWS SAM) dieselben Mechanismen verwenden wie. CloudFormation Weitere Informationen finden Sie unter Steuern des Zugriffs mit AWS Identity and Access Management im Benutzerhandbuch für AWS CloudFormation .

Es gibt drei Hauptoptionen, um einem Benutzer die Berechtigung zur Verwaltung serverloser Anwendungen zu erteilen. Jede Option bietet Benutzern unterschiedliche Stufen der Zugriffskontrolle.

- Gewähren Sie Administratorrechte.
- Hängen Sie die erforderlichen AWS verwalteten Richtlinien an.
- Erteilen Sie spezifische AWS Identity and Access Management (IAM-) Berechtigungen.

Je nachdem, welche Option Sie wählen, können Benutzer nur serverlose Anwendungen verwalten, die AWS Ressourcen enthalten, für deren Zugriff sie berechtigt sind.

In den folgenden Abschnitten werden die einzelnen Optionen ausführlicher beschrieben.

Gewähren Sie Administratorrechte

Wenn Sie einem Benutzer Administratorrechte gewähren, kann er serverlose Anwendungen verwalten, die eine beliebige Kombination von AWS Ressourcen enthalten. Dies ist die einfachste Option, gewährt Benutzern jedoch auch die umfassendsten Berechtigungen, sodass sie Aktionen mit der größten Wirkung ausführen können.

Weitere Informationen zum Erteilen von Administratorberechtigungen für einen Benutzer finden Sie unter Erstellen Ihres ersten IAM-Administratorbenutzers und Ihrer ersten Gruppe im IAM-Benutzerhandbuch.

Fügen Sie die erforderlichen verwalteten Richtlinien AWS hinzu

Sie können Benutzern mithilfe AWS verwalteter Richtlinien eine Teilmenge von Berechtigungen gewähren, anstatt ihnen vollständige Administratorrechte zu gewähren. Wenn Sie diese Option verwenden, stellen Sie sicher, dass der Satz AWS verwalteter Richtlinien alle Aktionen und Ressourcen abdeckt, die für die serverlosen Anwendungen erforderlich sind, die die Benutzer verwalten.

Die folgenden AWS verwalteten Richtlinien reichen beispielsweise aus, um die Hello World-Beispielanwendung bereitzustellen:

- AWSCloudFormationFullAccess
- IAMFullZugriff
- AWSLambda_FullAccess
- APIGatewayAmazon-Administrator
- Amazon S3 FullAccess
- Amazon EC2 ContainerRegistryFullAccess

Informationen zum Anhängen von Richtlinien an einen IAM-Benutzer finden Sie unter Ändern der Berechtigungen für einen IAM-Benutzer im IAM-Benutzerhandbuch.

Erteilen Sie bestimmte IAM-Berechtigungen

Für eine möglichst detaillierte Zugriffskontrolle können Sie Benutzern mithilfe von Richtlinienanweisungen bestimmte IAM-Berechtigungen gewähren. Wenn Sie diese Option verwenden, stellen Sie sicher, dass die Richtlinienanweisung alle Aktionen und Ressourcen enthält, die für die serverlosen Anwendungen erforderlich sind, die die Benutzer verwalten.

Die bewährte Methode bei dieser Option besteht darin, Benutzern die Berechtigung zum Erstellen von Rollen zu verweigern, einschließlich Lambda-Ausführungsrollen, sodass sie sich keine eskalierten Berechtigungen gewähren können. Sie als Administrator müssen also zunächst eine Lambda-Ausführungsrolle erstellen, die in den serverlosen Anwendungen angegeben wird, die Benutzer verwalten werden. Informationen zum Erstellen von Lambda-Ausführungsrollen finden Sie unter Ausführungsrolle in der IAM-Konsole erstellen.

Für die Hello World-Beispielanwendung AWSLambdaBasicExecutionRolereicht es aus, die Anwendung auszuführen. Nachdem Sie eine Lambda-Ausführungsrolle erstellt haben, ändern Sie die AWS SAM Vorlagendatei der Hello World-Beispielanwendung, um der `AWS::Serverless::Function` Ressource die folgende Eigenschaft hinzuzufügen:

```
Role: lambda-execution-role-arn
```

Sobald die geänderte Hello World-Anwendung installiert ist, gewährt die folgende Richtlinienerklärung Benutzern ausreichende Berechtigungen, um die Anwendung bereitzustellen, zu aktualisieren und zu löschen:

> **Note**
>
> Die Beispielrichtlinie in diesem Abschnitt gewährt Ihnen ausreichende Berechtigungen, um die Hello World-Beispielanwendung bereitzustellen, zu aktualisieren und zu löschen. Wenn Sie Ihrer Anwendung zusätzliche Ressourcentypen hinzufügen, müssen Sie die Richtlinienerklärung so aktualisieren, dass sie Folgendes enthält:
>
> 1. Erlaubnis für Ihre Anwendung, die Aktionen des Dienstes aufzurufen.
> 2. Der Dienstprinzipal, falls er für die Aktionen des Dienstes benötigt wird.
>
> Wenn Sie beispielsweise einen Step Functions-Workflow hinzufügen, müssen Sie möglicherweise Berechtigungen für die hier aufgeführten Aktionen und den `states.amazonaws.com` Dienstprinzipal hinzufügen.

Weitere Informationen zu IAM-Richtlinien finden Sie unter Verwaltung von IAM-Richtlinien im IAM-Benutzerhandbuch.

Kontrollieren Sie den API-Zugriff mit Ihrer AWS SAM Vorlage

Durch die Steuerung des Zugriffs auf Ihr API Gateway wird APIs sichergestellt, dass Ihre serverlose Anwendung sicher ist und nur mit der von Ihnen aktivierten Autorisierung darauf zugegriffen werden kann. Sie können die Autorisierung in Ihrer AWS SAM Vorlage aktivieren, um zu kontrollieren, wer auf Ihr API Gateway zugreifen kann APIs.

AWS SAM unterstützt mehrere Mechanismen zur Steuerung des Zugriffs auf Ihr API Gateway APIs. Die unterstützten Mechanismen unterscheiden `AWS::Serverless::HttpApi` sich je nach `AWS::Serverless::Api` Ressourcentyp.

In der folgenden Tabelle sind die Mechanismen zusammengefasst, die von den einzelnen Ressourcentypen unterstützt werden.

Mechanismen zur Zugriffsk ontrolle	AWS::Serverless::HttpApi	AWS::Serverless::Api
Lambda-Autorisierer	✓	✓
IAM-Berechtigungen		✓

Mechanismen zur Zugriffsk ontrolle	AWS::Serverless::HttpApi	AWS::Serverless::Api
Amazon-Cognito-Benutzerpool s	✓ *	✓
API-Schlüssel		✓
Ressourcenrichtlinien		✓
OAuth 2.0/JWT-Autorisierer	✓	

* Sie können Amazon Cognito als Aussteller von JSON Web Token (JWT) mit dem `AWS::Serverless::HttpApi` Ressourcentyp verwenden.

- Lambda-Autorisierer — Ein Lambda-Autorisierer (früher bekannt als benutzerdefinierter Autorisierer) ist eine Lambda-Funktion, die Sie bereitstellen, um den Zugriff auf Ihre API zu kontrollieren. Wenn Ihre API aufgerufen wird, wird diese Lambda-Funktion mit einem Anforderungskontext oder einem Autorisierungstoken aufgerufen, das die Client-Anwendung bereitstellt. Die Lambda-Funktion gibt an, ob der Aufrufer berechtigt ist, den angeforderten Vorgang auszuführen.

  `AWS::Serverless::HttpApi`Sowohl der Ressourcentyp als auch der `AWS::Serverless::Api` Ressourcentyp unterstützen Lambda-Autorisierer.

  Weitere Informationen zu Lambda-Autorisierern mit finden Sie unter Arbeiten mit *`AWS::Serverless::HttpApi`*AWS Lambda Autorisierern für HTTP APIs im API Gateway Developer Guide. Weitere Informationen zu Lambda-Autorisierern mit *`AWS::Serverless::Api`* finden Sie unter Verwenden von API Gateway-Lambda-Autorisierern im API Gateway Developer Guide.

  Beispiele für Lambda-Autorisierer für beide Ressourcentypen finden Sie unter. Beispiele für Lambda-Autorisierer für AWS SAM

- IAM-Berechtigungen — Sie können mithilfe von (IAM-) Berechtigungen steuern, wer Ihre API AWS Identity and Access Management aufrufen kann. Benutzer, die Ihre API aufrufen, müssen mit IAM-Anmeldeinformationen authentifiziert werden. Aufrufe Ihrer API sind nur erfolgreich, wenn dem

IAM-Benutzer eine IAM-Richtlinie zugeordnet ist, die den API-Aufrufer repräsentiert, eine IAM-Gruppe, die den Benutzer enthält, oder eine IAM-Rolle, die der Benutzer annimmt.

Nur der `AWS::Serverless::Api` Ressourcentyp unterstützt IAM-Berechtigungen.

Weitere Informationen finden Sie unter Steuern des Zugriffs auf eine API mit IAM-Berechtigungen im API Gateway Developer Guide. Ein Beispiel finden Sie unter Beispiel für eine IAM-Berechtigung für AWS SAM.

- Amazon Cognito-Benutzerpools — Amazon Cognito Cognito-Benutzerpools sind Benutzerverzeichnisse in Amazon Cognito. Ein Client Ihrer API muss zuerst einen Benutzer beim Benutzerpool anmelden und ein Identitäts- oder Zugriffstoken für den Benutzer erhalten. Dann ruft der Client Ihre API mit einem der zurückgegebenen Token auf. Der API-Aufruf ist nur erfolgreich, wenn das erforderliche Token gültig ist.

  Der `AWS::Serverless::Api` Ressourcentyp unterstützt Amazon Cognito Cognito-Benutzerpools. Der `AWS::Serverless::HttpApi` Ressourcentyp unterstützt die Verwendung von Amazon Cognito als JWT-Emittenten.

  Weitere Informationen finden Sie unter Control access to a REST API using Amazon cognito user pools as authorizer (Zugriff auf eine REST-API mit Amazon-Cognito-Benutzerpools als Autorisierer) im Entwicklerhandbuch von API Gateway. Ein Beispiel finden Sie unter Beispiel für einen Amazon Cognito Cognito-Benutzerpool für AWS SAM.
- API-Schlüssel — API-Schlüssel sind alphanumerische Zeichenkettenwerte, die Sie an Kunden von Anwendungsentwicklern verteilen, um Zugriff auf Ihre API zu gewähren.

  Nur der `AWS::Serverless::Api` Ressourcentyp unterstützt API-Schlüssel.

  Weitere Informationen zu API-Schlüsseln finden Sie unter Nutzungspläne mit API-Schlüsseln erstellen und verwenden im API Gateway Developer Guide. Ein Beispiel für API-Schlüssel finden Sie unterBeispiel für einen API-Schlüssel AWS SAM.
- Ressourcenrichtlinien — Ressourcenrichtlinien sind JSON-Richtliniendokumente, die Sie an eine API Gateway anhängen können. Verwenden Sie Ressourcenrichtlinien, um zu steuern, ob ein bestimmter Prinzipal (normalerweise ein IAM-Benutzer oder eine IAM-Rolle) die API aufrufen kann.

  Nur der `AWS::Serverless::Api` Ressourcentyp unterstützt Ressourcenrichtlinien als Mechanismus zur Steuerung des Zugriffs auf API Gateway APIs.

Weitere Informationen zu Ressourcenrichtlinien finden Sie unter Steuern des Zugriffs auf eine API mit API-Gateway-Ressourcenrichtlinien im API Gateway Developer Guide. Ein Beispiel für Ressourcenrichtlinien finden Sie unterBeispiel für eine Ressourcenrichtlinie für AWS SAM.

- OAuth 2.0/JWT-Autorisierer — Sie können sie JWTs als Teil der OpenID Connect (OIDC) - und OAuth 2.0-Frameworks verwenden, um den Zugriff auf Ihre zu kontrollieren. APIs API Gateway validiert die JWTs , die Clients mit API-Anfragen einreichen, und erlaubt oder lehnt Anfragen auf der Grundlage der Tokenvalidierung und optional der Bereiche im Token ab.

  Nur der `AWS::Serverless::HttpApi` Ressourcentyp unterstützt OAuth 2.0/JWT-Autorisierer.

  Weitere Informationen finden Sie unter Steuern des Zugriffs auf HTTP APIs mit JWT-Autorisierern im API Gateway Developer Guide. Ein Beispiel finden Sie unter OAuth 2.0/JWT-Autorisierungsbeispiel für AWS SAM.

Auswahl eines Mechanismus zur Zugriffskontrolle

Der Mechanismus, den Sie für die Steuerung des Zugriffs auf Ihr API Gateway verwenden, APIs hängt von einigen Faktoren ab. Wenn Sie beispielsweise ein Projekt auf der grünen Wiese haben, für das weder Autorisierung noch Zugriffskontrolle eingerichtet sind, sind Amazon Cognito Cognito-Benutzerpools möglicherweise die beste Option. Das liegt daran, dass Sie bei der Einrichtung von Benutzerpools auch automatisch sowohl die Authentifizierung als auch die Zugriffskontrolle einrichten.

Wenn für Ihre Anwendung jedoch bereits eine Authentifizierung eingerichtet ist, ist die Verwendung von Lambda-Autorisierern möglicherweise die beste Option. Dies liegt daran, dass Sie Ihren bestehenden Authentifizierungsdienst anrufen und auf der Grundlage der Antwort ein Richtliniendokument zurücksenden können. Wenn Ihre Anwendung eine benutzerdefinierte Authentifizierungs- oder Zugriffskontrolllogik erfordert, die Benutzerpools nicht unterstützen, sind Lambda-Autorisierer möglicherweise die beste Option.

Wenn Sie den zu verwendenden Mechanismus ausgewählt haben, finden Sie im entsprechenden Abschnitt unter Informationen AWS SAM zur Konfiguration Ihrer Anwendung Beispiele für die Verwendung dieses Mechanismus.

Anpassen von Fehlerantworten

Sie können AWS SAM damit den Inhalt einiger API-Gateway-Fehlerantworten anpassen. Nur der `AWS::Serverless::Api` Ressourcentyp unterstützt benutzerdefinierte API-Gateway-Antworten.

Weitere Informationen zu API-Gateway-Antworten finden Sie unter Gateway-Antworten in API Gateway im API Gateway Developer Guide. Ein Beispiel für benutzerdefinierte Antworten finden Sie unterBeispiel für eine benutzerdefinierte Antwort für AWS SAM.

Beispiele

- Beispiele für Lambda-Autorisierer für AWS SAM
- Beispiel für eine IAM-Berechtigung für AWS SAM
- Beispiel für einen Amazon Cognito Cognito-Benutzerpool für AWS SAM
- Beispiel für einen API-Schlüssel AWS SAM
- Beispiel für eine Ressourcenrichtlinie für AWS SAM
- OAuth 2.0/JWT-Autorisierungsbeispiel für AWS SAM
- Beispiel für eine benutzerdefinierte Antwort für AWS SAM

Beispiele für Lambda-Autorisierer für AWS SAM

Der `AWS::Serverless::Api` Ressourcentyp unterstützt zwei Arten von Lambda-Autorisierern: Autorisierer und `TOKEN` Autorisierer. `REQUEST` Der Ressourcentyp unterstützt nur Autorisierer. `AWS::Serverless::HttpApi` `REQUEST` Im Folgenden finden Sie Beispiele für jeden Typ.

Beispiel für einen **`TOKEN`** Lambda-Authorizer () AWS::Serverless::Api

Sie können den Zugriff auf Ihre steuern, APIs indem Sie in Ihrer AWS SAM Vorlage einen `TOKEN` Lambda-Autorisierer definieren. Dazu verwenden Sie den ApiAuth Datentyp.

Im Folgenden finden Sie ein Beispiel für einen AWS SAM Vorlagenabschnitt für einen `TOKEN` Lambda-Autorisierer:

> **Note**
>
> Im folgenden Beispiel wird das SAM implizit `FunctionRole` generiert.

```
Resources:
  MyApi:
    Type: AWS::Serverless::Api
```

```
    Properties:
      StageName: Prod
      Auth:
        DefaultAuthorizer: MyLambdaTokenAuthorizer
        Authorizers:
          MyLambdaTokenAuthorizer:
            FunctionArn: !GetAtt MyAuthFunction.Arn

  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: ./src
      Handler: index.handler
      Runtime: nodejs12.x
      Events:
        GetRoot:
          Type: Api
          Properties:
            RestApiId: !Ref MyApi
            Path: /
            Method: get

  MyAuthFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: ./src
      Handler: authorizer.handler
      Runtime: nodejs12.x
```

Weitere Informationen zu Lambda-Autorisierern finden Sie unter Verwenden von API Gateway-Lambda-Autorisierern im API Gateway Developer Guide.

Beispiel für einen **`REQUEST`** Lambda-Authorizer () AWS::Serverless::Api

Sie können den Zugriff auf Ihre steuern, APIs indem Sie in Ihrer AWS SAM Vorlage einen `REQUEST` Lambda-Autorisierer definieren. Dazu verwenden Sie den ApiAuth Datentyp.

Im Folgenden finden Sie ein Beispiel für einen AWS SAM Vorlagenabschnitt für einen `REQUEST` Lambda-Autorisierer:

```
Resources:
  MyApi:
    Type: AWS::Serverless::Api
```

```
    Properties:
      StageName: Prod
      Auth:
        DefaultAuthorizer: MyLambdaRequestAuthorizer
        Authorizers:
          MyLambdaRequestAuthorizer:
            FunctionPayloadType: REQUEST
            FunctionArn: !GetAtt MyAuthFunction.Arn
            Identity:
              QueryStrings:
                - auth

  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: ./src
      Handler: index.handler
      Runtime: nodejs12.x
      Events:
        GetRoot:
          Type: Api
          Properties:
            RestApiId: !Ref MyApi
            Path: /
            Method: get

  MyAuthFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: ./src
      Handler: authorizer.handler
      Runtime: nodejs12.x
```

Weitere Informationen zu Lambda-Autorisierern finden Sie unter Verwenden von API Gateway-Lambda-Autorisierern im API Gateway Developer Guide.

Beispiel für einen Lambda-Authorizer () AWS::Serverless::HttpApi

Sie können den Zugriff auf Ihr HTTP steuern, APIs indem Sie in Ihrer AWS SAM Vorlage einen Lambda-Authorizer definieren. Dazu verwenden Sie den HttpApiAuth Datentyp.

Im Folgenden finden Sie ein Beispiel für einen AWS SAM Vorlagenabschnitt für einen Lambda-Autorisierer:

```
Resources:
  MyApi:
    Type: AWS::Serverless::HttpApi
    Properties:
      StageName: Prod
      Auth:
        DefaultAuthorizer: MyLambdaRequestAuthorizer
        Authorizers:
          MyLambdaRequestAuthorizer:
            FunctionArn: !GetAtt MyAuthFunction.Arn
            FunctionInvokeRole: !GetAtt MyAuthFunctionRole.Arn
            Identity:
              Headers:
                - Authorization
            AuthorizerPayloadFormatVersion: 2.0
            EnableSimpleResponses: true

  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: ./src
      Handler: index.handler
      Runtime: nodejs12.x
      Events:
        GetRoot:
          Type: HttpApi
          Properties:
            ApiId: !Ref MyApi
            Path: /
            Method: get
            PayloadFormatVersion: "2.0"

  MyAuthFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: ./src
      Handler: authorizer.handler
      Runtime: nodejs12.x
```

Beispiel für eine IAM-Berechtigung für AWS SAM

Sie können den Zugriff auf Ihre steuern, APIs indem Sie IAM-Berechtigungen in Ihrer AWS SAM Vorlage definieren. Dazu verwenden Sie den ApiAuth Datentyp.

Im Folgenden finden Sie eine AWS SAM Beispielvorlage, die für IAM-Berechtigungen verwendet wird:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
  MyApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      Description: 'API with IAM authorization'
      Auth:
        DefaultAuthorizer: AWS_IAM #sets AWS_IAM auth for all methods in this API
  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.handler
      Runtime: python3.10
      Events:
        GetRoot:
          Type: Api
          Properties:
            RestApiId: !Ref MyApi
            Path: /
            Method: get
      InlineCode: |
        def handler(event, context):
          return {'body': 'Hello World!', 'statusCode': 200}
```

Weitere Informationen zu IAM-Berechtigungen finden Sie unter Zugriffskontrolle für das Aufrufen einer API im API Gateway Developer Guide.

Beispiel für einen Amazon Cognito Cognito-Benutzerpool für AWS SAM

Sie können den Zugriff auf Ihre steuern, APIs indem Sie Amazon Cognito Cognito-Benutzerpools in Ihrer AWS SAM Vorlage definieren. Dazu verwenden Sie den ApiAuth Datentyp.

Im Folgenden finden Sie ein Beispiel für einen AWS SAM Vorlagenabschnitt für einen Benutzerpool:

```
Resources:
  MyApi:
    Type: AWS::Serverless::Api
    Properties:
```

```
      StageName: Prod
      Cors: "'*'"
      Auth:
        DefaultAuthorizer: MyCognitoAuthorizer
        Authorizers:
          MyCognitoAuthorizer:
            UserPoolArn: !GetAtt MyCognitoUserPool.Arn

  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: ./src
      Handler: lambda.handler
      Runtime: nodejs12.x
      Events:
        Root:
          Type: Api
          Properties:
            RestApiId: !Ref MyApi
            Path: /
            Method: GET

  MyCognitoUserPool:
    Type: AWS::Cognito::UserPool
    Properties:
      UserPoolName: !Ref CognitoUserPoolName
      Policies:
        PasswordPolicy:
          MinimumLength: 8
      UsernameAttributes:
        - email
      Schema:
        - AttributeDataType: String
          Name: email
          Required: false

  MyCognitoUserPoolClient:
    Type: AWS::Cognito::UserPoolClient
    Properties:
      UserPoolId: !Ref MyCognitoUserPool
      ClientName: !Ref CognitoUserPoolClientName
      GenerateSecret: false
```

Weitere Informationen zu Amazon Cognito Cognito-Benutzerpools finden Sie unter Steuern des Zugriffs auf eine REST-API mithilfe von Amazon Cognito Cognito-Benutzerpools als Autorisierer im API Gateway Developer Guide.

Beispiel für einen API-Schlüssel AWS SAM

Sie können den Zugriff auf Ihre steuern, APIs indem Sie API-Schlüssel in Ihrer AWS SAM Vorlage angeben. Dazu verwenden Sie den ApiAuth Datentyp.

Im Folgenden finden Sie ein Beispiel für einen AWS SAM Vorlagenabschnitt für API-Schlüssel:

```
Resources:
  MyApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      Auth:
        ApiKeyRequired: true # sets for all methods

  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: .
      Handler: index.handler
      Runtime: nodejs12.x
      Events:
        ApiKey:
          Type: Api
          Properties:
            RestApiId: !Ref MyApi
            Path: /
            Method: get
            Auth:
              ApiKeyRequired: true
```

Weitere Informationen zu API-Schlüsseln finden Sie unter Nutzungspläne mit API-Schlüsseln erstellen und verwenden im API Gateway Developer Guide.

Beispiel für eine Ressourcenrichtlinie für AWS SAM

Sie können den Zugriff auf Ihre steuern, APIs indem Sie Ihrer AWS SAM Vorlage eine Ressourcenrichtlinie anhängen. Dazu verwenden Sie den ApiAuth Datentyp.

Im Folgenden finden Sie eine AWS SAM Beispielvorlage für eine private API. Eine private API muss über eine Ressourcenrichtlinie verfügen, damit sie bereitgestellt werden kann.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
  MyPrivateApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      EndpointConfiguration: PRIVATE  # Creates a private API. Resource policies are
 required for all private APIs.
      Auth:
        ResourcePolicy:
          CustomStatements:
            - Effect: 'Allow'
              Action: 'execute-api:Invoke'
              Resource: ['execute-api:/*/*/*']
              Principal: '*'
            - Effect: 'Deny'
              Action: 'execute-api:Invoke'
              Resource: ['execute-api:/*/*/*']
              Principal: '*'
  MyFunction:
    Type: 'AWS::Serverless::Function'
    Properties:
      InlineCode: |
        def handler(event, context):
          return {'body': 'Hello World!', 'statusCode': 200}
      Handler: index.handler
      Runtime: python3.10
      Events:
        AddItem:
          Type: Api
          Properties:
            RestApiId:
              Ref: MyPrivateApi
            Path: /
            Method: get
```

Weitere Informationen zu Ressourcenrichtlinien finden Sie unter Steuern des Zugriffs auf eine API mit API-Gateway-Ressourcenrichtlinien im API Gateway Developer Guide. Weitere Informationen

zu Private APIs finden Sie unter Creating a private API in Amazon API Gateway im API Gateway Developer Guide.

OAuth 2.0/JWT-Autorisierungsbeispiel für AWS SAM

Sie können den Zugriff auf Ihre APIs Nutzung JWTs als Teil der Frameworks OpenID Connect (OIDC) und OAuth 2.0 kontrollieren. Dazu verwenden Sie den Datentyp. HttpApiAuth

Im Folgenden finden Sie ein Beispiel für einen AWS SAM Vorlagenabschnitt für einen OAuth 2.0/JWT-Authorizer:

```
Resources:
  MyApi:
    Type: AWS::Serverless::HttpApi
    Properties:
      Auth:
        Authorizers:
          MyOauth2Authorizer:
            AuthorizationScopes:
              - scope
            IdentitySource: $request.header.Authorization
            JwtConfiguration:
              audience:
                - audience1
                - audience2
              issuer: "https://www.example.com/v1/connect/oidc"
        DefaultAuthorizer: MyOauth2Authorizer
      StageName: Prod
  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: ./src
      Events:
        GetRoot:
          Properties:
            ApiId: MyApi
            Method: get
            Path: /
            PayloadFormatVersion: "2.0"
          Type: HttpApi
      Handler: index.handler
      Runtime: nodejs12.x
```

Weitere Informationen zu OAuth 2.0/JWT-Autorisierern finden Sie unter Steuern des Zugriffs auf HTTP APIs mit JWT-Autorisierern im API Gateway Developer Guide.

Beispiel für eine benutzerdefinierte Antwort für AWS SAM

Sie können einige API-Gateway-Fehlerantworten anpassen, indem Sie Antwortheader in Ihrer AWS SAM Vorlage definieren. Dazu verwenden Sie den Datentyp Gateway Response Object.

Im Folgenden finden Sie eine AWS SAM Beispielvorlage, die eine benutzerdefinierte Antwort für den `DEFAULT_5XX` Fehler erstellt.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
  MyApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      GatewayResponses:
        DEFAULT_5XX:
          ResponseParameters:
            Headers:
              Access-Control-Expose-Headers: "'WWW-Authenticate'"
              Access-Control-Allow-Origin: "'*'"
              ErrorHeader: "'MyCustomErrorHeader'"
          ResponseTemplates:
            application/json: "{\"message\": \"Error on the $context.resourcePath
 resource\" }"

  GetFunction:
    Type: AWS::Serverless::Function
    Properties:
      Runtime: python3.10
      Handler: index.handler
      InlineCode: |
        def handler(event, context):
          raise Exception('Check out the new response!')
      Events:
        GetResource:
          Type: Api
          Properties:
            Path: /error
            Method: get
```

```
            RestApiId: !Ref MyApi
```

Weitere Informationen zu API-Gateway-Antworten finden Sie unter Gateway-Antworten in API Gateway im API Gateway Developer Guide.

Steigern Sie die Effizienz mithilfe von Lambda-Schichten mit AWS SAM

Mithilfe von AWS SAM können Sie Ebenen in Ihre serverlosen Anwendungen einbeziehen. AWS Lambda Ebenen ermöglichen es Ihnen, Code aus einer Lambda-Funktion in eine Lambda-Schicht zu extrahieren, die dann für mehrere Lambda-Funktionen verwendet werden kann. Auf diese Weise können Sie die Größe Ihrer Bereitstellungspakete reduzieren, die Logik der Kernfunktionen von Abhängigkeiten trennen und Abhängigkeiten für mehrere Funktionen gemeinsam nutzen. Weitere Informationen zu Layern finden Sie unter Lambda-Schichten im AWS Lambda Developer Guide.

Dieses Thema enthält Informationen zu folgenden Themen:

- Ebenen in Ihre Anwendung einbeziehen
- Wie werden Ebenen lokal zwischengespeichert

Informationen zum Erstellen von benutzerdefinierten Layern finden Sie unterAufbau von Lambda-Schichten in AWS SAM.

Ebenen in Ihre Anwendung einbeziehen

Verwenden Sie die `Layers` Eigenschaft des AWS::Serverless: :Funktion Ressourcentyps, um Ebenen in Ihre Anwendung aufzunehmen.

Im Folgenden finden Sie eine AWS SAM Beispielvorlage mit einer Lambda-Funktion, die eine Ebene enthält:

```
ServerlessFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: .
    Handler: my_handler
    Runtime: Python3.7
    Layers:
        - <LayerVersion ARN>
```

Wie werden Ebenen lokal zwischengespeichert

Wenn Sie Ihre Funktion mit einem der `sam local` Befehle aufrufen, wird das Layer-Paket Ihrer Funktion heruntergeladen und auf Ihrem lokalen Host zwischengespeichert.

Die folgende Tabelle zeigt die Standard-Cache-Verzeichnisspeicherorte für verschiedene Betriebssysteme.

BS	Speicherort
Windows 7	`C:\Users\<user>\AppData\Roaming\AWS SAM`
Windows 8	`C:\Users\<user>\AppData\Roaming\AWS SAM`
Windows 10	`C:\Users\<user>\AppData\Roaming\AWS SAM`
macOS	`~/.aws-sam/layers-pkg`
Unix	`~/.aws-sam/layers-pkg`

Nachdem das Paket zwischengespeichert wurde, werden die Ebenen auf ein Docker-Image AWS SAMCLI überlagert, das zum Aufrufen Ihrer Funktion verwendet wird. Das AWS SAMCLI generiert die Namen der Bilder, die es erstellt, sowie der Bilder, LayerVersions die im Cache gespeichert sind. Weitere Informationen zum Schema finden Sie in den folgenden Abschnitten.

Um die überlagerten Ebenen zu untersuchen, führen Sie den folgenden Befehl aus, um eine Bash-Sitzung in dem Bild zu starten, das Sie untersuchen möchten:

```
docker run -it --entrypoint=/bin/bash samcli/lambda:<Tag following the schema outlined
 in Docker Image Tag Schema> -i
```

Layer-Caching: Namensschema für Verzeichnisse

Bei gegebener LayerVersionArn Angabe, die in Ihrer Vorlage definiert ist, AWS SAMCLI extrahiert das die LayerName und Version aus dem ARN. Es erstellt ein Verzeichnis, in dem der Layer-Inhalt abgelegt wird`LayerName-Version-<first 10 characters of sha256 of ARN>`.

Beispiel:

```
ARN = arn:aws:lambda:us-west-2:111111111111:layer:myLayer:1
Directory name = myLayer-1-926eeb5ff1
```

Tag-Schema für Docker Images

Um den eindeutigen Layer-Hash zu berechnen, kombinieren Sie alle eindeutigen Layer-Namen mit dem Trennzeichen '-', nehmen Sie den SHA256 Hash und dann die ersten 10 Zeichen.

Beispiel:

```
ServerlessFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: .
    Handler: my_handler
    Runtime: Python3.7
    Layers:
        - arn:aws:lambda:us-west-2:111111111111:layer:myLayer:1
        - arn:aws:lambda:us-west-2:111111111111:layer:mySecondLayer:1
```

Eindeutige Namen werden genauso berechnet wie das Namensschema für das Layer-Caching-Verzeichnis:

```
arn:aws:lambda:us-west-2:111111111111:layer:myLayer:1 = myLayer-1-926eeb5ff1
arn:aws:lambda:us-west-2:111111111111:layer:mySecondLayer:1 =
 mySecondLayer-1-6bc1022bdf
```

Um den eindeutigen Layer-Hash zu berechnen, kombinieren Sie alle eindeutigen Layer-Namen mit dem Trennzeichen '-', nehmen Sie den SHA256-Hash und dann die ersten 25 Zeichen:

```
myLayer-1-926eeb5ff1-mySecondLayer-1-6bc1022bdf = 2dd7ac5ffb30d515926aef
```

Kombinieren Sie dann diesen Wert mit der Laufzeit und Architektur der Funktion mit dem Trennzeichen '-':

```
python3.7-x86_64-2dd7ac5ffb30d515926aefffd
```

Verwenden Sie Code und Ressourcen mithilfe verschachtelter Anwendungen in AWS SAM

Eine serverlose Anwendung kann eine oder mehrere verschachtelte Anwendungen enthalten. Eine verschachtelte Anwendung ist Teil einer größeren Anwendung und kann entweder als eigenständiges Artefakt oder als Komponente einer größeren Anwendung gepackt und bereitgestellt werden. Verschachtelte Anwendungen ermöglichen es Ihnen, häufig verwendeten Code in eine eigene Anwendung umzuwandeln, die dann in einer größeren serverlosen Anwendung oder mehreren serverlosen Anwendungen wiederverwendet werden kann.

Wenn Ihre serverlosen Architekturen wachsen, entstehen in der Regel allgemeine Muster, bei denen dieselben Komponenten in mehreren Anwendungsvorlagen definiert sind. Verschachtelte Anwendungen ermöglichen es Ihnen, gemeinsamen Code, Funktionen, Ressourcen und Konfigurationen in separaten AWS SAM Vorlagen wiederzuverwenden, sodass Sie nur Code aus einer einzigen Quelle verwalten müssen. Dadurch werden doppelter Code und Konfigurationen reduziert. Darüber hinaus optimiert dieser modulare Ansatz die Entwicklung, verbessert die Codeorganisation und erleichtert die Konsistenz serverloser Anwendungen. Mit verschachtelten Anwendungen können Sie sich mehr auf die Geschäftslogik konzentrieren, die für Ihre Anwendung einzigartig ist.

Verwenden Sie den Ressourcentyp, um eine verschachtelte Anwendung in Ihrer serverlosen Anwendung zu definieren. AWS::Serverless::Application

Sie können verschachtelte Anwendungen aus den folgenden zwei Quellen definieren:

- Eine AWS Serverless Application Repository Anwendung — Sie können verschachtelte Anwendungen definieren, indem Sie Anwendungen verwenden, die für Ihr Konto in verfügbar sind. AWS Serverless Application Repository Dies können private Anwendungen in Ihrem Konto sein, Anwendungen, die privat mit Ihrem Konto geteilt werden, oder Anwendungen, die öffentlich in der AWS Serverless Application Repository geteilt werden. Weitere Informationen zu den verschiedenen Stufen der Bereitstellungsberechtigungen finden Sie unter Berechtigungen zur Anwendungsbereitstellung und Veröffentlichung von Anwendungen im AWS Serverless Application Repository Entwicklerhandbuch.
- Eine lokale Anwendung — Sie können verschachtelte Anwendungen definieren, indem Sie Anwendungen verwenden, die in Ihrem lokalen Dateisystem gespeichert sind.

In den folgenden Abschnitten finden Sie Einzelheiten AWS SAM zur Definition dieser beiden Arten von verschachtelten Anwendungen in Ihrer serverlosen Anwendung.

> **Note**
>
> Die maximale Anzahl von Anwendungen, die in einer serverlosen Anwendung verschachtelt werden können, beträgt 200.
> Die maximale Anzahl von Parametern, die eine verschachtelte Anwendung haben kann, beträgt 60.

Definition einer verschachtelten Anwendung aus dem AWS Serverless Application Repository

Sie können verschachtelte Anwendungen definieren, indem Sie Anwendungen verwenden, die in der verfügbar sind. AWS Serverless Application Repository Sie können Anwendungen, die verschachtelte Anwendungen enthalten, auch mithilfe von speichern und verteilen. AWS Serverless Application Repository Um die Details einer verschachtelten Anwendung in der zu überprüfen AWS Serverless Application Repository, können Sie das AWS SDK, die oder die AWS CLI Lambda-Konsole verwenden.

Um eine Anwendung zu definieren, die AWS Serverless Application Repository in der AWS SAM Vorlage Ihrer serverlosen Anwendung gehostet wird, klicken Sie auf der Detailseite jeder Anwendung auf die Schaltfläche Als SAM-Ressource kopieren. AWS Serverless Application Repository Führen Sie dazu die folgenden Schritte aus:

1. Stellen Sie sicher, dass Sie bei der angemeldet sind. AWS-Managementkonsole
2. Suchen Sie die Anwendung, in der Sie sich verschachteln möchten, AWS Serverless Application Repository indem Sie die Schritte im Abschnitt „Anwendungen durchsuchen, suchen und bereitstellen“ des AWS Serverless Application Repository Entwicklerhandbuchs ausführen.
3. Wählen Sie die Schaltfläche Als SAM-Ressource kopieren. Der SAM-Vorlagenbereich für die Anwendung, die Sie gerade ansehen, befindet sich jetzt in Ihrer Zwischenablage.
4. Fügen Sie den Abschnitt mit der SAM-Vorlage in den `Resources:` Abschnitt der SAM-Vorlagendatei für die Anwendung ein, die Sie in dieser Anwendung verschachteln möchten.

Im Folgenden finden Sie ein Beispiel für einen SAM-Vorlagenabschnitt für eine verschachtelte Anwendung, die AWS Serverless Application Repository in folgender Datei gehostet wird:

```
Transform: AWS::Serverless-2016-10-31

Resources:
  applicationaliasname:
    Type: AWS::Serverless::Application
    Properties:
      Location:
        ApplicationId: arn:aws:serverlessrepo:us-
east-1:123456789012:applications/application-alias-name
        SemanticVersion: 1.0.0
      Parameters:
        # Optional parameter that can have default value overridden
        # ParameterName1: 15 # Uncomment to override default value
        # Required parameter that needs value to be provided
        ParameterName2: YOUR_VALUE
```

Wenn keine Parametereinstellungen erforderlich sind, können Sie den `Parameters:` Abschnitt der Vorlage weglassen.

> **Important**
>
> Anwendungen, die verschachtelte Anwendungen enthalten, die in der gehostet werden, AWS Serverless Application Repository übernehmen die Freigabebeschränkungen der verschachtelten Anwendungen.
> Nehmen wir beispielsweise an, eine Anwendung ist öffentlich freigegeben, sie enthält jedoch eine verschachtelte Anwendung, die nur privat mit dem AWS Konto geteilt wird, mit dem die übergeordnete Anwendung erstellt wurde. In diesem Fall können Sie die übergeordnete Anwendung nicht bereitstellen, wenn Ihr AWS Konto nicht berechtigt ist, die verschachtelte Anwendung bereitzustellen. Weitere Informationen zu Berechtigungen für die Bereitstellung von Anwendungen finden Sie unter Berechtigungen zur Anwendungsbereitstellung und Veröffentlichung von Anwendungen im AWS Serverless Application Repository Entwicklerhandbuch.

Definieren einer verschachtelten Anwendung aus dem lokalen Dateisystem

Sie können verschachtelte Anwendungen definieren, indem Sie Anwendungen verwenden, die in Ihrem lokalen Dateisystem gespeichert sind. Dazu geben Sie den Pfad zur AWS SAM Vorlagendatei an, die in Ihrem lokalen Dateisystem gespeichert ist.

Im Folgenden finden Sie ein Beispiel für einen SAM-Vorlagenabschnitt für eine verschachtelte lokale Anwendung:

```
Transform: AWS::Serverless-2016-10-31

Resources:
  applicationaliasname:
    Type: AWS::Serverless::Application
    Properties:
      Location: ../my-other-app/template.yaml
      Parameters:
        # Optional parameter that can have default value overridden
        # ParameterName1: 15 # Uncomment to override default value
        # Required parameter that needs value to be provided
        ParameterName2: YOUR_VALUE
```

Wenn es keine Parametereinstellungen gibt, können Sie den `Parameters:` Abschnitt der Vorlage weglassen.

Bereitstellen verschachtelter Anwendungen

Sie können Ihre verschachtelte Anwendung mithilfe des AWS SAMCLI Befehls bereitstellen. `sam deploy` Weitere Details finden Sie unter Stellen Sie Ihre Anwendung und Ressourcen bereit mit AWS SAM.

> **Note**
>
> Wenn Sie eine Anwendung bereitstellen, die verschachtelte Anwendungen enthält, müssen Sie bestätigen, dass sie verschachtelte Anwendungen enthält. Sie tun dies, indem Sie `CAPABILITY_AUTO_EXPAND` es an die CreateCloudFormationChangeSet API übergeben oder den `aws serverlessrepo create-cloud-formation-change-set` AWS CLI Befehl verwenden.
> Weitere Informationen zur Bestätigung verschachtelter Anwendungen finden Sie unter Bestätigung von IAM-Rollen, Ressourcenrichtlinien und verschachtelten Anwendungen bei der Bereitstellung von Anwendungen im Entwicklerhandbuch.AWS Serverless Application Repository

Verwalten Sie zeitbasierte Ereignisse mit dem EventBridge Scheduler in AWS SAM

Der Inhalt dieses Themas enthält Informationen darüber, was Amazon EventBridge Scheduler ist, was Support AWS SAM bietet, wie Sie Scheduler-Ereignisse erstellen können, und Beispiele, auf die Sie beim Erstellen von Scheduler-Ereignissen zurückgreifen können.

Was ist Amazon EventBridge Scheduler?

Verwenden Sie EventBridge Scheduler, um Ereignisse in Ihren AWS SAM Vorlagen zu planen. Amazon EventBridge Scheduler ist ein Planungsservice, mit dem Sie zig Millionen Ereignisse und Aufgaben für alle AWS Services erstellen, initiieren und verwalten können. Dieser Service ist besonders nützlich für zeitbezogene Ereignisse. Sie können ihn verwenden, um Ereignisse und wiederkehrende zeitbasierte Aufrufe zu planen. Es unterstützt auch einmalige Ereignisse sowie Rate- und Chron-Ausdrücke mit Start- und Endzeit.

Weitere Informationen zu Amazon EventBridge Scheduler finden Sie unter Was ist Amazon EventBridge Scheduler? im EventBridge Scheduler-Benutzerhandbuch.

Topics

- EventBridge Scheduler-Unterstützung in AWS SAM
- EventBridge Scheduler-Ereignisse erstellen in AWS SAM
- Beispiele
- Weitere Informationen

EventBridge Scheduler-Unterstützung in AWS SAM

Die Vorlagenspezifikation AWS Serverless Application Model (AWS SAM) bietet eine einfache Kurzsyntax, die Sie verwenden können, um Ereignisse mit dem EventBridge Scheduler für und zu planen. AWS Lambda AWS Step Functions

EventBridge Scheduler-Ereignisse erstellen in AWS SAM

Legen Sie die `ScheduleV2` Eigenschaft in Ihrer AWS SAM Vorlage als Ereignistyp fest, um Ihr EventBridge Scheduler-Ereignis zu definieren. Diese Eigenschaft unterstützt die `AWS::Serverless::StateMachine` Ressourcentypen `AWS::Serverless::Function` und.

```
MyFunction:
```

```
  Type: AWS::Serverless::Function
  Properties:
    Events:
      CWSchedule:
        Type: ScheduleV2
        Properties:
          ScheduleExpression: 'rate(1 minute)'
          Name: TestScheduleV2Function
          Description: Test schedule event

MyStateMachine:
  Type: AWS::Serverless::StateMachine
  Properties:
    Events:
      CWSchedule:
        Type: ScheduleV2
        Properties:
          ScheduleExpression: 'rate(1 minute)'
          Name: TestScheduleV2StateMachine
          Description: Test schedule event
```

EventBridge Die Scheduler-Ereignisplanung unterstützt auch Dead-Letter-Warteschlangen (DLQ) für unverarbeitete Ereignisse. Weitere Informationen zu Warteschlangen mit uneingeschränkten Briefen finden Sie im Scheduler-Benutzerhandbuch unter Konfiguration einer Warteschlange für unzustellbare Nachrichten für Scheduler. EventBridge EventBridge

Wenn ein DLQ-ARN angegeben ist, werden die Berechtigungen für den Scheduler-Zeitplan zum Senden von Nachrichten an den DLQ AWS SAM konfiguriert. Wenn kein DLQ-ARN angegeben ist, AWS SAM wird die DLQ-Ressource erstellt.

Beispiele

Einfaches Beispiel für die Definition eines EventBridge Scheduler-Ereignisses mit AWS SAM

```
Transform: AWS::Serverless-2016-10-31
Resources:
  MyLambdaFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.handler
      Runtime: python3.8
      InlineCode: |
        def handler(event, context):
```

```
            print(event)
            return {'body': 'Hello World!', 'statusCode': 200}
      MemorySize: 128
      Events:
        Schedule:
          Type: ScheduleV2
          Properties:
            ScheduleExpression: rate(1 minute)
            Input: '{"hello": "simple"}'

  MySFNFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.handler
      Runtime: python3.8
      InlineCode: |
        def handler(event, context):
            print(event)
            return {'body': 'Hello World!', 'statusCode': 200}
      MemorySize: 128

  StateMachine:
    Type: AWS::Serverless::StateMachine
    Properties:
      Type: STANDARD
      Definition:
        StartAt: MyLambdaState
        States:
          MyLambdaState:
            Type: Task
            Resource: !GetAtt MySFNFunction.Arn
            End: true
      Policies:
        - LambdaInvokePolicy:
            FunctionName: !Ref MySFNFunction
      Events:
        Schedule:
          Type: ScheduleV2
          Properties:
            ScheduleExpression: rate(1 minute)
            Input: '{"hello": "simple"}'
```

Weitere Informationen

Weitere Informationen zur Definition der `ScheduleV2` EventBridge Scheduler-Eigenschaft finden Sie unter:

- ScheduleV2für`AWS::Serverless::Function`.
- ScheduleV2für`AWS::Serverless::StateMachine`.

Orchestrierung von AWS SAM Ressourcen mit AWS Step Functions

Sie können AWS Step Functionses verwenden, um AWS Lambda Funktionen und andere AWS Ressourcen zu orchestrieren, um komplexe und robuste Workflows zu erstellen. Step Functions, um Ihrer Anwendung mitzuteilen, wann und unter welchen Bedingungen Ihre AWS Ressourcen, wie AWS Lambda Funktionen, verwendet werden. Dies vereinfacht den Prozess der Erstellung komplexer und robuster Workflows. Mit AWS::Serverless::StateMachine dieser Methode definieren Sie die einzelnen Schritte in Ihrem Workflow, ordnen jedem Schritt Ressourcen zu und ordnen diese Schritte dann zusammen. Außerdem fügen Sie dort, wo sie benötigt werden, Übergänge und Bedingungen hinzu. Dies vereinfacht den Prozess der Erstellung eines komplexen und robusten Workflows.

> **Note**
>
> Um AWS SAM Vorlagen zu verwalten, die Step Functions Functions-Zustandsmaschinen enthalten, müssen Sie Version 0.52.0 oder höher von verwenden. AWS SAMCLI Um zu überprüfen, welche Version Sie haben, führen Sie den Befehl `sam --version` aus.

Step Functions basiert auf den Konzepten von Aufgaben und Zustandsmaschinen. Sie definieren Zustandsmaschinen mithilfe der JSON-basierten Amazon States-Sprache. Die Step Functions Functions-Konsole zeigt eine grafische Ansicht der Struktur Ihrer Zustandsmaschine an, sodass Sie die Logik Ihrer Zustandsmaschine visuell überprüfen und Ausführungen überwachen können.

Mit der Unterstützung von Step Functions in AWS Serverless Application Model (AWS SAM) können Sie Folgendes tun:

- Definieren Sie Zustandsmaschinen, entweder direkt in einer AWS SAM Vorlage oder in einer separaten Datei
- Erstellen Sie Rollen für die Ausführung von Zustandsmaschinen mithilfe von AWS SAM Richtlinienvorlagen, Inline-Richtlinien oder verwalteten Richtlinien

- Triggern Sie State-Machine-Ausführungen mit API Gateway- oder EventBridge Amazon-Ereignissen, nach einem Zeitplan innerhalb einer AWS SAM Vorlage oder durch direkten Aufruf APIs
- Verwenden Sie verfügbare AWS SAM Richtlinienvorlagen für allgemeine Step Functions Functions-Entwicklungsmuster.

Beispiel

Der folgende Beispielausschnitt aus einer AWS SAM Vorlagendatei definiert eine Step Functions Functions-Zustandsmaschine in einer Definitionsdatei. Beachten Sie, dass die `my_state_machine.asl.json` Datei in der Sprache der Amazon-Staaten verfasst sein muss.

```
AWSTemplateFormatVersion: "2010-09-09"
Transform: AWS::Serverless-2016-10-31
Description: Sample SAM template with Step Functions State Machine

Resources:
  MyStateMachine:
    Type: AWS::Serverless::StateMachine
    Properties:
      DefinitionUri: statemachine/my_state_machine.asl.json
      ...
```

Informationen zum Herunterladen einer AWS SAM Beispielanwendung, die eine Step Functions-Zustandsmaschine enthält, finden Sie AWS SAM im AWS Step Functions Entwicklerhandbuch unter Create a Step Functions State Machine Using.

Weitere Informationen

Weitere Informationen zu Step Functions und deren Verwendung mit AWS SAM finden Sie im Folgenden:

- Funktionsweise von AWS Step Functions
- AWS Step Functions und AWS Serverless Application Model
- Tutorial: Erstellen Sie eine Step Functions Functions-Zustandsmaschine mit AWS SAM
- AWS SAM Spezifikation: AWS::Serverless::StateMachine

Richten Sie die Codesignatur für Ihre AWS SAM Anwendung ein

Um sicherzustellen, dass nur vertrauenswürdiger Code bereitgestellt wird, können Sie AWS SAM die Codesignatur für Ihre serverlosen Anwendungen aktivieren. Durch das Signieren Ihres Codes können Sie sicherstellen, dass der Code seit dem Signieren nicht geändert wurde und dass nur signierte Codepakete von vertrauenswürdigen Herausgebern in Ihren Lambda-Funktionen ausgeführt werden. Dadurch werden Unternehmen von der Last befreit, Gatekeeper-Komponenten in ihren Bereitstellungspipelines zu erstellen.

Weitere Informationen zur Codesignatur finden Sie unter Configuring Code Signing for Lambda-Funktionen im AWS Lambda Developer Guide.

Bevor Sie die Codesignatur für Ihre serverlose Anwendung konfigurieren können, müssen Sie mit Signer ein Signaturprofil erstellen AWS . Sie verwenden dieses Signaturprofil für die folgenden Aufgaben:

1. Codesignaturkonfiguration erstellen — Deklarieren Sie eine `AWS::Lambda::CodeSigningConfig`Ressource, um die Signaturprofile vertrauenswürdiger Herausgeber anzugeben und die Richtlinienaktion für Validierungsprüfungen festzulegen. Sie können dieses Objekt in derselben AWS SAM Vorlage wie Ihre serverlose Funktion, in einer anderen AWS SAM Vorlage oder in einer CloudFormation Vorlage deklarieren. Anschließend aktivieren Sie die Codesignatur für eine serverlose Funktion, indem Sie der `CodeSigningConfigArn`Eigenschaft die Funktion mit dem Amazon-Ressourcennamen (ARN) einer `AWS::Lambda::CodeSigningConfig`Ressource angeben.
2. Signieren Sie Ihren Code — Verwenden Sie den `sam deploy`Befehl `sam package`oder mit der `--signing-profiles` Option.

> **Note**
>
> Um Ihren Code erfolgreich mit den `sam deploy` Befehlen `sam package` oder zu signieren, muss die Versionierung für den Amazon S3 S3-Bucket aktiviert sein, den Sie mit diesen Befehlen verwenden. Wenn Sie den Amazon S3 S3-Bucket verwenden, der für Sie AWS SAM erstellt, wird die Versionierung automatisch aktiviert. Weitere Informationen zur Amazon S3 S3-Bucket-Versionierung und Anweisungen zur Aktivierung der Versionierung in einem von Ihnen bereitgestellten Amazon S3 S3-Bucket finden Sie unter Verwenden der Versionierung in Amazon S3 S3-Buckets im Amazon Simple Storage Service-Benutzerhandbuch.

Wenn Sie eine serverlose Anwendung bereitstellen, führt Lambda Validierungsprüfungen für alle Funktionen durch, für die Sie die Codesignatur aktiviert haben. Lambda führt auch Validierungsprüfungen auf allen Ebenen durch, von denen diese Funktionen abhängen. Weitere Informationen zu den Validierungsprüfungen von Lambda finden Sie unter Signaturvalidierung im AWS Lambda Entwicklerhandbuch.

Beispiel

Ein Signaturprofil erstellen

Führen Sie den folgenden Befehl aus, um ein Signaturprofil zu erstellen:

```
aws signer put-signing-profile --platform-id "AWSLambda-SHA384-ECDSA" --profile-
name MySigningProfile
```

Wenn der vorherige Befehl erfolgreich war, wird der ARN des Signaturprofils zurückgegeben. Beispiel:

```
{
    "arn": "arn:aws:signer:us-east-1:111122223333:/signing-profiles/MySigningProfile",
    "profileVersion": "SAMPLEverx",
    "profileVersionArn": "arn:aws:signer:us-east-1:111122223333:/signing-
profiles/MySigningProfile/SAMPLEverx"
}
```

Das `profileVersionArn` Feld enthält den ARN, der verwendet werden soll, wenn Sie die Codesignaturkonfiguration erstellen.

Eine Codesignaturkonfiguration erstellen und die Codesignatur für eine Funktion aktivieren

Die folgende AWS SAM Beispielvorlage deklariert eine `AWS::Lambda::CodeSigningConfig`Ressource und aktiviert die Codesignatur für eine Lambda-Funktion. In diesem Beispiel gibt es ein vertrauenswürdiges Profil, und Bereitstellungen werden abgelehnt, wenn die Signaturprüfungen fehlschlagen.

```
Resources:
  HelloWorld:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: hello_world/
      Handler: app.lambda_handler
```

```
      Runtime: python3.7
      CodeSigningConfigArn: !Ref MySignedFunctionCodeSigningConfig

  MySignedFunctionCodeSigningConfig:
    Type: AWS::Lambda::CodeSigningConfig
    Properties:
      Description: "Code Signing for MySignedLambdaFunction"
      AllowedPublishers:
        SigningProfileVersionArns:
          - MySigningProfile-profileVersionArn
      CodeSigningPolicies:
        UntrustedArtifactOnDeployment: "Enforce"
```

Signieren Sie Ihren Code

Sie können Ihren Code signieren, wenn Sie Ihre Anwendung verpacken oder bereitstellen. Geben Sie die `--signing-profiles` Option entweder mit dem `sam deploy` Befehl `sam package` oder an, wie in den folgenden Beispielbefehlen gezeigt.

Signieren Sie Ihren Funktionscode beim Paketieren Ihrer Anwendung:

```
sam package --signing-profiles HelloWorld=MySigningProfile --s3-bucket amzn-s3-demo-
bucket --output-template-file packaged.yaml
```

Signieren Sie beim Verpacken Ihrer Anwendung sowohl Ihren Funktionscode als auch eine Ebene, von der Ihre Funktion abhängt:

```
sam package --signing-profiles HelloWorld=MySigningProfile MyLayer=MySigningProfile --
s3-bucket amzn-s3-demo-bucket --output-template-file packaged.yaml
```

Signieren Sie Ihren Funktionscode und eine Ebene und führen Sie dann eine Bereitstellung durch:

```
sam deploy --signing-profiles HelloWorld=MySigningProfile MyLayer=MySigningProfile --
s3-bucket amzn-s3-demo-bucket --template-file packaged.yaml --stack-name --region us-
east-1 --capabilities CAPABILITY_IAM
```

> **Note**
>
> Um Ihren Code erfolgreich mit den `sam deploy` Befehlen `sam package` oder zu signieren, muss die Versionierung für den Amazon S3 S3-Bucket aktiviert sein, den Sie mit diesen

Befehlen verwenden. Wenn Sie den Amazon S3 S3-Bucket verwenden, der für Sie AWS SAM erstellt, wird die Versionierung automatisch aktiviert. Weitere Informationen zur Amazon S3 S3-Bucket-Versionierung und Anweisungen zur Aktivierung der Versionierung in einem von Ihnen bereitgestellten Amazon S3 S3-Bucket finden Sie unter Verwenden der Versionierung in Amazon S3 S3-Buckets im Amazon Simple Storage Service-Benutzerhandbuch.

Bereitstellung von Signaturprofilen mit `sam deploy --guided`

Wenn Sie den `sam deploy --guided` Befehl mit einer serverlosen Anwendung ausführen, die mit Codesignatur konfiguriert ist, werden Sie AWS SAM aufgefordert, das Signaturprofil anzugeben, das für die Codesignatur verwendet werden soll. Weitere Informationen zu `sam deploy --guided` Eingabeaufforderungen finden Sie sam deploy in der AWS SAMCLI Befehlsreferenz.

AWS SAM Vorlagendateien validieren

Validieren Sie Ihre Vorlagen mit`sam validate`. Derzeit überprüft dieser Befehl, ob es sich bei der bereitgestellten Vorlage um eine gültige JSON/YAML-Vorlage handelt. Wie bei den meisten AWS SAMCLI Befehlen sucht er standardmäßig nach einer `template.[yaml|yml]` Datei in Ihrem aktuellen Arbeitsverzeichnis. file/location Mit der `--template` Option `-t` oder können Sie eine andere Vorlage angeben.

Beispiel:

```
$ sam validate
<path-to-template>/template.yaml is a valid SAM Template
```

> **Note**
>
> Für den `sam validate` Befehl müssen AWS Anmeldeinformationen konfiguriert werden. Weitere Informationen finden Sie unter Konfiguration der AWS SAMCLI.

Erstellen Sie Ihre Anwendung mit AWS SAM

Nachdem Sie Ihrer AWS SAM Vorlage Ihre Infrastruktur als Code (IaC) hinzugefügt haben, können Sie mit dem Erstellen Ihrer Anwendung mithilfe des sam build Befehls beginnen. Dieser Befehl

erstellt Build-Artefakte aus den Dateien in Ihrem Anwendungsprojektverzeichnis (d. h. Ihrer AWS SAM Vorlagendatei, dem Anwendungscode und allen anwendbaren sprachspezifischen Dateien und Abhängigkeiten). Diese Build-Artefakte bereiten Ihre serverlose Anwendung auf spätere Schritte der Anwendungsentwicklung vor, z. B. lokales Testen und Bereitstellen in der Cloud. AWS Sowohl beim Testen als auch bei der Bereitstellung werden Build-Artefakte als Eingaben verwendet.

Sie können sam build damit Ihre gesamte serverlose Anwendung erstellen. Darüber hinaus können Sie benutzerdefinierte Builds erstellen, z. B. solche mit bestimmten Funktionen, Ebenen oder benutzerdefinierten Laufzeiten. Weitere Informationen darüber, wie und warum Sie sie verwendensam build, finden Sie in den Themen in diesem Abschnitt. Eine Einführung in die Verwendung des `sam build` Befehls finden Sie unterEinführung in das Bauen mit AWS SAM.

Themen

- Einführung in das Bauen mit AWS SAM
- Standard-Build mit AWS SAM
- Passen Sie Builds an mit AWS SAM

Einführung in das Bauen mit AWS SAM

Verwenden Sie den AWS Serverless Application Model Befehl Command Line Interface (AWS SAMCLI)`sam build`, um Ihre serverlose Anwendung auf nachfolgende Schritte in Ihrem Entwicklungsworkflow vorzubereiten, z. B. lokale Tests oder die Bereitstellung auf der. AWS Cloud Mit diesem Befehl wird ein `.aws-sam` Verzeichnis erstellt, das Ihre Anwendung in einem Format und an einem Speicherort strukturiert, die `sam local` `sam deploy` erforderlich sind.

- Eine Einführung in das finden AWS SAMCLI Sie unterWas ist der AWS SAM CLI?.
- Eine Liste der `sam build` Befehlsoptionen finden Sie untergleicher Build.
- Ein Beispiel für die Verwendung `sam build` während eines typischen Entwicklungsworkflows finden Sie unterSchritt 2: Erstellen Sie Ihre Anwendung.

> **Note**
>
> `sam build`Für die Verwendung müssen Sie mit den grundlegenden Komponenten einer serverlosen Anwendung auf Ihrem Entwicklungscomputer beginnen. Dazu gehören eine AWS SAM Vorlage, AWS Lambda Funktionscode und alle sprachspezifischen Dateien und

Abhängigkeiten. Weitere Informationen hierzu finden Sie unter Erstellen Sie Ihre Bewerbung in AWS SAM.

Themen

- Anwendungen mit Sam Build erstellen
- Lokales Testen und Bereitstellen
- Best Practices
- Optionen für Sam Build
- Fehlerbehebung
- Beispiele
- Weitere Informationen

Anwendungen mit Sam Build erstellen

Ziehen Sie vor der Verwendung in Betracht`sam build`, Folgendes zu konfigurieren:

1. Lambda-Funktionen und -Layer — Der `sam build` Befehl kann Lambda-Funktionen und -Layer erstellen. Weitere Informationen zu Lambda-Schichten finden Sie unterAufbau von Lambda-Schichten in AWS SAM.
2. Lambda-Laufzeit — Die Laufzeit bietet eine sprachspezifische Umgebung, in der Ihre Funktion in einer Ausführungsumgebung ausgeführt wird, wenn sie aufgerufen wird. Sie können native und benutzerdefinierte Laufzeiten konfigurieren.
   a. Native Laufzeit — Verfassen Sie Ihre Lambda-Funktionen in einer unterstützten Lambda-Laufzeit und erstellen Sie Funktionen für die Verwendung einer nativen Lambda-Laufzeit in der. AWS Cloud
   b. Benutzerdefinierte Laufzeit — Verfassen Sie Ihre Lambda-Funktionen mit einer beliebigen Programmiersprache und erstellen Sie Ihre Laufzeit mit einem benutzerdefinierten Prozess, der in einem Builder makefile oder einem Drittanbieter definiert ist, z. B. esbuild Weitere Informationen hierzu finden Sie unter Erstellen von Lambda-Funktionen mit benutzerdefinierten Laufzeiten in AWS SAM.
3. Lambda-Pakettyp — Lambda-Funktionen können in den folgenden Lambda-Bereitstellungspakettypen verpackt werden:
   a. ZIP-Dateiarchiv — Enthält Ihren Anwendungscode und seine Abhängigkeiten.

b. Container-Image — Enthält das Basisbetriebssystem, die Laufzeit, Lambda-Erweiterungen, Ihren Anwendungscode und dessen Abhängigkeiten.

Diese Anwendungseinstellungen können konfiguriert werden, wenn eine Anwendung initialisiert wird mit. `sam init`

- Weitere Informationen zur Verwendung von finden Sie `sam init` unterErstellen Sie Ihre Bewerbung in AWS SAM.
- Weitere Informationen zur Konfiguration dieser Einstellungen in Ihrer Anwendung finden Sie unterStandard-Build mit AWS SAM.

So erstellen Sie eine Anwendung

1. `cd`zur Wurzel Ihres Projekts. Dies ist derselbe Speicherort wie Ihre AWS SAM Vorlage.

```
$ cd sam-app
```

2. Führen Sie Folgendes aus:

```
sam-app $ sam build <arguments> <options>
```

> **Note**
>
> Eine häufig verwendete Option ist`--use-container`. Weitere Informationen hierzu finden Sie unter Eine Lambda-Funktion innerhalb eines bereitgestellten Containers erstellen.

Das Folgende ist ein Beispiel für die AWS SAMCLI Ausgabe:

```
sam-app $ sam build
Starting Build use cache
Manifest file is changed (new hash: 3298f1304...d4d421) or dependency folder (.aws-
sam/deps/4d3dfad6-a267-47a6-a6cd-e07d6fae318c) is missing for (HelloWorldFunction),
 downloading dependencies and copying/building source
Building codeuri: /Users/.../sam-app/hello_world runtime: python3.12 metadata: {}
 architecture: x86_64 functions: HelloWorldFunction
Running PythonPipBuilder:CleanUp
```

```
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource
Running PythonPipBuilder:CopySource

Build Succeeded

Built Artifacts  : .aws-sam/build
Built Template   : .aws-sam/build/template.yaml

Commands you can use next
=========================
[*] Validate SAM template: sam validate
[*] Invoke Function: sam local invoke
[*] Test Function in the Cloud: sam sync --stack-name {{stack-name}} --watch
[*] Deploy: sam deploy --guided
```

3. Das AWS SAMCLI erstellt ein `.aws-sam` Build-Verzeichnis. Im Folgenden wird ein Beispiel gezeigt:

```
.aws-sam
### build
#   ### HelloWorldFunction
#   #   ### __init__.py
#   #   ### app.py
#   #   ### requirements.txt
#   ### template.yaml
### build.toml
```

Je nachdem, wie Ihre Anwendung konfiguriert ist, AWS SAMCLI macht der Folgendes:

1. Lädt Abhängigkeiten im `.aws-sam/build` Verzeichnis herunter, installiert und organisiert sie.
2. Bereitet Ihren Lambda-Code vor. Dies kann das Kompilieren Ihres Codes, das Erstellen ausführbarer Binärdateien und das Erstellen von Container-Images umfassen.
3. Kopiert Build-Artefakte in das `.aws-sam` Verzeichnis. Das Format hängt vom Typ Ihres Anwendungspakets ab.
   a. Bei Pakettypen im Format .zip sind die Artefakte noch nicht gezippt, sodass sie für lokale Tests verwendet werden können. Das AWS SAMCLI komprimiert Ihre Anwendung, wenn Sie sie verwenden. `sam deploy`

   b. Für Container-Image-Pakettypen wird ein Container-Image lokal erstellt und in der `.aws-sam/build.toml` Datei referenziert.
4. Kopiert die AWS SAM Vorlage in das `.aws-sam` Verzeichnis und ändert sie bei Bedarf mit neuen Dateipfaden.

Im Folgenden sind die Hauptkomponenten aufgeführt, aus denen Ihre Build-Artefakte im `.aws-sam` Verzeichnis bestehen:

- Das Build-Verzeichnis — Enthält Ihre Lambda-Funktionen und -Layer, die unabhängig voneinander strukturiert sind. Dies führt zu einer eindeutigen Struktur für jede Funktion oder Ebene im `.aws-sam/build` Verzeichnis.
- Die AWS SAM Vorlage — Modifiziert mit aktualisierten Werten, die auf Änderungen während des Erstellungsprozesses basieren.
- Die Datei build.toml — Eine Konfigurationsdatei, die Build-Einstellungen enthält, die von der verwendet werden. AWS SAMCLI

Lokales Testen und Bereitstellen

Bei der Durchführung lokaler Tests mit `sam local` oder bei der Bereitstellung mit `sam deploy` AWS SAMCLI führt der wie folgt aus:

1. Zunächst wird geprüft, ob ein `.aws-sam` Verzeichnis existiert und ob sich eine AWS SAM Vorlage in diesem Verzeichnis befindet. Wenn diese Bedingungen erfüllt sind, AWS SAMCLI betrachtet der dieses Verzeichnis als das Stammverzeichnis Ihrer Anwendung.
2. Wenn diese Bedingungen nicht erfüllt sind, AWS SAMCLI betrachtet der den ursprünglichen Speicherort Ihrer AWS SAM Vorlage als Stammverzeichnis Ihrer Anwendung.

Wenn bei der Entwicklung Änderungen an Ihren ursprünglichen Anwendungsdateien vorgenommen werden, führen `sam build` Sie zunächst die Aktualisierung des `.aws-sam` Verzeichnisses durch, bevor Sie lokal testen.

Best Practices

- Bearbeiten Sie keinen Code im `.aws-sam/build` Verzeichnis. Aktualisieren Sie stattdessen Ihren ursprünglichen Quellcode in Ihrem Projektordner und führen Sie den Befehl aus`sam build`, um das `.aws-sam/build` Verzeichnis zu aktualisieren.

- Wenn Sie Ihre Originaldateien ändern, führen Sie den Befehl aus, `sam build` um das `.aws-sam/build` Verzeichnis zu aktualisieren.
- Möglicherweise möchten Sie AWS SAMCLI, dass der auf das ursprüngliche Stammverzeichnis Ihres Projekts statt auf das `.aws-sam` Verzeichnis verweist, z. B. beim Entwickeln und Testen mit`sam local`. Löschen Sie das `.aws-sam` Verzeichnis oder die AWS SAM Vorlage im `.aws-sam` Verzeichnis, damit Ihr ursprüngliches Projektverzeichnis als Stammprojektverzeichnis AWS SAMCLI erkannt wird. Wenn Sie bereit sind, führen Sie den `sam build` Vorgang erneut aus, um das `.aws-sam` Verzeichnis zu erstellen.
- Bei der Ausführung `sam build` wird das `.aws-sam/build` Verzeichnis jedes Mal überschrieben. Das `.aws-sam` Verzeichnis tut dies nicht. Wenn Sie Dateien wie Protokolle speichern möchten, speichern Sie sie in, `.aws-sam` damit sie nicht überschrieben werden.

Optionen für Sam Build

Aufbau einer einzigen Ressource

Geben Sie die logische ID der Ressource an, um nur diese Ressource zu erstellen. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam build HelloWorldFunction
```

Um eine Ressource aus einer verschachtelten Anwendung oder einem geschachtelten Stack zu erstellen, geben Sie die logische ID der Anwendung oder des Stacks zusammen mit der logischen ID der Ressource im folgenden Format *<stack-logical-id>*/*<resource-logical-id>* an:

```
$ sam build MyNestedStack/MyFunction
```

Eine Lambda-Funktion innerhalb eines bereitgestellten Containers erstellen

Die `--use-container` Option lädt ein Container-Image herunter und verwendet es, um Ihre Lambda-Funktionen zu erstellen. Der lokale Container wird dann in Ihrer `.aws-sam/build.toml` Datei referenziert.

Diese Option Docker muss installiert sein. Detaillierte Anweisungen finden Sie unter Installieren von Docker.

Das Folgende ist ein Beispiel für diesen Befehl:

```
$ sam build --use-container
```

Sie können das Container-Image angeben, das mit der `--build-image` Option verwendet werden soll. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam build --use-container --build-image amazon/aws-sam-cli-build-image-nodejs20.x
```

Um das Container-Image anzugeben, das für eine einzelne Funktion verwendet werden soll, geben Sie die logische ID der Funktion an. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam build --use-container --build-image Function1=amazon/aws-sam-cli-build-image-
python3.12
```

Container-Images erstellen mit BuildKit

Wenn Sie Lambda-Funktionen mit dem `Image` Pakettyp erstellen, können Sie ihn BuildKit als Build-Engine verwenden, indem Sie die `--use-buildkit` Option übergeben. BuildKitbietet im Vergleich zur älteren Build-Engine eine verbesserte Build-Leistung, besseres Caching und eine verbesserte Docker Build-Ausgabe.

Für diese Option muss entweder das Docker oder Finch CLIs installiert sein und in IhremPATH.

Sie können es `--use-buildkit` mit anderen Build-Optionen kombinieren. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam build --use-buildkit --parallel
```

> **Note**
>
> Das `--use-buildkit` Flag gilt nur für Container-Image-Builds. Es hat keine Auswirkung auf ZIP-Funktionen, mit denen gebaut wird`--use-container`.

Übergeben Sie Umgebungsvariablen an den Build-Container

Verwenden Sie die`--container-env-var`, um Umgebungsvariablen an den Build-Container zu übergeben. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam build --use-container --container-env-var Function1.GITHUB_TOKEN=<token1> --
container-env-var GLOBAL_ENV_VAR=<global-token>
```

Verwenden Sie die `--container-env-var-file` Option, um Umgebungsvariablen aus einer Datei zu übergeben. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam build --use-container --container-env-var-file <env.json>
```

Beispiel für die `env.json` Datei:

```
{
  "MyFunction1": {
    "GITHUB_TOKEN": "TOKEN1"
  },
  "MyFunction2": {
    "GITHUB_TOKEN": "TOKEN2"
  }
}
```

Beschleunigen Sie die Erstellung von Anwendungen, die mehrere Funktionen enthalten

Wenn Sie eine Anwendung mit mehreren Funktionen ausführen`sam build`, AWS SAMCLI erstellt sie jede Funktion einzeln. Verwenden Sie die `--parallel` Option, um den Erstellungsprozess zu beschleunigen. Dadurch werden alle Ihre Funktionen und Ebenen gleichzeitig erstellt.

Im Folgenden finden Sie ein Beispiel für diesen Befehl:

```
$ sam build –-parallel
```

Beschleunigen Sie die Build-Zeiten, indem Sie Ihr Projekt im Quellordner erstellen

Für unterstützte Laufzeiten und Build-Methoden können Sie die `--build-in-source` Option verwenden, Ihr Projekt direkt im Quellordner zu erstellen. Standardmäßig werden die AWS SAM CLI Builds in einem temporären Verzeichnis gespeichert, was das Kopieren von Quellcode und Projektdateien beinhaltet. Damit befinden `--build-in-source` sich die AWS SAM CLI Builds direkt in Ihrem Quellordner, was den Build-Prozess beschleunigt, da keine Dateien mehr in ein temporäres Verzeichnis kopiert werden müssen.

Eine Liste der unterstützten Laufzeiten und Build-Methoden finden Sie unter`--build-in-source`.

Fehlerbehebung

Informationen zur Behebung von Problemen finden Sie unterAWS SAMCLIProblembehandlung. AWS SAMCLI

Beispiele

Erstellen einer Anwendung, die eine native Runtime- und .zip-Pakettyp verwendet

Dieses Beispiel finden Sie unterTutorial: Stellen Sie eine Hello World-Anwendung bereit mit AWS SAM.

Eine Anwendung erstellen, die einen systemeigenen Laufzeit- und Image-Pakettyp verwendet

Zuerst starten wir, `sam init` um eine neue Anwendung zu initialisieren. Während des interaktiven Ablaufs wählen wir den `Image` Pakettyp aus. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam init
...
Which template source would you like to use?
        1 - AWS Quick Start Templates
        2 - Custom Template Location
Choice: 1

Choose an AWS Quick Start application template
        1 - Hello World Example
        2 - Multi-step workflow
        3 - Serverless API
        4 - Scheduled task
        5 - Standalone function
        6 - Data processing
        7 - Hello World Example With Powertools
        8 - Infrastructure event management
        9 - Serverless Connector Hello World Example
        10 - Multi-step workflow with Connectors
        11 - Lambda EFS example
        12 - DynamoDB Example
        13 - Machine Learning
Template: 1

Use the most popular runtime and package type? (Python and zip) [y/N]: ENTER

Which runtime would you like to use?
```

```
        ...
        10 - java8
        11 - nodejs20.x
        12 - nodejs18.x
        13 - nodejs16.x
        ...
Runtime: 12

What package type would you like to use?
        1 - Zip
        2 - Image
Package type: 2

Based on your selections, the only dependency manager available is npm.
We will proceed copying the template using npm.

Would you like to enable X-Ray tracing on the function(s) in your application?  [y/
N]: ENTER

Would you like to enable monitoring using CloudWatch Application Insights?
For more info, please view https://docs.aws.amazon.com/AmazonCloudWatch/latest/
monitoring/cloudwatch-application-insights.html [y/N]: ENTER

Project name [sam-app]: ENTER

Cloning from https://github.com/aws/aws-sam-cli-app-templates (process may take a
 moment)

    -----------------------
    Generating application:
    -----------------------
    Name: sam-app
    Base Image: amazon/nodejs18.x-base
    Architectures: x86_64
    Dependency Manager: npm
    Output Directory: .
    Configuration file: sam-app/samconfig.toml

    Next steps can be found in the README file at sam-app/README.md

...
```

Das AWS SAMCLI initialisiert eine Anwendung und erstellt das folgende Projektverzeichnis:

```
sam-app
### README.md
### events
#   ### event.json
### hello-world
#   ### Dockerfile
#   ### app.mjs
#   ### package.json
#   ### tests
#       ### unit
#           ### test-handler.mjs
### samconfig.toml
### template.yaml
```

Als Nächstes starten wir, um unsere `sam build` Anwendung zu erstellen:

```
sam-app $ sam build
Building codeuri: /Users/.../build-demo/sam-app runtime: None metadata: {'DockerTag':
 'nodejs18.x-v1', 'DockerContext': '/Users/.../build-demo/sam-app/hello-world',
 'Dockerfile': 'Dockerfile'} architecture: arm64 functions: HelloWorldFunction
Building image for HelloWorldFunction function
Setting DockerBuildArgs: {} for HelloWorldFunction function
Step 1/4 : FROM public.ecr.aws/lambda/nodejs:18
 ---> f5b68038c080
Step 2/4 : COPY app.mjs package*.json ./
 ---> Using cache
 ---> 834e565aae80
Step 3/4 : RUN npm install
 ---> Using cache
 ---> 31c2209dd7b5
Step 4/4 : CMD ["app.lambdaHandler"]
 ---> Using cache
 ---> 2ce2a438e89d
Successfully built 2ce2a438e89d
Successfully tagged helloworldfunction:nodejs18.x-v1

Build Succeeded

Built Artifacts  : .aws-sam/build
Built Template   : .aws-sam/build/template.yaml

Commands you can use next
=========================
```

```
[*] Validate SAM template: sam validate
[*] Invoke Function: sam local invoke
[*] Test Function in the Cloud: sam sync --stack-name {{stack-name}} --watch
[*] Deploy: sam deploy --guided
```

Erstellen einer Anwendung, die eine kompilierte Programmiersprache enthält

In diesem Beispiel erstellen wir mithilfe der Go Laufzeit eine Anwendung, die eine Lambda-Funktion enthält.

Zunächst initialisieren wir eine neue Anwendung `sam init` und konfigurieren unsere Anwendung für die Verwendung von: Go

```
$ sam init

...

Which template source would you like to use?
        1 - AWS Quick Start Templates
        2 - Custom Template Location
Choice: 1

Choose an AWS Quick Start application template
        1 - Hello World Example
        2 - Multi-step workflow
        3 - Serverless API
        ...
Template: 1

Use the most popular runtime and package type? (Python and zip) [y/N]: ENTER

Which runtime would you like to use?
        ...
        4 - dotnetcore3.1
        5 - go1.x
        6 - go (provided.al2)
        ...
Runtime: 5

What package type would you like to use?
        1 - Zip
        2 - Image
Package type: 1
```

```
Based on your selections, the only dependency manager available is mod.
We will proceed copying the template using mod.

Would you like to enable X-Ray tracing on the function(s) in your application?  [y/
N]: ENTER

Would you like to enable monitoring using CloudWatch Application Insights?
For more info, please view https://docs.aws.amazon.com/AmazonCloudWatch/latest/
monitoring/cloudwatch-application-insights.html [y/N]: ENTER

Project name [sam-app]: ENTER

Cloning from https://github.com/aws/aws-sam-cli-app-templates (process may take a
 moment)

    -----------------------
    Generating application:
    -----------------------
    Name: sam-app
    Runtime: go1.x
    Architectures: x86_64
    Dependency Manager: mod
    Application Template: hello-world
    Output Directory: .
    Configuration file: sam-app/samconfig.toml

    Next steps can be found in the README file at sam-app-go/README.md

...
```

Das AWS SAMCLI initialisiert die Anwendung. Im Folgenden finden Sie ein Beispiel für die Verzeichnisstruktur der Anwendung:

```
sam-app
### Makefile
### README.md
### events
#   ### event.json
### hello-world
#   ### go.mod
#   ### go.sum
#   ### main.go
```

```
#   ### main_test.go
### samconfig.toml
### template.yaml
```

Wir verweisen auf die `README.md` Datei für die Anforderungen dieser Anwendung.

```
...
## Requirements
* AWS CLI already configured with Administrator permission
* [Docker installed](https://www.docker.com/community-edition)
* [Golang](https://golang.org)
* SAM CLI - [Install the SAM CLI](https://docs.aws.amazon.com/serverless-application-
model/latest/developerguide/serverless-sam-cli-install.html)
...
```

Als Nächstes testen `sam local invoke` wir unsere Funktion. Dieser Befehl Go ist fehlerhaft, da er nicht auf unserem lokalen Computer installiert ist:

```
sam-app $ sam local invoke
Invoking hello-world (go1.x)
Local image was not found.
Removing rapid images for repo public.ecr.aws/sam/emulation-go1.x
Building
 image.....................................................................................
Using local image: public.ecr.aws/lambda/go:1-rapid-x86_64.

Mounting /Users/.../Playground/build/sam-app/hello-world as /var/task:ro,delegated
 inside runtime container
START RequestId: c6c5eddf-042b-4e1e-ba66-745f7c86dd31 Version: $LATEST
fork/exec /var/task/hello-world: no such file or directory: PathError
null
END RequestId: c6c5eddf-042b-4e1e-ba66-745f7c86dd31
REPORT RequestId: c6c5eddf-042b-4e1e-ba66-745f7c86dd31  Init Duration: 0.88 ms
 Duration: 175.75 ms Billed Duration: 176 ms Memory Size: 128 MB     Max Memory Used:
 128 MB
{"errorMessage":"fork/exec /var/task/hello-world: no such file or
 directory","errorType":"PathError"}%
```

Als Nächstes erstellen `sam build` wir unsere Anwendung. Wir stoßen auf einen Fehler, da Go es nicht auf unserem lokalen Computer installiert ist:

```
sam-app $ sam build
```

```
Starting Build use cache
Cache is invalid, running build and copying resources for following functions
 (HelloWorldFunction)
Building codeuri: /Users/.../Playground/build/sam-app/hello-world runtime: go1.x
 metadata: {} architecture: x86_64 functions: HelloWorldFunction

Build Failed
Error: GoModulesBuilder:Resolver - Path resolution for runtime: go1.x of binary: go was
 not successful
```

Wir könnten unseren lokalen Computer zwar so konfigurieren, dass er unsere Funktion korrekt erstellt, aber wir verwenden stattdessen die `--use-container` Option mit`sam build`. Der AWS SAMCLI lädt ein Container-Image herunter, erstellt unsere Funktion mit dem nativen GoModulesBuilder und kopiert die resultierende Binärdatei in unser `.aws-sam/build/HelloWorldFunction` Verzeichnis.

```
sam-app $ sam build --use-container
Starting Build use cache
Starting Build inside a container
Cache is invalid, running build and copying resources for following functions
 (HelloWorldFunction)
Building codeuri: /Users/.../build/sam-app/hello-world runtime: go1.x metadata: {}
 architecture: x86_64 functions: HelloWorldFunction

Fetching public.ecr.aws/sam/build-go1.x:latest-x86_64 Docker container
 image.....................................................................................
Mounting /Users/.../build/sam-app/hello-world as /tmp/samcli/source:ro,delegated inside
 runtime container
Running GoModulesBuilder:Build

Build Succeeded

Built Artifacts  : .aws-sam/build
Built Template   : .aws-sam/build/template.yaml

Commands you can use next
=========================
[*] Validate SAM template: sam validate
[*] Invoke Function: sam local invoke
[*] Test Function in the Cloud: sam sync --stack-name {{stack-name}} --watch
[*] Deploy: sam deploy --guided
```

Das Folgende ist ein Beispiel für das `.aws-sam` Verzeichnis:

```
.aws-sam
### build
#   ### HelloWorldFunction
#   #   ### hello-world
#   ### template.yaml
### build.toml
### cache
#   ### c860d011-4147-4010-addb-2eaa289f4d95
#       ### hello-world
### deps
```

Als nächstes rennen wir`sam local invoke`. Unsere Funktion wurde erfolgreich aufgerufen:

```
sam-app $ sam local invoke
Invoking hello-world (go1.x)
Local image is up-to-date
Using local image: public.ecr.aws/lambda/go:1-rapid-x86_64.

Mounting /Users/.../Playground/build/sam-app/.aws-sam/build/HelloWorldFunction as /var/
task:ro,delegated inside runtime container
START RequestId: cfc8ffa8-29f2-49d4-b461-45e8c7c80479 Version: $LATEST
END RequestId: cfc8ffa8-29f2-49d4-b461-45e8c7c80479
REPORT RequestId: cfc8ffa8-29f2-49d4-b461-45e8c7c80479  Init Duration: 1.20 ms
 Duration: 1782.46 ms        Billed Duration: 1783 ms        Memory Size: 128 MB
 Max Memory Used: 128 MB
{"statusCode":200,"headers":null,"multiValueHeaders":null,"body":"Hello,
 72.21.198.67\n"}%
```

Weitere Informationen

Weitere Informationen zur Verwendung des `sam build` Befehls finden Sie in den folgenden Abschnitten:

- Lernen AWS SAM: Sam Build — Serie „Lernen AWS SAM“ von Serverless Land amYouTube.
- Lernen AWS SAM | Sam Build | E3 — Serverless Land-Serie „Lernen AWS SAM“ weiter. YouTube
- AWS SAM build: wie es Artefakte für den Einsatz bereitstellt (Sessions With SAM S2E8) — Sessions mit Serien weiter. AWS SAM YouTube
- AWS SAM benutzerdefinierte Builds: Wie man Makefiles verwendet, um Builds in SAM anzupassen (S2E9) — Sessions mit laufenden Serien. AWS SAM YouTube

Standard-Build mit AWS SAM

Verwenden Sie den Befehl, um Ihre serverlose Anwendung zu erstellen. `gleicher Build` Dieser Befehl sammelt auch die Build-Artefakte der Abhängigkeiten Ihrer Anwendung und platziert sie im richtigen Format und am richtigen Speicherort für die nächsten Schritte, z. B. lokales Testen, Paketieren und Bereitstellen.

Sie geben die Abhängigkeiten Ihrer Anwendung in einer Manifestdatei wie `requirements.txt` (Python) oder `package.json` (Node.js) an, oder indem Sie die `Layers` Eigenschaft einer Funktionsressource verwenden. Die `Layers` Eigenschaft enthält eine Liste von AWS Lambda Layer-Ressourcen, von denen die Lambda-Funktion abhängt.

Das Format der Build-Artefakte Ihrer Anwendung hängt von den `PackageType` Eigenschaften der einzelnen Funktionen ab. Die Optionen für diese Eigenschaft sind:

- **`Zip`**— Ein ZIP-Dateiarchiv, das Ihren Anwendungscode und seine Abhängigkeiten enthält. Wenn Sie Ihren Code als ZIP-Dateiarchiv verpacken, müssen Sie eine Lambda-Laufzeit für Ihre Funktion angeben.
- **`Image`**— Ein Container-Image, das neben Ihrem Anwendungscode und seinen Abhängigkeiten auch das Basisbetriebssystem, die Laufzeit und Erweiterungen enthält.

Weitere Informationen zu Lambda-Pakettypen finden Sie unter Lambda-Bereitstellungspakete im AWS Lambda Developer Guide.

Themen

- Erstellen eines ZIP-Dateiarchivs
- Ein Container-Image erstellen
- Datei mit Container-Umgebungsvariablen
- Beschleunigen Sie die Build-Zeiten, indem Sie Ihr Projekt im Quellordner erstellen
- Beispiele
- Gebäudefunktionen außerhalb von AWS SAM

Erstellen eines ZIP-Dateiarchivs

Um Ihre serverlose Anwendung als ZIP-Dateiarchiv zu erstellen, deklarieren Sie Ihre serverlose `PackageType: Zip` Funktion.

AWS SAM erstellt Ihre Anwendung für die von Ihnen angegebene Architektur. Wenn Sie keine Architektur angeben, AWS SAM verwendet `x86_64` standardmäßig.

Wenn Ihre Lambda-Funktion von Paketen abhängt, die nativ kompilierte Programme enthalten, verwenden Sie das `--use-container` Flag. Dieses Flag kompiliert Ihre Funktionen lokal in einem Container, der sich wie eine Lambda-Umgebung verhält, sodass sie im richtigen Format vorliegen, wenn Sie sie in der Cloud bereitstellen. AWS

Wenn Sie die `--use-container` Option verwenden, wird das Container-Image standardmäßig aus Amazon ECR AWS SAM Public abgerufen. Wenn Sie ein Container-Image aus einem anderen Repository oder für eine bestimmte Version von AWS SAM CLI abrufen möchten, können Sie die `--build-image` Option verwenden und den URI eines alternativen Container-Images angeben. Im Folgenden finden Sie zwei Beispielbefehle zum Erstellen von Anwendungen mit Container-Images aus einer bestimmten Version von AWS SAM CLI:

```
# Build a Node.js 20 application using a container image for a specific version of AWS
 SAM CLI (1.136.0)
sam build --use-container --build-image public.ecr.aws/sam/build-nodejs22.x:1.136.0

# Build a function resource using the Python 3.13 container image from a specific
 version of AWS SAM CLI (1.136.0)(
sam build --use-container --build-image Function1=public.ecr.aws/sam/build-
python3.13:1.136.0
```

Weitere Beispiele für die Erstellung einer ZIP-Dateiarchivanwendung finden Sie im Abschnitt Beispiele weiter unten in diesem Thema.

Ein Container-Image erstellen

Um Ihre serverlose Anwendung als Container-Image zu erstellen, deklarieren `PackageType: Image` Sie Ihre serverlose Funktion. Sie müssen auch das `Metadata` Ressourcenattribut mit den folgenden Einträgen deklarieren:

`Dockerfile`

Der Name der Dockerfile, die der Lambda-Funktion zugeordnet ist.

`DockerContext`

Der Speicherort des Dockerfiles.

`DockerTag`

(Optional) Ein Tag, das auf das erstellte Image angewendet werden soll.

`DockerBuildArgs`

Generieren Sie Argumente für den Build.

> **Important**
>
> Das AWS SAMCLI redigiert oder verschleiert keine Informationen, die Sie in Argumenten angeben. `DockerBuildArgs` Es wird dringend empfohlen, mit diesem Abschnitt keine vertraulichen Informationen wie Passwörter oder Secrets zu speichern.

`DockerImage`

(Optional) Ein benutzerdefinierter Name für das erstellte Docker-Image. Wenn nicht angegeben, wird die logische Ressourcen-ID der Funktion verwendet.

Im Folgenden finden Sie ein Beispiel für einen Abschnitt mit einem `Metadata` Ressourcenattribut:

```
Metadata:
  Dockerfile: Dockerfile
  DockerContext: ./hello_world
  DockerTag: v1
  DockerImage: my-custom-image
```

Informationen zum Herunterladen einer Beispielanwendung, die mit dem `Image` Pakettyp konfiguriert ist, finden Sie unterTutorial: Stellen Sie eine Hello World-Anwendung bereit mit AWS SAM. Wenn Sie gefragt werden, welchen Pakettyp Sie installieren möchten, wählen Sie`Image`.

> **Note**
>
> Wenn Sie in Ihrem Dockerfile ein Basis-Image mit mehreren Architekturen angeben, AWS SAM erstellt es Ihr Container-Image für die Architektur Ihres Host-Computers. Um für eine andere Architektur zu erstellen, geben Sie ein Basis-Image an, das die spezifische Zielarchitektur verwendet.

Standardmäßig AWS SAMCLI verwendet der beim Erstellen von Container-Images die Docker Legacy-Build-Engine. Um ein verbessertes Container-Backend zu verwendenBuildKit, das die Leistung verbessert und komplexere Builds verarbeitet, übergeben Sie das `--use-buildkit` Flag mit Ihrem Build-Befehl:

```
$ sam build --use-buildkit
```

`--use-buildkit`Für die Verwendung muss das Docker oder Finch CLI installiert sein.

Datei mit Container-Umgebungsvariablen

Um eine JSON-Datei bereitzustellen, die Umgebungsvariablen für den Build-Container enthält, verwenden Sie das `--container-env-var-file` Argument zusammen mit dem `sam build` Befehl. Sie können eine einzelne Umgebungsvariable angeben, die für alle serverlosen Ressourcen gilt, oder unterschiedliche Umgebungsvariablen für jede Ressource.

Format

Das Format für die Übergabe von Umgebungsvariablen an einen Build-Container hängt davon ab, wie viele Umgebungsvariablen Sie für Ihre Ressourcen bereitstellen.

Um eine einzige Umgebungsvariable für alle Ressourcen bereitzustellen, geben Sie ein `Parameters` Objekt wie das Folgende an:

```
{
  "Parameters": {
    "GITHUB_TOKEN": "TOKEN_GLOBAL"
  }
}
```

Um für jede Ressource unterschiedliche Umgebungsvariablen bereitzustellen, geben Sie Objekte für jede Ressource wie folgt an:

```
{
  "MyFunction1": {
    "GITHUB_TOKEN": "TOKEN1"
  },
  "MyFunction2": {
    "GITHUB_TOKEN": "TOKEN2"
```

```
  }
}
```

Speichern Sie Ihre Umgebungsvariablen als Datei, z. B. mit dem Namen`env.json`. Der folgende Befehl verwendet diese Datei, um Ihre Umgebungsvariablen an den Build-Container zu übergeben:

```
sam build --use-container --container-env-var-file env.json
```

Precedence

- Die Umgebungsvariablen, die Sie für bestimmte Ressourcen angeben, haben Vorrang vor der einzelnen Umgebungsvariablen für alle Ressourcen.
- Umgebungsvariablen, die Sie in der Befehlszeile angeben, haben Vorrang vor Umgebungsvariablen in einer Datei.

Beschleunigen Sie die Build-Zeiten, indem Sie Ihr Projekt im Quellordner erstellen

Für unterstützte Laufzeiten und Build-Methoden können Sie die `--build-in-source` Option verwenden, um Ihr Projekt direkt im Quellordner zu erstellen. Standardmäßig werden die AWS SAM CLI Builds in einem temporären Verzeichnis gespeichert, was das Kopieren von Quellcode und Projektdateien beinhaltet. Damit befinden `--build-in-source` sich die AWS SAM CLI Builds direkt in Ihrem Quellordner, was den Build-Prozess beschleunigt, da keine Dateien mehr in ein temporäres Verzeichnis kopiert werden müssen.

Eine Liste der unterstützten Laufzeiten und Build-Methoden finden Sie unter`--build-in-source`.

Beispiele

Beispiel 1: ZIP-Dateiarchiv

Mit den folgenden `sam build` Befehlen wird ein ZIP-Dateiarchiv erstellt:

```
# Build all functions and layers, and their dependencies
sam build

# Run the build process inside a Docker container that functions like a Lambda
 environment
sam build --use-container
```

```
# Build a Node.js 20 application using a container image for a specific version of AWS
 SAM CLI (1.136.0)
sam build --use-container --build-image public.ecr.aws/sam/build-nodejs22.x:1.136.0

# Build a function resource using the Python 3.13 container image from a specific
 version of AWS SAM CLI (1.136.0)(
sam build --use-container --build-image Function1=public.ecr.aws/sam/build-
python3.13:1.136.0

# Build and run your functions locally
sam build && sam local invoke

# For more options
sam build --help
```

Beispiel 2: Container-Image

Die folgende AWS SAM Vorlage wird als Container-Image erstellt:

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      PackageType: Image
      ImageConfig:
        Command: ["app.lambda_handler"]
    Metadata:
      Dockerfile: Dockerfile
      DockerContext: ./hello_world
      DockerTag: v1
```

Im Folgenden finden Sie ein Beispiel für eine Dockerfile:

```
FROM public.ecr.aws/lambda/python:3.12

COPY app.py requirements.txt ./

RUN python3.12 -m pip install -r requirements.txt

# Overwrite the command by providing a different command directly in the template.
CMD ["app.lambda_handler"]
```

Beispiel 3: npm ci

Für Anwendungen mit der Datei Node.js können Sie `npm ci` anstelle von `npm install` Abhängigkeiten verwenden. Geben Sie zur Verwendung `npm ci` `BuildProperties` im `Metadata` Ressourcenattribut Ihrer Lambda-Funktion `UseNpmCi: True` unter an. Um sie verwenden zu können`npm ci`, muss Ihre Anwendung eine `package-lock.json` `npm-shrinkwrap.json` OR-Datei in der `CodeUri` for your Lambda-Funktion enthalten.

Das folgende Beispiel dient `npm ci` zur Installation von Abhängigkeiten bei der Ausführung`sam build`:

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: hello-world/
      Handler: app.handler
      Runtime: nodejs20.x
      Architectures:
        - x86_64
      Events:
        HelloWorld:
          Type: Api
          Properties:
            Path: /hello
            Method: get
    Metadata:
      BuildProperties:
        UseNpmCi: True
```

Python-Elternpakete

Bei Python-Anwendungen können Sie Ihre Paketstruktur während des Build-Prozesses beibehalten, um absolute Importe zu ermöglichen. Um die Paketstruktur beizubehalten, geben Sie `BuildProperties` im `Metadata` Ressourcenattribut Ihrer Lambda-Funktion `ParentPackageMode` unter an.

Im folgenden Beispiel wird die `app` Paketstruktur bei der Ausführung beibehalten: `sam build`

```
Resources:
  HelloWorldFunction:
```

```
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: hello-world/
      Handler: app.main.handler
      Runtime: python3.12
      Architectures:
        - x86_64
    Metadata:
      BuildProperties:
        ParentPackageMode: explicit
        ParentPackages: app
```

Mit dieser Konfiguration kann Ihr Code absolute Importe wie `from app.utils import logger` anstelle von relativen Importen wie `from .utils import logger` verwenden.

Gebäudefunktionen außerhalb von AWS SAM

Standardmäßig werden beim Ausführen sam build alle Ihre Funktionsressourcen AWS SAM erstellt. Weitere Optionen sind:

- Alle Funktionsressourcen außerhalb von erstellen AWS SAM— Wenn Sie all Ihre Funktionsressourcen manuell oder mit einem anderen Tool erstellen, sam build ist dies nicht erforderlich. Sie können den Vorgang überspringen sam build und mit dem nächsten Schritt in Ihrem Prozess fortfahren, z. B. dem Durchführen lokaler Tests oder dem Bereitstellen Ihrer Anwendung.
- Erstellen Sie einige Funktionsressourcen außerhalb von AWS SAM — Wenn Sie einige Ihrer Funktionsressourcen erstellen und gleichzeitig andere Funktionsressourcen außerhalb von erstellen möchten AWS SAM AWS SAM, können Sie dies in Ihrer AWS SAM Vorlage angeben.

Erstellen Sie einige Funktionsressourcen außerhalb von AWS SAM

Damit bei der Verwendung eine Funktion AWS SAM übersprungen sam build wird, konfigurieren Sie in Ihrer AWS SAM Vorlage Folgendes:

1. Fügen Sie die `SkipBuild: True` Metadaten-Eigenschaft zu Ihrer Funktion hinzu.
2. Geben Sie den Pfad zu Ihren erstellten Funktionsressourcen an.

Hier ist ein Beispiel, das `TestFunction` so konfiguriert ist, dass es übersprungen wird. Die erstellten Ressourcen befinden sich unter`built-resources/TestFunction.zip`.

```
TestFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: built-resources/TestFunction.zip
    Handler: TimeHandler::handleRequest
    Runtime: java11
  Metadata:
    SkipBuild: True
```

Wenn Sie jetzt laufensam build, AWS SAM werden Sie Folgendes tun:

1. AWS SAM überspringt Funktionen, die mit konfiguriert wurden`SkipBuild: True`.
2. AWS SAM erstellt alle anderen Funktionsressourcen und speichert sie im `.aws-sam` Build-Verzeichnis.
3. Bei übersprungenen Funktionen wird ihre Vorlage im `.aws-sam` Build-Verzeichnis automatisch aktualisiert, sodass sie auf den angegebenen Pfad zu Ihren erstellten Funktionsressourcen verweist.

   Hier ist ein Beispiel für die zwischengespeicherte Vorlage für `TestFunction` im `.aws-sam` Build-Verzeichnis:

   ```
   TestFunction:
     Type: AWS::Serverless::Function
     Properties:
       CodeUri: ../../built-resources/TestFunction.zip
       Handler: TimeHandler::handleRequest
       Runtime: java11
     Metadata:
       SkipBuild: True
   ```

Passen Sie Builds an mit AWS SAM

Sie können Ihren Build so anpassen, dass er bestimmte Lambda-Funktionen oder Lambda-Schichten enthält. Eine Funktion ist eine Ressource, die Sie aufrufen können, um Ihren Code in Lambda auszuführen. Eine Lambda-Schicht ermöglicht es Ihnen, Code aus einer Lambda-Funktion zu extrahieren, der dann für mehrere Lambda-Funktionen wiederverwendet werden kann. Sie können Ihren Build mit bestimmten Lambda-Funktionen anpassen, wenn Sie sich auf die Entwicklung und Bereitstellung einzelner serverloser Funktionen konzentrieren möchten, ohne die Komplexität der

Verwaltung gemeinsam genutzter Abhängigkeiten oder Ressourcen. Darüber hinaus können Sie eine Lambda-Schicht erstellen, um die Größe Ihrer Bereitstellungspakete zu reduzieren, die Logik der Kernfunktionen von Abhängigkeiten zu trennen und Abhängigkeiten für mehrere Funktionen gemeinsam zu nutzen.

In den Themen in diesem Abschnitt werden einige der verschiedenen Möglichkeiten beschrieben, mit AWS SAM denen Sie Lambda-Funktionen erstellen können. Dazu gehören die Erstellung von Lambda-Funktionen mit Kundenlaufzeiten und die Erstellung von Lambda-Layern. Mit benutzerdefinierten Laufzeiten können Sie eine Sprache installieren und verwenden, die nicht in den Lambda-Laufzeiten im Developer Guide aufgeführt ist AWS Lambda . Auf diese Weise können Sie eine spezielle Ausführungsumgebung für die Ausführung serverloser Funktionen und Anwendungen erstellen. Wenn Sie nur Lambda-Schichten erstellen (anstatt Ihre gesamte Anwendung zu erstellen), können Sie in mehrfacher Hinsicht davon profitieren. Es kann Ihnen helfen, die Größe Ihrer Bereitstellungspakete zu reduzieren, die Logik der Kernfunktionen von Abhängigkeiten zu trennen und Abhängigkeiten für mehrere Funktionen gemeinsam zu nutzen.

Weitere Informationen zu Funktionen finden Sie unter Lambda-Konzepte im AWS Lambda Developer Guide.

Themen

- Lambda-Funktionen von Node.js mit esbuild in erstellen AWS SAM
- Erstellen von.NET-Lambda-Funktionen mit nativer AOT-Kompilierung in AWS SAM
- Erstellen von Rust Lambda-Funktionen mit Cargo Lambda in AWS SAM
- Python-Lambda-Funktionen mit uv in erstellen AWS SAM
- Erstellen von Lambda-Funktionen mit benutzerdefinierten Laufzeiten in AWS SAM
- Aufbau von Lambda-Schichten in AWS SAM

Lambda-Funktionen von Node.js mit esbuild in erstellen AWS SAM

Um die AWS Lambda Funktionen von Node.js zu erstellen und zu verpacken, können Sie den AWS SAMCLI mit dem JavaScript Esbuild-Bundler verwenden. Der Esbuild-Bundler unterstützt Lambda-Funktionen, in die Sie schreiben. TypeScript

Um eine Lambda-Funktion von Node.js mit esbuild zu erstellen, fügen Sie Ihrer `AWS:Serverless::Function` Ressource ein `Metadata` Objekt hinzu und geben Sie `esbuild` für die an. `BuildMethod` Wenn Sie den sam build Befehl ausführen, AWS SAM verwendet esbuild, um Ihren Lambda-Funktionscode zu bündeln.

Eigenschaften von Metadaten

Das `Metadata` Objekt unterstützt die folgenden Eigenschaften für esbuild.

BuildMethod

Gibt den Bundler für Ihre Anwendung an. Der einzige unterstützte Wert ist `esbuild`.

BuildProperties

Gibt die Build-Eigenschaften für Ihren Lambda-Funktionscode an.

Das `BuildProperties` Objekt unterstützt die folgenden Eigenschaften für esbuild. Alle Eigenschaften sind optional. AWS SAM Verwendet standardmäßig Ihren Lambda-Funktionshandler für den Einstiegspunkt.

EntryPoints

Gibt Einstiegspunkte für Ihre Anwendung an.

Extern

Gibt die Liste der Pakete an, die beim Build weggelassen werden sollen. Weitere Informationen finden Sie auf der esbuildWebsite unter Extern.

Format

Gibt das Ausgabeformat der generierten JavaScript Dateien in Ihrer Anwendung an. Weitere Informationen finden Sie unter Format auf der Esbuild-Website.

Loader

Gibt die Liste der Konfigurationen zum Laden von Daten für einen bestimmten Dateityp an.

MainFields

Gibt an, welche `package.json` Felder beim Auflösen eines Pakets importiert werden sollen. Der Standardwert ist `main,module`.

Verkleinern

Gibt an, ob der gebündelte Ausgabecode minimiert werden soll. Der Standardwert ist `true`.

OutExtension

Passen Sie die Dateierweiterung der Dateien an, die esbuild generiert. Weitere Informationen finden Sie unter Out-Erweiterung auf der Esbuild-Website.

Quellenkarte

Gibt an, ob der Bundler eine Quellzuordnungsdatei erzeugt. Der Standardwert ist `false`.

Wenn auf gesetzt`true`, `NODE_OPTIONS: --enable-source-maps` wird es an die Umgebungsvariablen der Lambda-Funktion angehängt, und eine Quellenzuordnung wird generiert und in die Funktion aufgenommen.

Alternativ `NODE_OPTIONS: --enable-source-maps` wird, wenn in den Umgebungsvariablen der Funktion enthalten `Sourcemap` ist, automatisch auf gesetzt. `true`

Im Konfliktfall hat `Sourcemap: false` dies Vorrang vor. `NODE_OPTIONS: --enable-source-maps`

> **Note**
>
> Standardmäßig verschlüsselt Lambda alle ruhenden Umgebungsvariablen mit AWS Key Management Service ()AWS KMS. Wenn Sie Quellzuordnungen verwenden, muss die Ausführungsrolle Ihrer Funktion über die Berechtigung zum Ausführen der Aktion verfügen, damit die `kms:Encrypt` Bereitstellung erfolgreich ist.

SourcesContent

Gibt an, ob der Quellcode in die Quellzuordnungsdatei aufgenommen werden soll. Konfigurieren Sie diese Eigenschaft, wenn sie auf gesetzt `Sourcemap` ist`'true'`.

- Geben Sie `SourcesContent: 'true'` an, dass der gesamte Quellcode eingeschlossen werden soll.
- Geben Sie `SourcesContent: 'false'` an, dass der gesamte Quellcode ausgeschlossen werden soll. Dies führt zu kleineren Quellzuordnungsdateien, was in der Produktion nützlich ist, da die Startzeiten reduziert werden. Der Quellcode wird jedoch nicht im Debugger verfügbar sein.

Der Standardwert ist `SourcesContent: true`.

Weitere Informationen finden Sie unter Quelleninhalt auf der Esbuild-Website.

Target

Gibt die ECMAScript Zielversion an. Der Standardwert ist `es2020`.

TypeScript Beispiel für eine Lambda-Funktion

Das folgende Beispiel für einen AWS SAM Vorlagenausschnitt verwendet esbuild, um eine Node.js Lambda-Funktion aus dem Code in zu erstellen. TypeScript `hello-world/app.ts`

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: hello-world/
      Handler: app.handler
      Runtime: nodejs20.x
      Architectures:
        - x86_64
      Events:
        HelloWorld:
          Type: Api
          Properties:
            Path: /hello
            Method: get
      Environment:
        Variables:
          NODE_OPTIONS: --enable-source-maps
    Metadata:
      BuildMethod: esbuild
      BuildProperties:
        Format: esm
        Minify: false
        OutExtension:
          - .js=.mjs
        Target: "es2020"
        Sourcemap: true
        EntryPoints:
          - app.ts
        External:
          - "<package-to-exclude>"
```

Erstellen von.NET-Lambda-Funktionen mit nativer AOT-Kompilierung in AWS SAM

Erstellen und verpacken Sie Ihre .NET AWS Lambda 8-Funktionen mit AWS Serverless Application Model (AWS SAM). Verwenden Sie dabei die native Kompilierung Ahead-of-Time (AOT), um die AWS Lambda Kaltstartzeiten zu verbessern.

Topics

- Überblick über .NET 8 Native AOT
- Verwendung AWS SAM mit Ihren.NET-8-Lambda-Funktionen
- Voraussetzungen für die Installation
- Definieren Sie.NET 8 Lambda-Funktionen in Ihrer Vorlage AWS SAM
- Erstellen Sie Ihre Anwendung mit AWS SAMCLI
- Weitere Informationen

Überblick über .NET 8 Native AOT

In der Vergangenheit hatten die.NET-Lambda-Funktionen Kaltstartzeiten, die sich auf die Benutzererfahrung, die Systemlatenz und die Nutzungskosten Ihrer serverlosen Anwendungen auswirken. Mit.NET Native AOT-Kompilierung können Sie die Kaltstartzeiten Ihrer Lambda-Funktionen verbessern. Weitere Informationen zu Native AOT für .NET 8 finden Sie unter Verwenden von Native AOT im Dotnet-Repository. GitHub

Verwendung AWS SAM mit Ihren.NET-8-Lambda-Funktionen

Gehen Sie wie folgt vor, um Ihre .NET 8-Lambda-Funktionen mit der AWS Serverless Application Model (AWS SAM) zu konfigurieren:

- Installieren Sie die erforderlichen Komponenten auf Ihrem Entwicklungscomputer.
- Definieren Sie .NET 8 Lambda-Funktionen in Ihrer AWS SAM Vorlage.
- Erstellen Sie Ihre Anwendung mit dem AWS SAMCLI.

Voraussetzungen für die Installation

Die folgenden Voraussetzungen sind erforderlich:

- Die AWS SAMCLI
- Die.NET Core CLI
- Das globale .NET Core-Tool von Amazon.Lambda.Tools
- Docker

Installieren Sie das AWS SAMCLI

1. Um zu überprüfen, ob Sie das bereits AWS SAMCLI installiert haben, führen Sie Folgendes aus:

   ```
   sam --version
   ```

2. Informationen zur Installation von AWS SAMCLI finden Sie unterInstallieren Sie das AWS SAMCLI.
3. Informationen zum Upgrade einer installierten Version von finden Sie unterAktualisierung des AWS SAMCLI. AWS SAMCLI

Installieren Sie das.NET Core CLI

1. Informationen zum Herunterladen und Installieren von.NET Core-CLI finden Sie unter .NET von der Microsoft-Website herunterladen.
2. Weitere Informationen zur.NET Core CLI finden Sie unter.NET Core CLI im AWS Lambda Developer Guide.

Installieren Sie das Amazon.Lambda.Tools .NET Core Global Tool

1. Führen Sie den folgenden Befehl aus:

   ```
   dotnet tool install -g Amazon.Lambda.Tools
   ```

2. Wenn Sie das Tool bereits installiert haben, können Sie sich mit dem folgenden Befehl vergewissern, dass Sie die neueste Version verwenden:

   ```
   dotnet tool update -g Amazon.Lambda.Tools
   ```

3. Weitere Informationen zum Amazon.Lambda.Tools .NET Core Global Tool finden Sie im AWS Extensions for .NET CLI Repository unter. GitHub

Installieren Docker

- Das Erstellen mit nativem AOT muss installiert werden. Docker Installationsanweisungen finden Sie unter Installation von Docker zur Verwendung mit dem AWS SAMCLI.

Definieren Sie.NET 8 Lambda-Funktionen in Ihrer Vorlage AWS SAM

Um eine zu definieren. NET8 Gehen Sie in der Lambda-Funktion in Ihrer AWS SAM Vorlage wie folgt vor:

1. Führen Sie den folgenden Befehl in einem Startverzeichnis Ihrer Wahl aus:

   ```
   sam init
   ```

2. Wählen Sie `AWS Quick Start Templates` diese Option, um eine Startvorlage auszuwählen.
3. Wählen Sie als Vorlage `Hello World Example` aus.
4. Geben Sie ein, dass Sie nicht den gängigsten Laufzeit- und Pakettyp verwenden möchtenn.
5. Wählen Sie für Runtime`dotnet8`.
6. Wählen Sie als Pakettyp`Zip`.
7. Wählen Sie für Ihre Starter-Vorlage`Hello World Example using native AOT`.

Installieren Docker

- Building with Native AOT, Docker muss installiert sein. Installationsanweisungen finden Sie unter Installation von Docker zur Verwendung mit dem AWS SAMCLI.

```
Resources:
HelloWorldFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: ./src/HelloWorldAot/
    Handler: bootstrap
    Runtime: dotnet8
    Architectures:
      - x86_64
    Events:
      HelloWorldAot:
        Type: Api
        Properties:
          Path: /hello
          Method: get
```

> **Note**
>
> Wenn die `Event` Eigenschaft von auf gesetzt `AWS::Serverless::Function` ist`Api`, die `RestApiId` Eigenschaft aber nicht angegeben ist, wird die `AWS::ApiGateway::RestApi` CloudFormation Ressource AWS SAM generiert.

Erstellen Sie Ihre Anwendung mit AWS SAMCLI

Führen Sie im Stammverzeichnis Ihres Projekts den `sam build` Befehl aus, um mit der Erstellung Ihrer Anwendung zu beginnen. Wenn die `PublishAot` Eigenschaft in Ihrer .NET 8-Projektdatei definiert wurde, AWS SAMCLI wird sie mit der systemeigenen AOT-Kompilierung erstellt. Weitere Informationen zu dieser `PublishAot` Eigenschaft finden Sie unter Native AOT Deployment in der .NET-Dokumentation von Microsoft.

Um Ihre Funktion zu erstellen, AWS SAMCLI ruft die .NET Core CLI auf, die das Amazon.Lambda.Tools .NET Core Global Tool verwendet.

> **Note**
>
> Wenn beim Erstellen eine `.sln` Datei im selben oder einem übergeordneten Verzeichnis Ihres Projekts vorhanden ist, wird das Verzeichnis, das die `.sln` Datei enthält, in den Container gemountet. Wenn eine `.sln` Datei nicht gefunden wird, wird nur der Projektordner bereitgestellt. Wenn Sie eine Multiprojektanwendung erstellen, stellen Sie daher sicher, dass sich die `.sln` Datei im richtigen Verzeichnis befindet.

Weitere Informationen

Weitere Informationen zum Erstellen von.NET 8-Lambda-Funktionen finden Sie unter Einführung in die .NET 8-Laufzeit für AWS Lambda.

Eine Referenz zu dem sam build Befehl finden Sie untergleicher Build.

Erstellen von Rust Lambda-Funktionen mit Cargo Lambda in AWS SAM

Verwenden Sie die AWS Serverless Application Model Befehlszeilenschnittstelle (AWS SAMCLI) mit Ihren AWS Lambda Rust-Funktionen.

Topics

- Voraussetzungen
- Konfigurieren AWS SAM zur Verwendung mit Rust Lambda-Funktionen
- Beispiele
- Die Optimierung von Rust-Builds ist eingebaut GitHub Aktionen

Voraussetzungen

RustSprache

Informationen zur Installation Rust finden Sie unter Installieren Rust auf der Rust Sprachwebsite.

Cargo Lambda

Das AWS SAMCLI erfordert die Installation von Cargo Lambda, einen Unterbefehl fürCargo. Anweisungen zur Installation finden Sie in der Cargo Lambda Dokumentation unter Installation.

Docker

Das Erstellen und Testen von Rust Lambda-Funktionen erfordertDocker. Installationsanweisungen finden Sie unter Installieren von Docker.

Konfigurieren AWS SAM zur Verwendung mit Rust Lambda-Funktionen

Schritt 1: Konfiguriere deine AWS SAM Vorlage

Konfigurieren Sie Ihre AWS SAM Vorlage wie folgt:

- Binär — Optional. Geben Sie an, wenn ein einzelnes Cargo Paket mehr als eine Binärdatei definiert, um zu identifizieren, welche Binärdatei für diese Funktion erstellt werden soll. Sie benötigen diese Eigenschaft nicht, wenn es sich bei jeder Funktion um ein eigenes Cargo Paket handelt, z. B. in einem Cargo Workspace.
- BuildMethod – `rust-cargolambda`.
- CodeUri— Pfad zu Ihrer `Cargo.toml` Datei.
- Handler —`bootstrap`.
- Laufzeit —`provided.al2023`.

Weitere Informationen zu benutzerdefinierten Laufzeiten finden Sie unter Benutzerdefinierte AWS Lambda Laufzeiten im AWS Lambda Entwicklerhandbuch.

Hier ist ein Beispiel für eine konfigurierte AWS SAM Vorlage:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Metadata:
      BuildMethod: rust-cargolambda
      BuildProperties: function_a
    Properties:
      CodeUri: ./rust_app
      Handler: bootstrap
      Runtime: provided.al2023
...
```

Schritt 2: Verwenden Sie die AWS SAM CLI mit deiner Rust Lambda-Funktion

Verwenden Sie einen beliebigen AWS SAMCLI Befehl mit Ihrer AWS SAM Vorlage. Weitere Informationen finden Sie unter AWS SAM CLI.

Beispiele

Beispiel „Hallo Welt“

In diesem Beispiel erstellen wir die Hello World-Beispielanwendung, die wir Rust als Laufzeit verwenden.

Zuerst initialisieren wir eine neue serverlose Anwendung mit. `sam init` Während des interaktiven Ablaufs wählen wir die Hello World-Anwendung aus und wählen die Rust-Laufzeit aus.

```
$ sam init
...
Which template source would you like to use?
        1 - AWS Quick Start Templates
        2 - Custom Template Location
Choice: 1

Choose an AWS Quick Start application template
        1 - Hello World Example
        2 - Multi-step workflow
        3 - Serverless API
```

```
        ...
Template: 1

Use the most popular runtime and package type? (Python and zip) [y/N]: ENTER

Which runtime would you like to use?
        1 - dotnet8
        2 - dotnet6
        3 - go (provided.al2)
        ...
        18 - python3.11
        19 - python3.10
        20 - ruby4.0
        21 - ruby3.3
        22 - ruby3.2
        23 - rust (provided.al2)
        24 - rust (provided.al2023)
Runtime: 24

Based on your selections, the only Package type available is Zip.
We will proceed to selecting the Package type as Zip.

Based on your selections, the only dependency manager available is cargo.
We will proceed copying the template using cargo.

Would you like to enable X-Ray tracing on the function(s) in your application?  [y/
N]: ENTER

Would you like to enable monitoring using CloudWatch Application Insights?
For more info, please view https://docs.aws.amazon.com/AmazonCloudWatch/latest/
monitoring/cloudwatch-application-insights.html [y/N]: ENTER

Project name [sam-app]: hello-rust

    -----------------------
    Generating application:
    -----------------------
    Name: hello-rust
    Runtime: rust (provided.al2023)
    Architectures: x86_64
    Dependency Manager: cargo
    Application Template: hello-world
    Output Directory: .
    Configuration file: hello-rust/samconfig.toml
```

```
    Next steps can be found in the README file at hello-rust/README.md


Commands you can use next
=========================
[*] Create pipeline: cd hello-rust && sam pipeline init --bootstrap
[*] Validate SAM template: cd hello-rust && sam validate
[*] Test Function in the Cloud: cd hello-rust && sam sync --stack-name {stack-name} --
watch
```

Das Folgende ist die Struktur unserer Hello World-Anwendung:

```
hello-rust
### README.md
### events
#   ### event.json
### rust_app
#   ### Cargo.toml
#   ### src
#       ### main.rs
### samconfig.toml
### template.yaml
```

In unserer AWS SAM Vorlage ist unsere Rust Funktion wie folgt definiert:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Metadata:
      BuildMethod: rust-cargolambda
    Properties:
      CodeUri: ./rust_app
      Handler: bootstrap
      Runtime: provided.al2023
      Architectures:
        - x86_64
      Events:
        HelloWorld:
          Type: Api
```

```
          Properties:
            Path: /hello
            Method: get
```

Als Nächstes erstellen wir `sam build` unsere Anwendung und bereiten uns auf die Bereitstellung vor. Das AWS SAMCLI erstellt ein `.aws-sam` Verzeichnis und organisiert unsere Build-Artefakte dort. Unsere Funktion wird unter Verwendung einer ausführbaren Binärdatei unter `.aws-sam/build/HelloWorldFunction/bootstrap` erstellt Cargo Lambda und gespeichert.

> **Note**
>
> Wenn Sie den sam local invoke Befehl in macOS ausführen möchten, müssen Sie vor dem Aufruf andere Funktionen erstellen. Verwenden Sie dazu den folgenden Befehl:
>
> - SAM_BUILD_MODE=debug sam build
>
> Dieser Befehl wird nur benötigt, wenn lokale Tests durchgeführt werden. Dies wird beim Erstellen für die Bereitstellung nicht empfohlen.

```
hello-rust$ sam build
Starting Build use cache
Cache is invalid, running build and copying resources for following functions
 (HelloWorldFunction)
Building codeuri: /Users/.../hello-rust/rust_app runtime: provided.al2023 metadata:
 {'BuildMethod': 'rust-cargolambda'} architecture: x86_64 functions: HelloWorldFunction
Running RustCargoLambdaBuilder:CargoLambdaBuild
Running RustCargoLambdaBuilder:RustCopyAndRename

Build Succeeded

Built Artifacts  : .aws-sam/build
Built Template   : .aws-sam/build/template.yaml

Commands you can use next
=========================
[*] Validate SAM template: sam validate
[*] Invoke Function: sam local invoke
[*] Test Function in the Cloud: sam sync --stack-name {{stack-name}} --watch
[*] Deploy: sam deploy --guided
```

Als Nächstes stellen wir unsere Anwendung mithilfe von bereits`sam deploy --guided`.

```
hello-rust$ sam deploy --guided

Configuring SAM deploy
======================

        Looking for config file [samconfig.toml] :  Found
        Reading default arguments  :  Success

        Setting default arguments for 'sam deploy'
        =========================================
        Stack Name [hello-rust]: ENTER
        AWS Region [us-west-2]: ENTER
        #Shows you resources changes to be deployed and require a 'Y' to initiate
 deploy
        Confirm changes before deploy [Y/n]: ENTER
        #SAM needs permission to be able to create roles to connect to the resources in
 your template
        Allow SAM CLI IAM role creation [Y/n]: ENTER
        #Preserves the state of previously provisioned resources when an operation
 fails
        Disable rollback [y/N]: ENTER
        HelloWorldFunction may not have authorization defined, Is this okay? [y/N]: y
        Save arguments to configuration file [Y/n]: ENTER
        SAM configuration file [samconfig.toml]: ENTER
        SAM configuration environment [default]: ENTER

        Looking for resources needed for deployment:

        ...

        Uploading to hello-rust/56ba6585d80577dd82a7eaaee5945c0b  817973 / 817973
 (100.00%)

        Deploying with following values
        ===============================
        Stack name                   : hello-rust
        Region                       : us-west-2
        Confirm changeset            : True
        Disable rollback             : False
        Deployment s3 bucket         : aws-sam-cli-managed-default-samclisam-s3-demo-
bucket-1a4x26zbcdkqr
        Capabilities                 : ["CAPABILITY_IAM"]
```

```
        Parameter overrides          : {}
        Signing Profiles             : {}

Initiating deployment
=====================

        Uploading to hello-rust/a4fc54cb6ab75dd0129e4cdb564b5e89.template  1239 / 1239
 (100.00%)


Waiting for changeset to be created..

CloudFormation stack changeset
-----------------------------------------------------------------------------------------------
Operation                   LogicalResourceId          ResourceType
 Replacement
-----------------------------------------------------------------------------------------------
+ Add                       HelloWorldFunctionHelloW   AWS::Lambda::Permission    N/A

                            orldPermissionProd

...
-----------------------------------------------------------------------------------------------

Changeset created successfully. arn:aws:cloudformation:us-
west-2:012345678910:changeSet/samcli-deploy1681427201/
f0ef1563-5ab6-4b07-9361-864ca3de6ad6


Previewing CloudFormation changeset before deployment
=====================================================
Deploy this changeset? [y/N]: y

2023-04-13 13:07:17 - Waiting for stack create/update to complete

CloudFormation events from stack operations (refresh every 5.0 seconds)
-----------------------------------------------------------------------------------------------
ResourceStatus              ResourceType               LogicalResourceId
 ResourceStatusReason
-----------------------------------------------------------------------------------------------
CREATE_IN_PROGRESS          AWS::IAM::Role             HelloWorldFunctionRole     -

CREATE_IN_PROGRESS          AWS::IAM::Role             HelloWorldFunctionRole
 Resource creation
```

```
...
------------------------------------------------------------------------------------------------

CloudFormation outputs from deployed stack
------------------------------------------------------------------------------------------------
Outputs

------------------------------------------------------------------------------------------------
Key                 HelloWorldFunctionIamRole

Description         Implicit IAM Role created for Hello World function

Value               arn:aws:iam::012345678910:role/hello-rust-
HelloWorldFunctionRole-10II2P13AUDUY

Key                 HelloWorldApi

Description         API Gateway endpoint URL for Prod stage for Hello World function

Value               https://ggdxec9le9.execute-api.us-west-2.amazonaws.com/Prod/hello/


Key                 HelloWorldFunction

Description         Hello World Lambda Function ARN

Value               arn:aws:lambda:us-west-2:012345678910:function:hello-rust-
HelloWorldFunction-
yk4HzGzYeZBj

------------------------------------------------------------------------------------------------


Successfully created/updated stack - hello-rust in us-west-2
```

Zum Testen können wir unsere Lambda-Funktion über den API-Endpunkt aufrufen.

```
$ curl https://ggdxec9le9.execute-api.us-west-2.amazonaws.com/Prod/hello/
Hello World!%
```

Um unsere Funktion lokal zu testen, stellen wir zunächst sicher, dass die `Architectures` Eigenschaft unserer Funktion mit unserem lokalen Computer übereinstimmt.

```
...
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function # More info about Function Resource:
 https://github.com/awslabs/serverless-application-model/blob/master/
versions/2016-10-31.md#awsserverlessfunction
    Metadata:
      BuildMethod: rust-cargolambda # More info about Cargo Lambda: https://github.com/
cargo-lambda/cargo-lambda
    Properties:
      CodeUri: ./rust_app   # Points to dir of Cargo.toml
      Handler: bootstrap    # Do not change, as this is the default executable name
 produced by Cargo Lambda
      Runtime: provided.al2023
      Architectures:
        - arm64
...
```

Da wir `arm64` in diesem Beispiel unsere Architektur von `x86_64` auf geändert haben, laufen wir los, `sam build` um unsere Build-Artefakte zu aktualisieren. Dann laufen wir los`sam local invoke`, um unsere Funktion lokal aufzurufen.

```
hello-rust$ sam local invoke
Invoking bootstrap (provided.al2023)
Local image was not found.
Removing rapid images for repo public.ecr.aws/sam/emulation-provided.al2023
Building
 image.....................................................................................
Using local image: public.ecr.aws/lambda/provided:al2023-rapid-arm64.

Mounting /Users/.../hello-rust/.aws-sam/build/HelloWorldFunction as /var/
task:ro,delegated, inside runtime container
START RequestId: fbc55e6e-0068-45f9-9f01-8e2276597fc6 Version: $LATEST
{"statusCode":200,"body":"Hello World!"}END RequestId:
 fbc55e6e-0068-45f9-9f01-8e2276597fc6
REPORT RequestId: fbc55e6e-0068-45f9-9f01-8e2276597fc6  Init Duration: 0.68 ms
 Duration: 130.63 ms     Billed Duration: 131 ms     Memory Size: 128 MB     Max Memory
 Used: 128 MB
```

Einzelnes Lambda-Funktionsprojekt

Hier ist ein Beispiel für eine serverlose Anwendung, die eine Rust-Lambda-Funktion enthält.

Struktur des Projektverzeichnisses:

```
.
### Cargo.lock
### Cargo.toml
### src
#   ### main.rs
### template.yaml
```

AWS SAM Vorlage:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Metadata:
      BuildMethod: rust-cargolambda
    Properties:
      CodeUri: ./
      Handler: bootstrap
      Runtime: provided.al2023
...
```

Projekt mit mehreren Lambda-Funktionen

Hier ist ein Beispiel für eine serverlose Anwendung, die mehrere Rust Lambda-Funktionen enthält und als Cargo Arbeitsbereich organisiert ist.

Wir empfehlen einen Cargo Arbeitsbereich für Anwendungen mit mehreren Rust Lambda-Funktionen. Jede Funktion ist ein eigenes Paket, sodass Funktionen unabhängige Abhängigkeiten deklarieren können und gleichzeitig gemeinsamen Code über ein Bibliothekspaket gemeinsam nutzen. Jedes Paket erzeugt eine einzelne Binärdatei, die nach dem Paket benannt ist, sodass Sie die `Binary` Build-Eigenschaft nicht festlegen müssen.

Struktur des Projektverzeichnisses:

```
.
### Cargo.lock
### Cargo.toml
```

```
### function_a
#   ### Cargo.toml
#   ### src
#       ### main.rs
### function_b
#   ### Cargo.toml
#   ### src
#       ### main.rs
### template.yaml
```

`Cargo.toml`Workspace-Datei, im Stammverzeichnis des Projekts:

```
[workspace]
resolver = "2"
members = [
    "function_a",
    "function_b",
]

[workspace.dependencies]
lambda_runtime = "0.13"
serde = { version = "1", features = ["derive"] }
tokio = { version = "1", features = ["macros", "rt"] }
```

`Cargo.toml`Datei für jede Funktion, z. B.`function_a/Cargo.toml`:

```
[package]
name = "function_a"
version = "0.1.0"
edition = "2021"

[dependencies]
lambda_runtime = { workspace = true }
serde = { workspace = true }
tokio = { workspace = true }
```

AWS SAM Vorlage. Die `CodeUri` jeder Funktion zeigt auf das Paketverzeichnis dieser Funktion:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
```

```
  FunctionA:
    Type: AWS::Serverless::Function
    Metadata:
      BuildMethod: rust-cargolambda
    Properties:
      CodeUri: ./function_a
      Handler: bootstrap
      Runtime: provided.al2023
  FunctionB:
    Type: AWS::Serverless::Function
    Metadata:
      BuildMethod: rust-cargolambda
    Properties:
      CodeUri: ./function_b
      Handler: bootstrap
      Runtime: provided.al2023
```

> **Note**
>
> Die AWS SAMCLI baut jede Funktion im Workspace in das gemeinsame `target` Verzeichnis des Workspace auf, Cargo kompiliert also gemeinsame Abhängigkeiten einmal statt einmal für jede Funktion. Dieses Verhalten erfordert AWS SAMCLI Version 1.165.0 oder höher. In früheren Versionen wurde jede Funktion in einem eigenen `target` Verzeichnis erstellt und der vollständige Abhängigkeitsbaum wird für jede Funktion neu kompiliert, wodurch Builds langsamer werden, wenn Sie Funktionen hinzufügen.

Geben Sie jedem Funktionspaket einen eindeutigen Binärnamen. Paketnamen sind innerhalb eines Arbeitsbereichs eindeutig, sodass der Standard-Binärname bereits eindeutig ist. Wenn Sie den Binärnamen mit einem `[[bin]]` Abschnitt überschreiben, geben Sie nicht zwei Paketen denselben Binärnamen. Sie kompilieren auf denselben Pfad im gemeinsam genutzten `target` Verzeichnis und überschreiben sich gegenseitig. Das AWS SAMCLI protokolliert eine Warnung, wenn es dies feststellt.

Alternativ kann ein einzelnes Paket mehrere Binärdateien definieren. Verwenden Sie in diesem Fall die Eigenschaft `Binary` build, um die Binärdatei für jede Funktion auszuwählen:

```
Resources:
  FunctionA:
    Type: AWS::Serverless::Function
```

```
    Metadata:
      BuildMethod: rust-cargolambda
      BuildProperties:
        Binary: function_a
    Properties:
      CodeUri: ./
      Handler: bootstrap
      Runtime: provided.al2023
```

Die Optimierung von Rust-Builds ist eingebaut GitHub Aktionen

Rust-Builds sind rechenintensiv, und ein Continuous Integration Runner beginnt ohne kompilierte Artefakte. Anwendungen mit mehreren Funktionen, die große Abhängigkeiten gemeinsam haben, wie z. B. an AWSSDK, können die meiste Zeit ihrer Build-Zeit damit verbringen, dieselben Abhängigkeiten zu kompilieren. Die folgenden Methoden reduzieren die Build-Zeit inGitHub Actions.

Verwenden Sie AWS SAMCLI Version 1.165.0 oder höher für Arbeitsbereiche

Version 1.165.0 und höher bauen jedes Mitglied eines Cargo Workspace in das gemeinsame `target` Verzeichnis des Workspace ein, sodass gemeinsame Abhängigkeiten einmal pro Build kompiliert werden und nicht einmal für jede Funktion. Geben Sie bei der Installation von die Mindestversion an, AWS SAMCLI damit ein Build nicht stillschweigend auf das langsamere Verhalten zurückfällt.

Zwischenspeichern Sie die Cargo Registrierung und **`target`** das Verzeichnis

Zwischenspeichern Sie die Cargo Registrierung (`~/.cargo/registry`und`~/.cargo/git/db`) und das `target` Workspace-Verzeichnis zwischen den Ausführungen, sodass unveränderte Abhängigkeiten wiederhergestellt und nicht neu kompiliert werden. Verwenden Sie für jedes Kompilierungsziel einen separaten Cache. Ein Job, für den Release-Artefakte übergreifend kompiliert werden, `arm64` erzeugt andere Artefakte als ein Job, für den nativ kompiliert wird. Ein gemeinsam genutzter `x86_64` Cache passt also nie zusammen.

Fügen Sie Build-Einstellungen in den Cache-Schlüssel ein

Cargoschließt Einstellungen wie `opt-level` und `codegen-units` in den Fingerabdruck ein, anhand dessen entschieden wird, ob ein kompiliertes Artefakt wiederverwendet werden kann. Wenn Sie den `[profile.release]` Abschnitt Ihrer `Cargo.toml` Workspace-Datei ändern, ohne den Cache-Schlüssel zu ändern, wird der Cache wiederhergestellt, aber trotzdem wird jede Crate neu kompiliert. Füge einen Hash der `Cargo.toml` Workspace-Datei in den Cache-Schlüssel ein, sodass beim Ändern einer Profileinstellung ein neuer Cache gestartet wird.

Übermitteln Sie Ihre **`Cargo.lock`** Datei

Lambda-Funktionen sind ausführbar, also übergeben Sie Ihre Datei. `Cargo.lock` Dadurch erhalten Sie reproduzierbare Builds und einen stabilen Cache-Schlüssel, der sich nur ändert, wenn sich Ihre Abhängigkeiten ändern.

Passen Sie das Release-Profil an die Build-Zeit und den Kaltstart an

Ihr Funktionscode wird bei jedem Lauf neu kompiliert, da er sich häufiger ändert als Ihre Abhängigkeiten. Das Standard-Release-Profil optimiert den Laufzeitdurchsatz, den viele Lambda-Funktionen nicht benötigen. Durch die Optimierung der Größe werden kleinere Binärdateien erzeugt, was auch die Kaltstartzeit erleichtert, und eine Erhöhung der Anzahl der Codegenerierungseinheiten erhöht die Parallelität bei der Kompilierung. Lassen Sie die Linkzeitoptimierung (`lto`) deaktiviert, da dies die Kompilierung verlangsamt. Füge deiner `Cargo.toml` Workspace-Datei Folgendes hinzu:

```
[profile.release]
opt-level = "s"
codegen-units = 256
lto = false
strip = true
```

Messen Sie den Effekt in Ihrer eigenen Anwendung. Bei diesen Einstellungen wird ein kleiner Teil der Laufzeitleistung gegen die Build-Zeit und die Binärgröße eingetauscht.

Vermeiden Sie doppelte Workflow-Ausführungen

Ein Workflow, der `push` sowohl für Ereignisse als auch für `pull_request` Ereignisse ausgeführt wird, wird zweimal für denselben Commit ausgeführt. GitHub ActionsCaches sind nach Branch- und Pull-Requests aufgeteilt, sodass die beiden Läufe in unterschiedliche Cache-Bereiche schreiben und keiner der beiden den Cache des anderen wiederverwendet. Verwenden Sie eine Parallelitätsgruppe, die im Head-Commit eingegeben wird, sodass nur ein Durchlauf jeden Commit erstellt.

Der folgende Workflow erstellt einen Cargo Arbeitsbereich mit Rust Lambda-Funktionen für die oben genannten `arm64` Praktiken und wendet sie an:

```
name: Build

on:
```

```
  push:
    branches: [main]
  pull_request:

# Collapse the push and pull_request runs for the same commit into a single run.
concurrency:
  group: ${{ github.workflow }}-${{ github.event.pull_request.head.sha || github.sha }}
  cancel-in-progress: true

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v5

      - uses: dtolnay/rust-toolchain@stable
        with:
          targets: aarch64-unknown-linux-gnu

      # Cache the Cargo registry and the workspace target directory. The key covers
      # the compilation target, Cargo.lock, and the workspace Cargo.toml, so that
      # changing a dependency or a release profile setting starts a new cache
      # instead of restoring one whose artifacts Cargo discards.
      - uses: actions/cache@v4
        with:
          path: |
            ~/.cargo/registry/index
            ~/.cargo/registry/cache
            ~/.cargo/git/db
            target
          key: cargo-arm64-${{ hashFiles('Cargo.lock', 'Cargo.toml') }}
          restore-keys: |
            cargo-arm64-

      - name: Install build tools
        run: pip install cargo-lambda 'aws-sam-cli>=1.165.0'

      - name: Build
        run: sam build
```

Der `restore-keys` Eintrag ermöglicht es, einen Lauf mit dem neuesten Cache zu starten, wenn der Schlüssel nicht genau übereinstimmt, sodass bei einer Änderung der Abhängigkeit die Crates, die sich nicht geändert haben, wiederverwendet werden, anstatt alles erneut zu kompilieren.

Python-Lambda-Funktionen mit uv in erstellen AWS SAM

Diese Funktion befindet sich in der Vorschauversion für AWS SAM und kann sich ändern.

Verwenden Sie die AWS Serverless Application Model Befehlszeilenschnittstelle (AWS SAMCLI) mituv, einem schnellen Installationsprogramm und Resolver für Python-Pakete, um Ihre AWS Lambda Python-Funktionen zu erstellen.

Topics

- Voraussetzungen
- Konfiguration AWS SAM für die Verwendung mit Python-Lambda-Funktionen und uv
- Beispiele

Voraussetzungen

Python

Informationen zur Installation von Python finden Sie unter Python herunterladen auf der Python-Website.

uv

Das AWS SAMCLI erfordert die Installation von uv, einem extrem schnellen Python-Paket-Installer und Resolver. Installationsanweisungen finden Sie in der uvDokumentation unter Installation.

Melden Sie sich für die AWS SAMCLI Beta-Funktion an

Da es sich bei dieser Funktion um eine Vorschauversion handelt, müssen Sie sich mit einer der folgenden Methoden anmelden:

1. Verwenden Sie die Umgebungsvariable:`SAM_CLI_BETA_PYTHON_UV=1`.
2. Fügen Sie der `samconfig.toml`-Datei Folgendes hinzu:

```
[default.build.parameters]
beta_features = true
[default.sync.parameters]
beta_features = true
```

3. Verwenden Sie die `--beta-features` Option, wenn Sie einen unterstützten AWS SAMCLI Befehl verwenden. Beispiel:

```
$ sam build --beta-features
```

4. Wählen Sie die Optiony, wenn Sie AWS SAMCLI aufgefordert werden, sich anzumelden. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam build
Starting Build use cache
Build method "python-uv" is a beta feature.
Please confirm if you would like to proceed
You can also enable this beta feature with "sam build --beta-features". [y/N]: y
```

Konfiguration AWS SAM für die Verwendung mit Python-Lambda-Funktionen und uv

Schritt 1: Konfiguriere deine AWS SAM Vorlage

Konfigurieren Sie Ihre AWS SAM Vorlage wie folgt:

- BuildMethod – `python-uv`.
- CodeUri— Pfad zu Ihrem Funktionscode-Verzeichnis, das `pyproject.toml` oder enthält`requirements.txt`.
- Handler — Ihr Funktionshandler (z. B.`app.lambda_handler`).
- Runtime — Python-Laufzeitversion (z. B.`python3.12`).

Hier ist ein Beispiel für eine konfigurierte AWS SAM Vorlage:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: ./my_function
      Handler: app.lambda_handler
      Runtime: python3.12
    Metadata:
      BuildMethod: python-uv
...
```

Beispiele

Beispiel Hello World

In diesem Beispiel erstellen wir eine Hello World-Beispielanwendung mit Python uv als Paketmanager.

uvkann entweder `pyproject.toml` oder verwenden`requirements.txt`, um Abhängigkeiten zu lesen. Wenn beide angegeben sind, `sam build` wird aus `requirements.txt` den Abhängigkeiten gelesen.

Das Folgende ist die Struktur unserer Hello World-Anwendung:

```
hello-python-uv
### README.md
### events
#   ### event.json
### hello_world
#   ### __init__.py
#   ### app.py
#   ### pyproject.toml
### samconfig.toml
### template.yaml
```

`pyproject.toml`-Datei:

```
[project]
name = "my-function"
version = "0.1.0"
requires-python = ">=3.12"
dependencies = [
    "requests>=2.31.0",
    "boto3>=1.28.0",
]
```

In unserer AWS SAM Vorlage ist unsere Python-Funktion wie folgt definiert:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
```

```
    Properties:
      CodeUri: hello_world/
      Handler: app.lambda_handler
      Runtime: python3.12
      Architectures:
        - x86_64
    Metadata:
      BuildMethod: python-uv
```

Als Nächstes erstellen wir `sam build` unsere Anwendung und bereiten sie auf die Bereitstellung vor. Das AWS SAMCLI erstellt ein `.aws-sam` Verzeichnis und organisiert dort unsere Build-Artefakte. Unsere Funktionsabhängigkeiten werden unter installiert uv und gespeichert. `.aws-sam/build/HelloWorldFunction/`

```
hello-python-uv$ sam build
Starting Build use cache
Build method "python-uv" is a beta feature.
Please confirm if you would like to proceed
You can also enable this beta feature with "sam build --beta-features". [y/N]: y

Experimental features are enabled for this session.
Visit the docs page to learn more about the AWS Beta terms https://aws.amazon.com/
service-terms/.

Cache is invalid, running build and copying resources for following functions
 (HelloWorldFunction)
Building codeuri: /Users/.../hello-python-uv/hello_world runtime: python3.12 metadata:
 {'BuildMethod': 'python-uv'} architecture: x86_64 functions: HelloWorldFunction
Running PythonUvBuilder:UvBuild
Running PythonUvBuilder:CopySource

Build Succeeded

Built Artifacts  : .aws-sam/build
Built Template   : .aws-sam/build/template.yaml

Commands you can use next
=========================
[*] Validate SAM template: sam validate
[*] Invoke Function: sam local invoke
[*] Test Function in the Cloud: sam sync --stack-name {{stack-name}} --watch
[*] Deploy: sam deploy --guided
```

> **Note**
>
> Die `python-uv` Build-Methode wird pro Funktion im `Metadata` Abschnitt konfiguriert. Jede Funktion in Ihrer Vorlage kann eine andere Build-Methode verwenden, sodass Sie uv basierte Funktionen mit `pip` basierten Funktionen in derselben AWS SAM Vorlage kombinieren können. Wenn keine Build-Methode angegeben `pip` ist, wird standardmäßig verwendet.

Erstellen von Lambda-Funktionen mit benutzerdefinierten Laufzeiten in AWS SAM

Sie können den `gleicher Build` Befehl verwenden, um benutzerdefinierte Laufzeiten zu erstellen, die für Ihre Lambda-Funktion erforderlich sind. Sie deklarieren Ihre Lambda-Funktion so, dass sie eine benutzerdefinierte Laufzeit verwendet, indem Sie `Runtime: provided` für die Funktion angeben.

Um eine benutzerdefinierte Laufzeit zu erstellen, deklarieren Sie das `Metadata` Ressourcenattribut mit einem `BuildMethod: makefile` Eintrag. Sie stellen ein benutzerdefiniertes Makefile bereit, in dem Sie ein Build-Ziel in der Form deklarieren`build-`*`function-logical-id`*, das die Build-Befehle für Ihre Laufzeit enthält. Ihr Makefile ist dafür verantwortlich, falls erforderlich, die benutzerdefinierte Runtime zu kompilieren und die Build-Artefakte an den richtigen Ort zu kopieren, der für nachfolgende Schritte in Ihrem Workflow erforderlich ist. Der Speicherort des Makefiles wird durch die `CodeUri` Eigenschaft der Funktionsressource angegeben und muss benannt werden. `Makefile`

Beispiele

Beispiel 1: Benutzerdefinierte Laufzeit für eine in Rust geschriebene Funktion

> **Note**
>
> Wir empfehlen, Lambda-Funktionen mit Cargo Lambda zu erstellen. Weitere Informationen hierzu finden Sie unter Erstellen von Rust Lambda-Funktionen mit Cargo Lambda in AWS SAM.

Die folgende AWS SAM Vorlage deklariert eine Funktion, die eine benutzerdefinierte Laufzeit für eine in Rust geschriebene Lambda-Funktion verwendet, und weist `sam build` an, die Befehle für das `build-HelloRustFunction` Build-Ziel auszuführen.

```
Resources:
  HelloRustFunction:
    Type: AWS::Serverless::Function
    Properties:
      FunctionName: HelloRust
      Handler: bootstrap.is.real.handler
      Runtime: provided
      MemorySize: 512
      CodeUri: .
    Metadata:
      BuildMethod: makefile
```

Das folgende Makefile enthält das Build-Ziel und die Befehle, die ausgeführt werden. Beachten Sie, dass die `CodeUri` Eigenschaft auf gesetzt ist`.`, sodass sich das Makefile im Stammverzeichnis des Projekts befinden muss (d. h. im selben Verzeichnis wie die AWS SAM Vorlagendatei der Anwendung). Der Dateiname muss sein. `Makefile`

```
build-HelloRustFunction:
 cargo build --release --target x86_64-unknown-linux-musl
 cp ./target/x86_64-unknown-linux-musl/release/bootstrap $(ARTIFACTS_DIR)
```

Weitere Informationen zum Einrichten Ihrer Entwicklungsumgebung zur Ausführung des `cargo build` Befehls aus dem vorherigen `makefile` Abschnitt finden Sie im AWS Lambda Blogbeitrag Rust Runtime for.

Beispiel 2: Makefile-Builder für Python 3.12 (Alternative zur Verwendung des mitgelieferten Builders)

Möglicherweise möchten Sie eine Bibliothek oder ein Modul verwenden, das nicht in einem gebündelten Builder enthalten ist. Dieses Beispiel zeigt eine AWS SAM Vorlage für eine Python3.12-Laufzeit mit einem Makefile-Builder.

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: hello_world/
      Handler: app.lambda_handler
      Runtime: python3.12
    Metadata:
      BuildMethod: makefile
```

Das folgende Makefile enthält das Build-Ziel und die Befehle, die ausgeführt werden. Beachten Sie, dass die `CodeUri` Eigenschaft auf `hello_world` gesetzt ist. Das Makefile muss sich also im Stammverzeichnis des `hello_world` Unterverzeichnisses befinden, und der Dateiname muss sein. `Makefile`

```
build-HelloWorldFunction:
 cp *.py $(ARTIFACTS_DIR)
 cp requirements.txt $(ARTIFACTS_DIR)
 python -m pip install -r requirements.txt -t $(ARTIFACTS_DIR)
 rm -rf $(ARTIFACTS_DIR)/bin
```

Aufbau von Lambda-Schichten in AWS SAM

Sie können sie verwenden AWS SAM , um benutzerdefinierte Lambda-Schichten zu erstellen. Mit Lambda-Schichten können Sie Code aus einer Lambda-Funktion extrahieren, der dann für mehrere Lambda-Funktionen wiederverwendet werden kann. Wenn Sie nur Lambda-Schichten erstellen (anstatt Ihre gesamte Anwendung zu erstellen), können Sie in mehrfacher Hinsicht davon profitieren. Es kann Ihnen helfen, die Größe Ihrer Bereitstellungspakete zu reduzieren, die Logik der Kernfunktionen von Abhängigkeiten zu trennen und Abhängigkeiten für mehrere Funktionen gemeinsam zu nutzen. Informationen zu Layern finden Sie unter AWS Lambda-Schichten im AWS Lambda Developer Guide.

So erstellen Sie eine Lambda-Schicht in AWS SAM

> **Note**
>
> Bevor Sie eine Lambda-Schicht erstellen können, müssen Sie zunächst eine Lambda-Schicht in Ihre AWS SAM Vorlage schreiben. Informationen und Beispiele dazu finden Sie unter. Steigern Sie die Effizienz mithilfe von Lambda-Schichten mit AWS SAM

Um eine benutzerdefinierte Ebene zu erstellen, deklarieren Sie sie in Ihrer Vorlagendatei AWS Serverless Application Model (AWS SAM) und fügen Sie einen Abschnitt mit einem `Metadata` Ressourcenattribut mit einem `BuildMethod` Eintrag hinzu. Gültige Werte für `BuildMethod` sind Bezeichner für eine AWS Lambda Laufzeit oder`makefile`. Fügen Sie einen `BuildArchitecture` Eintrag hinzu, um die Befehlssatzarchitekturen anzugeben, die Ihr Layer unterstützt. Gültige Werte für `BuildArchitecture` sind Lambda-Befehlssatzarchitekturen.

Wenn Sie angeben`makefile`, geben Sie das benutzerdefinierte Makefile an, in dem Sie ein Build-Ziel in der Form deklarierenbuild-*`layer-logical-id`*, das die Build-Befehle für Ihre Ebene enthält. Ihr Makefile ist dafür verantwortlich, die Ebene bei Bedarf zu kompilieren und die Build-Artefakte an den richtigen Ort zu kopieren, der für nachfolgende Schritte in Ihrem Workflow erforderlich ist. Der Speicherort des Makefiles wird durch die `ContentUri` Eigenschaft der Layer-Ressource angegeben und muss benannt werden. `Makefile`

> **Note**
>
> Wann Sie eine benutzerdefinierte Ebene erstellen, AWS Lambda hängt von Umgebungsvariablen ab, um Ihren Layer-Code zu finden. Lambda-Laufzeiten enthalten Pfade in dem `/opt` Verzeichnis, in das Ihr Layer-Code kopiert wird. Die Ordnerstruktur für Build-Artefakte Ihres Projekts muss mit der erwarteten Ordnerstruktur der Laufzeit übereinstimmen, damit Ihr benutzerdefinierter Layer-Code gefunden werden kann.
> Für Python können Sie beispielsweise Ihren Code im `python/` Unterverzeichnis platzieren. Für NodeJS können Sie Ihren Code im Unterverzeichnis platzieren. `nodejs/node_modules/`
> Weitere Informationen finden Sie im Entwicklerhandbuch unter Einbeziehen von Bibliotheksabhängigkeiten in eine Ebene.AWS Lambda

Im Folgenden finden Sie ein Beispiel für einen Abschnitt mit `Metadata` Ressourcenattributen.

```
Metadata:
  BuildMethod: python3.12
  BuildArchitecture: arm64
```

> **Note**
>
> Wenn Sie den Abschnitt mit den `Metadata` Ressourcenattributen nicht einbeziehen, AWS SAM wird der Layer nicht erstellt. Stattdessen werden die Build-Artefakte von dem Speicherort kopiert, der in der `CodeUri` Eigenschaft der Layer-Ressource angegeben ist. Weitere Informationen finden Sie unter der ContentUriEigenschaft des `AWS::Serverless::LayerVersion` Ressourcentyps.

Wenn Sie den Abschnitt mit den `Metadata` Ressourcenattributen einbeziehen, können Sie den `gleicher Build` Befehl verwenden, um den Layer sowohl als eigenständiges Objekt als auch als Abhängigkeit von einer AWS Lambda Funktion zu erstellen.

- Als unabhängiges Objekt. Möglicherweise möchten Sie nur das Layer-Objekt erstellen, wenn Sie beispielsweise lokal eine Codeänderung an der Ebene testen und nicht Ihre gesamte Anwendung erstellen müssen. Um die Ebene unabhängig zu erstellen, geben Sie die Layer-Ressource mit dem `sam build` *`layer-logical-id`* Befehl an.
- Als Abhängigkeit einer Lambda-Funktion. Wenn Sie die logische ID einer Ebene in die `Layers` Eigenschaft einer Lambda-Funktion in derselben AWS SAM Vorlagendatei aufnehmen, ist die Ebene eine Abhängigkeit von dieser Lambda-Funktion. Wenn diese Ebene auch einen Abschnitt mit einem `Metadata` Ressourcenattribut mit einem `BuildMethod` Eintrag enthält, erstellen Sie die Ebene, indem Sie entweder die gesamte Anwendung mit dem `sam build` Befehl erstellen oder indem Sie die Funktionsressource mit dem `sam build` *`function-logical-id`* Befehl angeben.

Beispiele

Vorlagenbeispiel 1: Erstellen Sie eine Ebene für die Python 3.12-Laufzeitumgebung

Die folgende AWS SAM Beispielvorlage erstellt eine Ebene für die Python 3.12-Laufzeitumgebung.

```
Resources:
  MyLayer:
    Type: AWS::Serverless::LayerVersion
    Properties:
      ContentUri: my_layer
      CompatibleRuntimes:
        - python3.12
    Metadata:
      BuildMethod: python3.12   # Required to have AWS SAM build this layer
```

Vorlagenbeispiel 2: Erstellen Sie eine Ebene mit einem benutzerdefinierten Makefile

Die folgende AWS SAM Beispielvorlage verwendet eine benutzerdefinierte Vorlage`makefile`, um die Ebene zu erstellen.

```
Resources:
  MyLayer:
```

```
    Type: AWS::Serverless::LayerVersion
    Properties:
      ContentUri: my_layer
      CompatibleRuntimes:
        - python3.12
    Metadata:
      BuildMethod: makefile
```

Das Folgende `makefile` enthält das Build-Ziel und die Befehle, die ausgeführt werden. Beachten Sie, dass die `ContentUri` Eigenschaft auf `my_layer` gesetzt ist. Das Makefile muss sich also im Stammverzeichnis des `my_layer` Unterverzeichnisses befinden, und der Dateiname muss sein. `Makefile` Beachten Sie auch, dass die Build-Artefakte in das `python/` Unterverzeichnis kopiert AWS Lambda werden, damit der Layer-Code gefunden werden kann.

```
build-MyLayer:
  mkdir -p "$(ARTIFACTS_DIR)/python"
  cp *.py "$(ARTIFACTS_DIR)/python"
  python -m pip install -r requirements.txt -t "$(ARTIFACTS_DIR)/python"
```

> **Note**
>
> Wenn der aufgerufen `makefile` wird, wird das entsprechende Ziel ausgelöst und die Artefakte sollten in die exponierte Umgebungsvariable `$ARTIFACTS_DIR` kopiert werden. Weitere Informationen finden Sie aws-lambda-builders unter GitHub.

Beispiel für Sam Build-Befehle

Mit den folgenden `sam build` Befehlen werden Ebenen erstellt, die die Abschnitte mit den `Metadata` Ressourcenattributen enthalten.

```
# Build the 'layer-logical-id' resource independently
$ sam build layer-logical-id

# Build the 'function-logical-id' resource and layers that this function depends on
$ sam build function-logical-id

# Build the entire application, including the layers that any function depends on
$ sam build
```

Testen Sie Ihre serverlose Anwendung mit AWS SAM

Nachdem Sie Ihre Anwendung geschrieben und erstellt haben, können Sie Ihre Anwendung testen, um sicherzustellen, dass sie ordnungsgemäß funktioniert. Mit der AWS SAM Befehlszeilenschnittstelle (CLI) können Sie Ihre serverlose Anwendung lokal testen, bevor Sie sie in die AWS Cloud hochladen. Durch das Testen Ihrer Anwendung können Sie die Funktionalität, Zuverlässigkeit und Leistung der Anwendung überprüfen und gleichzeitig Probleme (Bugs) identifizieren, die behoben werden müssen.

Dieser Abschnitt enthält Anleitungen zu gängigen Methoden, die Sie beim Testen Ihrer Anwendung anwenden können. Die Themen in diesem Abschnitt konzentrieren sich hauptsächlich auf die lokalen Tests, die Sie vor der Bereitstellung in der AWS Cloud durchführen können. Das Testen vor der Bereitstellung hilft Ihnen dabei, Probleme proaktiv zu identifizieren und so unnötige Kosten im Zusammenhang mit Bereitstellungsproblemen zu reduzieren. Jedes Thema in diesem Abschnitt beschreibt einen Test, den Sie durchführen können, und enthält Beispiele, die Ihnen zeigen, wie Sie den Test durchführen können. Nachdem Sie Ihre Anwendung getestet haben, sind Sie bereit, alle gefundenen Probleme zu debuggen.

Themen

- Einführung in das Testen mit dem sam local Befehl
- Lokal Lambda-Funktionen aufrufen mit AWS SAM
- Lokal ausgeführtes API Gateway mit AWS SAM
- Einführung in Cloud-Tests mit sam remote test-event
- Einführung in das Testen in der Cloud mit sam remote invoke
- Automatisieren Sie lokale Integrationstests mit AWS SAM
- Generieren Sie Beispielereignis-Payloads mit AWS SAM
- Testen und Debuggen langlebiger Funktionen

Einführung in das Testen mit dem sam local Befehl

Verwenden Sie den AWS Serverless Application Model Befehl Command Line Interface (AWS SAMCLI)`sam local`, um Ihre serverlosen Anwendungen lokal zu testen.

Eine Einführung in das finden Sie AWS SAMCLI unterWas ist der AWS SAM CLI?.

Voraussetzungen

Um das zu verwendensam local, installieren Sie das, AWS SAMCLI indem Sie die folgenden Schritte ausführen:

- AWS SAM Voraussetzungen.
- Installieren Sie das AWS SAMCLI.

Vor der Verwendung empfehlen wirsam local, sich mit folgenden Grundkenntnissen vertraut zu machen:

- Konfiguration der AWS SAMCLI.
- Erstellen Sie Ihre Bewerbung in AWS SAM.
- Einführung in das Bauen mit AWS SAM.
- Einführung in die Bereitstellung mit AWS SAM.

Verwenden des sam local Befehls

Verwenden Sie den sam local Befehl zusammen mit einem seiner Unterbefehle, um verschiedene Arten von lokalen Tests für Ihre Anwendung durchzuführen.

```
$ sam local <subcommand>
```

Weitere Informationen zu den einzelnen Unterbefehlen finden Sie im Folgenden:

- Einführung in sam local generate-event— Generiert AWS-Service Ereignisse für lokale Tests.
- Einführung in sam local invoke— Initiiert einen einmaligen Aufruf einer AWS Lambda Funktion lokal.
- Einführung in sam local start-api— Führen Sie Ihre Lambda-Funktionen über einen lokalen HTTP-Server aus.
- Einführung in sam local start-lambda— Führen Sie Ihre Lambda-Funktionen über einen lokalen HTTP-Server zur Verwendung mit dem AWS CLI oder SDKs aus.

Einführung in das Testen mit sam local generate-event

Verwenden Sie den AWS Serverless Application Model `sam local generate-event` Unterbefehl Command Line Interface (AWS SAMCLI), um Beispiele für Nutzdaten von Ereignissen für unterstützte Ereignisse zu generieren. AWS-Services Sie können diese Ereignisse dann ändern und zum Testen an lokale Ressourcen weitergeben.

- Eine Einführung in das finden AWS SAMCLI Sie unter Was ist der AWS SAM CLI?
- Eine Liste der `sam local generate-event` Befehlsoptionen finden Sie untersam local generate-event.

Ein Ereignis ist ein JSON-Objekt, das generiert wird, wenn ein eine Aktion oder Aufgabe AWS-Service ausführt. Diese Ereignisse enthalten spezifische Informationen, z. B. die verarbeiteten Daten oder den Zeitstempel des Ereignisses. Die meisten AWS-Services generieren Ereignisse, und die Ereignisse jedes Dienstes sind für den jeweiligen Dienst eindeutig formatiert.

Von einem Dienst generierte Ereignisse werden als Ereignisquelle an andere Dienste weitergegeben. Beispielsweise kann ein Artikel, der in einem Amazon Simple Storage Service (Amazon S3) -Bucket platziert wird, ein Ereignis auslösen. Dieses Ereignis kann dann als Ereignisquelle für eine AWS Lambda Funktion zur weiteren Verarbeitung der Daten verwendet werden.

Ereignisse, die Sie mit generieren, `sam local generate-event` werden in derselben Struktur formatiert wie die tatsächlichen Ereignisse, die vom AWS Service erstellt wurden. Sie können den Inhalt dieser Ereignisse ändern und sie zum Testen von Ressourcen in Ihrer Anwendung verwenden.

Voraussetzungen

Um das zu verwendensam local generate-event, installieren Sie das, AWS SAMCLI indem Sie die folgenden Schritte ausführen:

- AWS SAM Voraussetzungen.
- Installieren Sie das AWS SAMCLI.

Vor der Verwendung empfehlen wirsam local generate-event, sich mit folgenden Grundkenntnissen vertraut zu machen:

- Konfiguration der AWS SAMCLI.
- Erstellen Sie Ihre Bewerbung in AWS SAM.

- Einführung in das Bauen mit AWS SAM.
- Einführung in die Bereitstellung mit AWS SAM.

Generieren Sie Beispielereignisse

Verwenden Sie den AWS SAMCLI `sam local generate-event` Unterbefehl, um Ereignisse für unterstützte AWS-Services zu generieren.

Um eine Liste der unterstützten zu sehen AWS-Services

1. Führen Sie Folgendes aus:

```
$ sam local generate-event
```

2. Die Liste der unterstützten AWS-Services wird angezeigt. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam local generate-event
...
Commands:
  alb
  alexa-skills-kit
  alexa-smart-home
  apigateway
  appsync
  batch
  cloudformation
  ...
```

Um ein lokales Ereignis zu generieren

1. Führen Sie den unterstützten Dienstnamen aus `sam local generate-event` und geben Sie ihn an. Daraufhin wird eine Liste der Ereignistypen angezeigt, die Sie generieren können. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam local generate-event s3

Usage: sam local generate-event s3 [OPTIONS] COMMAND [ARGS]...

Options:
```

```
  -h, --help  Show this message and exit.

Commands:
  batch-invocation  Generates an Amazon S3 Batch Operations Invocation Event
  delete            Generates an Amazon S3 Delete Event
  put               Generates an Amazon S3 Put Event
```

2. Um das Beispielereignis zu generieren`sam local generate-event`, führen Sie es aus und geben Sie den Dienst und den Ereignistyp an.

```
$ sam local generate-event <service> <event>
```

Im Folgenden wird ein Beispiel gezeigt:

```
$ sam local generate-event s3 put
{
  "Records": [
    {
      "eventVersion": "2.0",
      "eventSource": "aws:s3",
      "awsRegion": "us-east-1",
      "eventTime": "1970-01-01T00:00:00.000Z",
      "eventName": "ObjectCreated:Put",
      "userIdentity": {
        "principalId": "EXAMPLE"
      },
      "requestParameters": {
        "sourceIPAddress": "127.0.0.1"
      },
      "responseElements": {
        "x-amz-request-id": "EXAMPLE123456789",
        "x-amz-id-2": "EXAMPLE123/5678abcdefghijklambdaisawesome/
mnopqrstuvwxyzABCDEFGH"
      },
      "s3": {
        "s3SchemaVersion": "1.0",
        "configurationId": "testConfigRule",
        "bucket": {
          "name": "sam-s3-demo-bucket",
          "ownerIdentity": {
            "principalId": "EXAMPLE"
          },
          "arn": "arn:aws:s3:::sam-s3-demo-bucket"
```

```
        },
        "object": {
          "key": "test/key",
          "size": 1024,
          "eTag": "0123456789abcdef0123456789abcdef",
          "sequencer": "0A1B2C3D4E5F678901"
        }
      }
    }
  ]
}
```

Diese Beispielereignisse enthalten Platzhalterwerte. Sie können diese Werte ändern, um auf tatsächliche Ressourcen in Ihrer Anwendung oder auf Werte zu verweisen, die beim Testen vor Ort hilfreich sind.

Um ein Beispielereignis zu ändern

1. Sie können Beispielereignisse in der Befehlszeile ändern. Führen Sie den folgenden Befehl aus, um Ihre Optionen zu sehen:

   ```
   $ sam local generate-event <service> <event> --help
   ```

   Im Folgenden wird ein Beispiel gezeigt:

   ```
   $ sam local generate-event s3 put --help

   Usage: sam local generate-event s3 put [OPTIONS]

   Options:
     --region TEXT       Specify the region name you'd like, otherwise the
                         default = us-east-1
     --partition TEXT    Specify the partition name you'd like, otherwise the
                         default = aws
     --bucket TEXT       Specify the bucket name you'd like, otherwise the
                         default = example-bucket
     --key TEXT          Specify the key name you'd like, otherwise the default =
                         test/key
     --debug             Turn on debug logging to print debug message generated
                         by AWS SAM CLI and display timestamps.
     --config-file TEXT  Configuration file containing default parameter values.
   ```

```
                        [default: samconfig.toml]
  --config-env TEXT     Environment name specifying default parameter values in
                        the configuration file.  [default: default]
  -h, --help            Show this message and exit.
```

2. Verwenden Sie eine dieser Optionen in der Befehlszeile, um Ihre Beispielereignis-Payload zu ändern. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam local generate-event s3 put--bucket sam-s3-demo-bucket

{
  "Records": [
    {
      "eventVersion": "2.0",
      "eventSource": "aws:s3",
      "awsRegion": "us-east-1",
      "eventTime": "1970-01-01T00:00:00.000Z",
      "eventName": "ObjectCreated:Put",
      "userIdentity": {
        "principalId": "EXAMPLE"
      },
      "requestParameters": {
        "sourceIPAddress": "127.0.0.1"
      },
      "responseElements": {
        "x-amz-request-id": "EXAMPLE123456789",
        "x-amz-id-2": "EXAMPLE123/5678abcdefghijklambdaisawesome/
mnopqrstuvwxyzABCDEFGH"
      },
      "s3": {
        "s3SchemaVersion": "1.0",
        "configurationId": "testConfigRule",
        "bucket": {
          "name": "sam-s3-demo-bucket",
          "ownerIdentity": {
            "principalId": "EXAMPLE"
          },
          "arn": "arn:aws:s3:::sam-s3-demo-bucket"
        },
        "object": {
          "key": "test/key",
          "size": 1024,
          "eTag": "0123456789abcdef0123456789abcdef",
```

```
          "sequencer": "0A1B2C3D4E5F678901"
        }
      }
    }
  ]
}
```

Verwenden Sie generierte Ereignisse für lokale Tests

Speichern Sie Ihre generierten Ereignisse lokal und testen Sie sie mit anderen `sam local` Unterbefehlen.

Um Ihre generierten Ereignisse lokal zu speichern

- Führen Sie Folgendes aus:

  ```
  $ sam local generate-event <service> <event> <event-option> > <filename.json>
  ```

  Im Folgenden finden Sie ein Beispiel für ein Ereignis, das als `s3.json` Datei im `events` Ordner unseres Projekts gespeichert wird.

  ```
  sam-app$ sam local generate-event s3 put --bucket amzn-s3-demo-bucket > events/
  s3.json
  ```

Um ein generiertes Ereignis für lokale Tests zu verwenden

- Übergeben Sie das Ereignis mit anderen `sam local` Unterbefehlen, indem Sie die `--event` Option verwenden.

  Im Folgenden finden Sie ein Beispiel für die Verwendung des `s3.json` Ereignisses, um unsere Lambda-Funktion lokal aufzurufen:

  ```
  sam-app$ sam local invoke --event events/s3.json S3JsonLoggerFunction

  Invoking src/handlers/s3-json-logger.s3JsonLoggerHandler (nodejs18.x)
  Local image is up-to-date
  Using local image: public.ecr.aws/lambda/nodejs:18-rapid-x86_64.
  ```

```
Mounting /Users/.../sam-app/.aws-sam/build/S3JsonLoggerFunction as /var/
task:ro,delegated, inside runtime container
START RequestId: f4f45b6d-2ec6-4235-bc7b-495ec2ae0128 Version: $LATEST
END RequestId: f4f45b6d-2ec6-4235-bc7b-495ec2ae0128
REPORT RequestId: f4f45b6d-2ec6-4235-bc7b-495ec2ae0128  Init Duration: 1.23 ms
 Duration: 9371.93 ms      Billed Duration: 9372 ms        Memory Size: 128 MB
 Max Memory Used: 128 MB
```

Weitere Informationen

Eine Liste aller `sam local generate-event` Optionen finden Sie untersam local generate-event.

Eine Demo der Verwendung `sam local` finden Sie unter AWS SAM Lokale Entwicklung. Testen von AWS Cloud Ressourcen aus lokalen Entwicklungsumgebungen in der Serverless Land Sessions with SAM-Serie. YouTube

Einführung in das Testen mit sam local invoke

Verwenden Sie den AWS Serverless Application Model `sam local invoke` Unterbefehl Command Line Interface (AWS SAMCLI), um einen einmaligen lokalen Aufruf einer AWS Lambda Funktion zu initiieren.

- Eine Einführung in das finden Sie unter AWS SAMCLI Was ist der AWS SAM CLI?
- Eine Liste der `sam local invoke` Befehlsoptionen finden Sie untersam local invoke.
- Ein Beispiel für die Verwendung `sam local invoke` während eines typischen Entwicklungsworkflows finden Sie unterSchritt 7: (Optional) Testen Sie Ihre Anwendung lokal.

> **Note**
>
> Es wird nicht empfohlen, die lokalen Aufruffunktionen von SAM CLI in nicht vertrauenswürdigem Code zu verwenden. Um eine vollständige Isolierung von Ihrer lokalen Umgebung zu erreichen, führen Sie den Code direkt im Lambda-Service aus.

> **Note**
>
> `sam local invoke`unterstützt langlebige Funktionen mit automatischen Checkpoint- und Wiedergabefunktionen. Beim lokalen Aufrufen dauerhafter Funktionen wird der Ausführungsstatus automatisch verwaltet.

Voraussetzungen

Um das zu verwenden`sam local invoke`, installieren Sie das, AWS SAMCLI indem Sie die folgenden Schritte ausführen:

- AWS SAM Voraussetzungen.
- Installieren Sie das AWS SAMCLI.

Vor der Verwendung empfehlen wir`sam local invoke`, sich mit folgenden Grundkenntnissen vertraut zu machen:

- Konfiguration der AWS SAMCLI.
- Erstellen Sie Ihre Bewerbung in AWS SAM.
- Einführung in das Bauen mit AWS SAM.
- Einführung in die Bereitstellung mit AWS SAM.

Lokal eine Lambda-Funktion aufrufen

Bei der Ausführung AWS SAMCLI wird davon ausgegangen`sam local invoke`, dass Ihr aktuelles Arbeitsverzeichnis das Stammverzeichnis Ihres Projekts ist. Das AWS SAMCLI sucht zuerst nach einer `template.[yaml|yml]` Datei in einem `.aws-sam` Unterordner. Wenn nicht gefunden, sucht der AWS SAMCLI nach einer `template.[yaml|yml]` Datei in Ihrem aktuellen Arbeitsverzeichnis.

Um eine Lambda-Funktion lokal aufzurufen

1. Führen Sie im Stammverzeichnis Ihres Projekts Folgendes aus:

   ```
   $ sam local invoke <options>
   ```

2. Wenn Ihre Anwendung mehr als eine Funktion enthält, geben Sie die logische ID der Funktion an. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam local invoke HelloWorldFunction
```

3. Das AWS SAMCLI erstellt Ihre Funktion in einem lokalen Container mitDocker. Es ruft dann Ihre Funktion auf und gibt die Antwort Ihrer Funktion aus.

   Im Folgenden wird ein Beispiel gezeigt:

```
$ sam local invoke
Invoking app.lambda_handler (python3.9)
Local image is out of date and will be updated to the latest runtime. To skip this,
 pass in the parameter --skip-pull-image
Building
 image.....................................................................................
Using local image: public.ecr.aws/lambda/python:3.9-rapid-x86_64.

Mounting /Users/.../sam-app/.aws-sam/build/HelloWorldFunction as /var/
task:ro,delegated, inside runtime container
START RequestId: 64bf7e54-5509-4762-a97c-3d740498d3df Version: $LATEST
END RequestId: 64bf7e54-5509-4762-a97c-3d740498d3df
REPORT RequestId: 64bf7e54-5509-4762-a97c-3d740498d3df  Init Duration: 1.09 ms
  Duration: 608.42 ms       Billed Duration: 609 ms Memory Size: 128 MB     Max
 Memory Used: 128 MB
{"statusCode": 200, "body": "{\"message\": \"hello world\"}"}%
```

Verwalten von -Protokollen

Bei der Verwendung `sam local invoke` wird die Laufzeitausgabe der Lambda-Funktion (z. B. Protokolle) und das Ergebnis der Lambda-Funktion an ausgegeben. `stderr` `stdout`

Das Folgende ist ein Beispiel für eine grundlegende Lambda-Funktion:

```
def handler(event, context):
    print("some log") # this goes to stderr
    return "hello world" # this goes to stdout
```

Sie können diese Standardausgaben speichern. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam local invoke 1> stdout.log
...
```

```
$ cat stdout.log
"hello world"

$ sam local invoke 2> stderr.log
...

$ cat stderr.log
Invoking app.lambda_handler (python3.9)
Local image is up-to-date
Using local image: public.ecr.aws/lambda/python:3.9-rapid-x86_64.
Mounting /Users/.../sam-app/.aws-sam/build/HelloWorldFunction as /var/
task:ro,delegated, inside runtime container
START RequestId: 0b46e646-3bdf-4b58-8beb-242d00912c46 Version: $LATEST
some log
END RequestId: 0b46e646-3bdf-4b58-8beb-242d00912c46
REPORT RequestId: 0b46e646-3bdf-4b58-8beb-242d00912c46  Init Duration: 0.91 ms
 Duration: 589.19 ms Billed Duration: 590 ms Memory Size: 128 MB Max Memory Used: 128
 MB
```

Sie können diese Standardausgaben verwenden, um Ihre lokalen Entwicklungsprozesse weiter zu automatisieren.

Optionen

Übergeben Sie benutzerdefinierte Ereignisse, um die Lambda-Funktion aufzurufen

Verwenden Sie die `--event` Option, um ein Ereignis an die Lambda-Funktion zu übergeben. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam local invoke --event events/s3.json S3JsonLoggerFunction
```

Sie können Ereignisse mit dem `sam local generate-event` Unterbefehl erstellen. Weitere Informationen hierzu finden Sie unter Einführung in das Testen mit sam local generate-event.

Übergeben Sie Umgebungsvariablen, wenn Sie Ihre Lambda-Funktion aufrufen

Wenn Ihre Lambda-Funktion Umgebungsvariablen verwendet, können Sie diese bei lokalen Tests mit der `--env-vars` Option übergeben. Dies ist eine hervorragende Möglichkeit, eine Lambda-Funktion lokal mit Diensten in Ihrer Anwendung zu testen, die bereits in der Cloud bereitgestellt sind. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam local invoke --env-vars locals.json
```

Geben Sie eine Vorlage oder Funktion an

Verwenden Sie die `--template` Option, AWS SAMCLI um eine Vorlage anzugeben, auf die verwiesen werden soll. Es AWS SAMCLI werden nur diese AWS SAM Vorlage und die Ressourcen, auf die sie verweist, geladen.

Um eine Funktion einer verschachtelten Anwendung oder eines geschachtelten Stacks aufzurufen, geben Sie die logische ID der Anwendung oder des Stacks zusammen mit der logischen ID der Funktion an. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam local invoke StackLogicalId/FunctionLogicalId
```

Testen Sie eine Lambda-Funktion aus Ihrem Projekt Terraform

Verwenden Sie die `--hook-name` Option, um Lambda-Funktionen aus Ihren Terraform Projekten lokal zu testen. Weitere Informationen hierzu finden Sie unter Verwendung der AWS SAM CLI mit Terraform zum lokalen Debuggen und Testen.

Im Folgenden wird ein Beispiel gezeigt:

```
$ sam local invoke --hook-name terraform --beta-features
```

Best Practices

Wenn in Ihrer Anwendung ein `.aws-sam` Verzeichnis nicht ausgeführt wird`sam build`, stellen Sie sicher, dass es `sam build` jedes Mal ausgeführt wird, wenn Sie Ihren Funktionscode aktualisieren. Führen Sie dann `sam local invoke` den Befehl aus, um Ihren aktualisierten Funktionscode lokal zu testen.

Lokales Testen ist eine hervorragende Lösung für schnelles Entwickeln und Testen vor der Bereitstellung in der Cloud. Bei lokalen Tests wird jedoch nicht alles überprüft, z. B. die Berechtigungen zwischen Ihren Ressourcen in der Cloud. Testen Sie Ihre Anwendungen so oft wie möglich in der Cloud. Wir empfehlen `sam sync` die Verwendung, um Ihre Cloud-Test-Workflows zu beschleunigen.

Beispiele

Generieren Sie ein Amazon API Gateway Gateway-Beispielereignis und verwenden Sie es, um lokal eine Lambda-Funktion aufzurufen

Zunächst generieren wir eine API Gateway Gateway-HTTP-API-Event-Payload und speichern sie in unserem `events` Ordner.

```
$ sam local generate-event apigateway http-api-proxy > events/apigateway_event.json
```

Als Nächstes ändern wir unsere Lambda-Funktion, um einen Parameterwert aus dem Ereignis zurückzugeben.

```
def lambda_handler(event, context):
    print("HelloWorldFunction invoked")
    return {
        "statusCode": 200,
        "body": json.dumps({
            "message": event['queryStringParameters']['parameter2'],
        }),
    }
```

Als Nächstes rufen wir lokal unsere Lambda-Funktion auf und stellen unser benutzerdefiniertes Ereignis bereit.

```
$ sam local invoke --event events/apigateway_event.json

Invoking app.lambda_handler (python3.9)
Local image is up-to-date
Using local image: public.ecr.aws/lambda/python:3.9-rapid-x86_64.

Mounting /Users/...sam-app/.aws-sam/build/HelloWorldFunction as /var/task:ro,delegated,
 inside runtime container
START RequestId: 59535d0d-3d9e-493d-8c98-6264e8e961b8 Version: $LATEST
some log
END RequestId: 59535d0d-3d9e-493d-8c98-6264e8e961b8
REPORT RequestId: 59535d0d-3d9e-493d-8c98-6264e8e961b8  Init Duration: 1.63 ms
 Duration: 564.07 ms       Billed Duration: 565 ms Memory Size: 128 MB     Max Memory
 Used: 128 MB
{"statusCode": 200, "body": "{\"message\": \"value\"}"}%
```

Übergeben Sie Umgebungsvariablen, wenn Sie eine Lambda-Funktion lokal aufrufen

Diese Anwendung hat eine Lambda-Funktion, die eine Umgebungsvariable für einen Amazon DynamoDB-Tabellennamen verwendet. Das Folgende ist ein Beispiel für die in der Vorlage definierte Funktion: AWS SAM

```
AWSTemplateFormatVersion: 2010-09-09
Transform: AWS::Serverless-2016-10-31
...
Resources:
  getAllItemsFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: src/get-all-items.getAllItemsHandler
      Description: get all items
      Policies:
        - DynamoDBReadPolicy:
            TableName: !Ref SampleTable
      Environment:
        Variables:
          SAMPLE_TABLE: !Ref SampleTable
...
```

Wir möchten unsere Lambda-Funktion lokal testen und gleichzeitig mit unserer DynamoDB-Tabelle in der Cloud interagieren lassen. Dazu erstellen wir unsere Umgebungsvariablendatei und speichern sie im Stammverzeichnis unseres Projekts unter. `locals.json` Der hier angegebene Wert für `SAMPLE_TABLE` verweist auf unsere DynamoDB-Tabelle in der Cloud.

```
{
    "getAllItemsFunction": {
        "SAMPLE_TABLE": "dev-demo-SampleTable-1U991234LD5UM98"
    }
}
```

Als Nächstes führen wir unsere Umgebungsvariablen mit der Option aus `sam local invoke` und übergeben sie. `--env-vars`

```
$ sam local invoke getAllItemsFunction --env-vars locals.json

Mounting /Users/...sam-app/.aws-sam/build/HelloWorldFunction as /var/task:ro,delegated,
 inside runtime container
```

```
START RequestId: 59535d0d-3d9e-493d-8c98-6264e8e961b8 Version: $LATEST
some log
END RequestId: 59535d0d-3d9e-493d-8c98-6264e8e961b8
REPORT RequestId: 59535d0d-3d9e-493d-8c98-6264e8e961b8  Init Duration: 1.63 ms
 Duration: 564.07 ms       Billed Duration: 565 ms Memory Size: 128 MB     Max Memory
 Used: 128 MB
{"statusCode":200,"body":"{}"}
```

Weitere Informationen

Eine Liste aller `sam local invoke` Optionen finden Sie untersam local invoke.

Eine Demo der Verwendung `sam local` finden Sie unter AWS SAM Lokale Entwicklung. Testen von AWS Cloud Ressourcen aus lokalen Entwicklungsumgebungen in der Serverless Land Sessions with SAM-Serie. YouTube

Einführung in das Testen mit dieselbe lokale Start-API

Verwenden Sie den AWS Serverless Application Model `sam local start-api` Unterbefehl Command Line Interface (AWS SAMCLI), um Ihre AWS Lambda Funktionen lokal auszuführen und über einen lokalen HTTP-Serverhost zu testen. Diese Art von Test ist hilfreich für Lambda-Funktionen, die von einem Amazon API Gateway Gateway-Endpunkt aufgerufen werden.

- Eine Einführung in das finden Sie unter AWS SAMCLI Was ist der AWS SAM CLI?
- Eine Liste der `sam local start-api` Befehlsoptionen finden Sie unterSam lokale Start-API.
- Ein Beispiel für die Verwendung `sam local start-api` während eines typischen Entwicklungsworkflows finden Sie unterSchritt 7: (Optional) Testen Sie Ihre Anwendung lokal.

> **Note**
>
> `sam local start-api`unterstützt langlebige Funktionen mit automatischem Checkpoint und Wiedergabefunktionen. Dauerhafte Funktionen arbeiten nahtlos mit dem lokalen API-Gateway-Endpunkt zusammen.

Voraussetzungen

Um das zu verwendensam local start-api, installieren Sie das, AWS SAMCLI indem Sie wie folgt vorgehen:

- AWS SAM Voraussetzungen.
- Installieren Sie das AWS SAMCLI.

Vor der Verwendung empfehlen wir`sam local start-api`, sich mit folgenden Grundkenntnissen vertraut zu machen:

- Konfiguration der AWS SAMCLI.
- Erstellen Sie Ihre Bewerbung in AWS SAM.
- Einführung in das Bauen mit AWS SAM.
- Einführung in die Bereitstellung mit AWS SAM.

Verwenden Sie dieselbe lokale Start-API

Beim Ausführen AWS SAMCLI geht das davon aus`sam local start-api`, dass Ihr aktuelles Arbeitsverzeichnis das Stammverzeichnis Ihres Projekts ist. Das AWS SAMCLI sucht zuerst nach einer `template.[yaml|yml]` Datei in einem `.aws-sam` Unterordner. Wenn nicht gefunden, sucht der AWS SAMCLI nach einer `template.[yaml|yml]` Datei in Ihrem aktuellen Arbeitsverzeichnis.

Um einen lokalen HTTP-Server zu starten

1. Führen Sie im Stammverzeichnis Ihres Projekts Folgendes aus:

   ```
   $ sam local start-api <options>
   ```

2. Das AWS SAMCLI erstellt Ihre Lambda-Funktionen in einem lokalen Docker Container. Anschließend wird die lokale Adresse Ihres HTTP-Serverendpunkts ausgegeben. Im Folgenden wird ein Beispiel gezeigt:

   ```
   $ sam local start-api

   Initializing the lambda functions containers.
   Local image is up-to-date
   Using local image: public.ecr.aws/lambda/python:3.9-rapid-x86_64.

   Mounting /Users/.../sam-app/.aws-sam/build/HelloWorldFunction as /var/
   task:ro,delegated, inside runtime container
   Containers Initialization is done.
   Mounting HelloWorldFunction at http://127.0.0.1:3000/hello [GET]
   ```

```
You can now browse to the above endpoints to invoke your functions. You do not
 need to restart/reload SAM CLI while working on your functions, changes will be
 reflected instantly/automatically. If you used sam build before running local
 commands, you will need to re-run sam build for the changes to be picked up. You
 only need to restart SAM CLI if you update your AWS SAM template
2023-04-12 14:41:05 WARNING: This is a development server. Do not use it in a
 production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:3000
```

3. Sie können Ihre Lambda-Funktion über den Browser oder die Befehlszeile aufrufen. Im Folgenden wird ein Beispiel gezeigt:

```
sam-app$ curl http://127.0.0.1:3000/hello
{"message": "Hello world!"}%
```

4. Wenn Sie Änderungen an Ihrem Lambda-Funktionscode vornehmen, sollten Sie Folgendes beachten, um Ihren lokalen HTTP-Server zu aktualisieren:

   - Wenn Ihre Anwendung kein `.aws-sam` Verzeichnis hat und Ihre Funktion eine interpretierte Sprache verwendet, aktualisiert sie Ihre Funktion AWS SAMCLI automatisch, indem sie einen neuen Container erstellt und diesen hostet.
   - Wenn Ihre Anwendung über ein `.aws-sam` Verzeichnis verfügt, müssen Sie ausführen, `sam build` um Ihre Funktion zu aktualisieren. Führen Sie dann `sam local start-api` erneut aus, um die Funktion zu hosten.
   - Wenn Ihre Funktion eine kompilierte Sprache verwendet oder wenn Ihr Projekt komplexe Paketierungsunterstützung erfordert, führen Sie Ihre eigene Build-Lösung aus, um Ihre Funktion zu aktualisieren. Führen Sie dann `sam local start-api` erneut aus, um die Funktion zu hosten.

Lambda-Funktionen, die Lambda-Autorisierer verwenden

> **Note**
>
> Diese Funktion ist neu in Version 1.80.0. AWS SAMCLI Informationen zum Upgrade finden Sie unter Aktualisierung des AWS SAMCLI.

Bei Lambda-Funktionen, die Lambda-Autorisierer verwenden, rufen sie AWS SAMCLI automatisch Ihren Lambda-Autorisierer auf, bevor Sie Ihren Lambda-Funktionsendpunkt aufrufen.

Im Folgenden finden Sie ein Beispiel für das Starten eines lokalen HTTP-Servers für eine Funktion, die einen Lambda-Authorizer verwendet:

```
$ sam local start-api
2023-04-17 15:02:13 Attaching import module proxy for analyzing dynamic imports

AWS SAM CLI does not guarantee 100% fidelity between authorizers locally
and authorizers deployed on AWS. Any application critical behavior should
be validated thoroughly before deploying to production.

Testing application behaviour against authorizers deployed on AWS can be done using the
 sam sync command.

Mounting HelloWorldFunction at http://127.0.0.1:3000/authorized-request [GET]
You can now browse to the above endpoints to invoke your functions. You do not need
 to restart/reload SAM CLI while working on your functions, changes will be reflected
 instantly/automatically. If you used sam build before running local commands, you will
 need to re-run sam build for the changes to be picked up. You only need to restart SAM
 CLI if you update your AWS SAM template
2023-04-17 15:02:13 WARNING: This is a development server. Do not use it in a
 production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:3000
2023-04-17 15:02:13 Press CTRL+C to quit
```

Wenn Sie Ihren Lambda-Funktionsendpunkt über den lokalen HTTP-Server aufrufen, ruft der AWS SAMCLI erste Ihren Lambda-Authorizer auf. Wenn die Autorisierung erfolgreich ist, AWS SAMCLI wird der Endpunkt Ihrer Lambda-Funktion aufgerufen. Im Folgenden wird ein Beispiel gezeigt:

```
$ curl http://127.0.0.1:3000/authorized-request --header "header:my_token"
{"message": "from authorizer"}%

Invoking app.authorizer_handler (python3.8)
Local image is up-to-date
Using local image: public.ecr.aws/lambda/python:3.8-rapid-x86_64.

Mounting /Users/.../sam-app/... as /var/task:ro,delegated, inside runtime container
START RequestId: 38d3b472-a2c8-4ea6-9a77-9b386989bef0 Version: $LATEST
END RequestId: 38d3b472-a2c8-4ea6-9a77-9b386989bef0
REPORT RequestId: 38d3b472-a2c8-4ea6-9a77-9b386989bef0    Init Duration: 1.08 ms
 Duration: 628.26 msBilled Duration: 629 ms    Memory Size: 128 MB    Max Memory Used:
 128 MB
Invoking app.request_handler (python3.8)
Using local image: public.ecr.aws/lambda/python:3.8-rapid-x86_64.
```

```
Mounting /Users/.../sam-app/... as /var/task:ro,delegated, inside runtime container
START RequestId: fdc12255-79a3-4365-97e9-9459d06446ff Version: $LATEST
END RequestId: fdc12255-79a3-4365-97e9-9459d06446ff
REPORT RequestId: fdc12255-79a3-4365-97e9-9459d06446ff    Init Duration: 0.95 ms
 Duration: 659.13 msBilled Duration: 660 ms    Memory Size: 128 MB    Max Memory Used:
 128 MB
No Content-Type given. Defaulting to 'application/json'.
2023-04-17 15:03:03 127.0.0.1 - - [17/Apr/2023 15:03:03] "GET /authorized-request
 HTTP/1.1" 200 -
```

Optionen

Kontinuierliche Wiederverwendung von Containern zur Beschleunigung lokaler Funktionsaufrufe

Standardmäßig AWS SAMCLI erstellt der jedes Mal, wenn Ihre Funktion über den lokalen HTTP-Server aufgerufen wird, einen neuen Container. Verwenden Sie die `--warm-containers` Option, um Ihren Container automatisch für Funktionsaufrufe wiederzuverwenden. Dies beschleunigt die Zeit, die benötigt wird, AWS SAMCLI um Ihre Lambda-Funktion für den lokalen Aufruf vorzubereiten. Sie können diese Option weiter anpassen, indem Sie das Argument `eager` oder `lazy` angeben.

- `eager`— Container für alle Funktionen werden beim Start geladen und bleiben zwischen Aufrufen bestehen.
- `lazy`— Container werden nur geladen, wenn jede Funktion zum ersten Mal aufgerufen wird. Sie bleiben dann für weitere Aufrufe bestehen.

Im Folgenden wird ein Beispiel gezeigt:

```
$ sam local start-api --warm-containers eager
```

Wenn Sie Ihren Lambda-Funktionscode verwenden `--warm-containers` und ändern:

- Wenn Ihre Anwendung über ein `.aws-sam` Verzeichnis verfügt, führen Sie `sam build` den Befehl aus, um Ihren Funktionscode in den Build-Artefakten Ihrer Anwendung zu aktualisieren.
- Wenn eine Codeänderung erkannt wird, wird der Lambda-Funktionscontainer AWS SAMCLI automatisch heruntergefahren.
- Wenn Sie die Funktion erneut aufrufen, erstellt der AWS SAMCLI automatisch einen neuen Container.

Wenn die Dateiüberwachung zu einem hohen CPU-Aufwand oder I/O zu hohem Aufwand führt, z. B. in großen Monorepos, Umgebungen mit Antivirensoftware oder Projekten, in denen sich mehrere Funktionen eine teilen, `CodeUri` können Sie sie mit der folgenden Option deaktivieren: `--no-watch`

```
$ sam local start-api --warm-containers eager --no-watch
```

Bei werden Code `--no-watch` - und Vorlagenänderungen nicht automatisch erkannt. Beenden Sie den Befehl und führen Sie ihn erneut aus, um die Änderungen zu übernehmen.

Geben Sie ein Container-Image an, das für Ihre Lambda-Funktionen verwendet werden soll

Standardmäßig AWS SAMCLI verwendet der Lambda-Basisimages aus Amazon Elastic Container Registry (Amazon ECR), um Ihre Funktionen lokal aufzurufen. Verwenden Sie die `--invoke-image` Option, um auf ein benutzerdefiniertes Container-Image zu verweisen. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam local start-api --invoke-image public.ecr.aws/sam/emu-python3.8
```

Sie können die Funktion angeben, die mit dem benutzerdefinierten Container-Image verwendet werden soll. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam local start-api --invoke-image Function1=amazon/aws/sam-cli-emulation-image-
python3.8
```

Geben Sie eine Vorlage an, die lokal getestet werden soll

Verwenden Sie die `--template` Option, AWS SAMCLI um eine Vorlage anzugeben, auf die verwiesen werden soll. Es AWS SAMCLI werden nur diese AWS SAM Vorlage und die Ressourcen, auf die sie verweist, geladen. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam local start-api --template myTemplate.yaml
```

Geben Sie die Host-Entwicklungsumgebung Ihrer Lambda-Funktion an

Standardmäßig erstellt der `sam local start-api` Unterbefehl einen HTTP-Server `localhost` mit IP-Adresse. `127.0.0.1` Sie können diese Werte anpassen, wenn Ihre lokale Entwicklungsumgebung von Ihrem lokalen Computer isoliert ist.

Verwenden Sie die `--container-host` Option, um einen Host anzugeben. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam local start-api --container-host host.docker.internal
```

Verwenden Sie die `--container-host-interface` Option, um die IP-Adresse des Host-Netzwerks anzugeben, an das Container-Ports gebunden werden sollen. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam local start-api --container-host-interface 0.0.0.0
```

Best Practices

Wenn Ihre Anwendung ein `.aws-sam` Verzeichnis hat, das nicht ausgeführt wird`sam build`, stellen Sie sicher, dass es `sam build` jedes Mal ausgeführt wird, wenn Sie Ihren Funktionscode aktualisieren. Führen Sie dann `sam local start-api` die Ausführung aus, um Ihren aktualisierten Funktionscode lokal zu testen.

Lokales Testen ist eine hervorragende Lösung für schnelles Entwickeln und Testen vor der Bereitstellung in der Cloud. Bei lokalen Tests wird jedoch nicht alles überprüft, z. B. die Berechtigungen zwischen Ihren Ressourcen in der Cloud. Testen Sie Ihre Anwendungen so oft wie möglich in der Cloud. Wir empfehlen `sam sync` die Verwendung, um Ihre Cloud-Test-Workflows zu beschleunigen.

Weitere Informationen

Eine Liste aller `sam local start-api` Optionen finden Sie unterSam lokale Start-API.

Einführung in das Testen mit sam local start-lambda

Verwenden Sie den AWS SAMCLI Unterbefehl`sam local start-lambda`, um Ihre Lambda-Funktion über und aufzurufen. AWS CLI SDKs Dieser Befehl startet einen lokalen Endpunkt, der Lambda emuliert.

- Eine Einführung in das finden Sie unter AWS SAMCLI Was ist der AWS SAM CLI?
- Eine Liste der `sam local start-lambda` Befehlsoptionen finden Sie unterSam Local Start-Lambda.

> **Note**
>
> `sam local start-lambda`unterstützt langlebige Funktionen mit automatischem Checkpoint und Wiedergabefunktionen. Dauerhafte Funktionen arbeiten nahtlos mit dem lokalen Lambda-Endpunkt zusammen.

Voraussetzungen

Um das zu verwenden`sam local start-lambda`, installieren Sie das, AWS SAMCLI indem Sie wie folgt vorgehen:

- AWS SAM Voraussetzungen.
- Installieren Sie das AWS SAMCLI.

Vor der Verwendung empfehlen wir`sam local start-lambda`, sich mit folgenden Grundkenntnissen vertraut zu machen:

- Konfiguration der AWS SAMCLI.
- Erstellen Sie Ihre Bewerbung in AWS SAM.
- Einführung in das Bauen mit AWS SAM.
- Einführung in die Bereitstellung mit AWS SAM.

Sam Local Start-Lambda verwenden

Beim Ausführen AWS SAMCLI geht das davon aus`sam local start-lambda`, dass Ihr aktuelles Arbeitsverzeichnis das Stammverzeichnis Ihres Projekts ist. Das AWS SAMCLI sucht zuerst nach einer `template.[yaml|yml]` Datei in einem `.aws-sam` Unterordner. Wenn nicht gefunden, sucht der AWS SAMCLI nach einer `template.[yaml|yml]` Datei in Ihrem aktuellen Arbeitsverzeichnis.

Um Sam Local Start-Lambda zu verwenden

1. Führen Sie im Stammverzeichnis Ihres Projekts Folgendes aus:

```
$ sam local start-lambda <options>
```

2. Das AWS SAMCLI erstellt Ihre Lambda-Funktionen in einem lokalen Docker Container. Anschließend wird die lokale Adresse an Ihren HTTP-Serverendpunkt ausgegeben. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam local start-lambda
Initializing the lambda functions containers.
Local image is up-to-date
Using local image: public.ecr.aws/lambda/python:3.9-rapid-x86_64.

Mounting /Users/.../sam-app/hello_world as /var/task:ro,delegated, inside runtime
 container
Containers Initialization is done.
Starting the Local Lambda Service. You can now invoke your Lambda Functions defined
 in your template through the endpoint.
2023-04-13 07:25:43 WARNING: This is a development server. Do not use it in a
 production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:3001
2023-04-13 07:25:43 Press CTRL+C to quit
```

3. Verwenden Sie das AWS CLI oder SDKs , um Ihre Lambda-Funktion lokal aufzurufen.

   Das Folgende ist ein Beispiel für die Verwendung von: AWS CLI

```
$ aws lambda invoke --function-name "HelloWorldFunction" --endpoint-
url "http://127.0.0.1:3001" --no-verify-ssl out.txt

StatusCode: 200
(END)
```

   Das Folgende ist ein Beispiel AWS SDK für die Verwendung vonPython:

```
import boto3
from botocore.config import Config
from botocore import UNSIGNED

lambda_client = boto3.client('lambda',
                             endpoint_url="http://127.0.0.1:3001",
                             use_ssl=False,
                             verify=False,
                             config=Config(signature_version=UNSIGNED,
                                           read_timeout=1,
                                           retries={'max_attempts': 0}
```

```
                                        )
                                )
lambda_client.invoke(FunctionName="HelloWorldFunction")
```

Optionen

Geben Sie eine Vorlage an

Verwenden Sie die `--template` Option, AWS SAMCLI um eine Vorlage anzugeben, auf die verwiesen werden soll. Es AWS SAMCLI werden nur diese AWS SAM Vorlage und die Ressourcen, auf die sie verweist, geladen. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam local start-lambda --template myTemplate.yaml
```

Weitere Informationen zu AWS SAM Vorlagen finden Sie unterAWS SAM Anatomie der Vorlage.

Best Practices

Wenn Ihre Anwendung ein `.aws-sam` Verzeichnis hat, das ausgeführt wird`sam build`, stellen Sie sicher, dass es `sam build` jedes Mal ausgeführt wird, wenn Sie Ihren Funktionscode aktualisieren. Führen Sie dann `sam local start-lambda` die Ausführung aus, um Ihren aktualisierten Funktionscode lokal zu testen.

Lokales Testen ist eine hervorragende Lösung für schnelles Entwickeln und Testen vor der Bereitstellung in der Cloud. Bei lokalen Tests wird jedoch nicht alles überprüft, z. B. die Berechtigungen zwischen Ihren Ressourcen in der Cloud. Testen Sie Ihre Anwendungen so oft wie möglich in der Cloud. Wir empfehlen `sam sync` die Verwendung, um Ihre Cloud-Test-Workflows zu beschleunigen.

Weitere Informationen

Eine Liste aller `sam local start-lambda` Optionen finden Sie unterSam Local Start-Lambda.

Lokal Lambda-Funktionen aufrufen mit AWS SAM

Das lokale Aufrufen einer Lambda-Funktion vor dem Testen oder Bereitstellen in der Cloud kann eine Reihe von Vorteilen haben. Es ermöglicht Ihnen, die Logik Ihrer Funktion schneller zu testen. Wenn Sie zuerst lokal testen, verringert sich die Wahrscheinlichkeit, dass Probleme beim Testen in

der Cloud oder während der Bereitstellung erkannt werden, wodurch Sie unnötige Kosten vermeiden können. Darüber hinaus erleichtern lokale Tests das Debuggen.

Sie können Ihre Lambda-Funktion lokal aufrufen, indem Sie den sam local invoke Befehl verwenden und die logische ID der Funktion und eine Ereignisdatei angeben. sam local invokeakzeptiert auch `stdin` als Ereignis. Weitere Informationen zu Veranstaltungen finden Sie unter Ereignis im AWS Lambda Entwicklerhandbuch. Informationen zu den Formaten von Ereignisnachrichten verschiedener AWS Dienste finden Sie im AWS Lambda Entwicklerhandbuch unter Verwendung AWS Lambda mit anderen Diensten.

> **Note**
>
> Es wird nicht empfohlen, die lokalen Aufruffunktionen von SAM CLI in nicht vertrauenswürdigem Code zu verwenden. Um vollständig von Ihrer lokalen Umgebung isoliert zu sein, führen Sie den Code direkt im Lambda-Service aus.

> **Note**
>
> Der sam local invoke Befehl entspricht dem Befehl AWS Command Line Interface (AWS CLI). `aws lambda invoke` Sie können beide Befehle verwenden, um eine Lambda-Funktion aufzurufen.

Sie müssen den sam local invoke Befehl in dem Projektverzeichnis ausführen, das die Funktion enthält, die Sie aufrufen möchten.

Beispiele:

```
# Invoking function with event file
$ sam local invoke "Ratings" -e event.json

# Invoking function with event via stdin
$ echo '{"message": "Hey, are you there?" }' | sam local invoke --event - "Ratings"

# For more options
$ sam local invoke --help
```

Umgebungsvariablendatei

Gehen Sie wie folgt vor, um Umgebungsvariablen lokal zu deklarieren, die die in Ihren Vorlagen definierten Werte überschreiben:

1. Erstellen Sie eine `.env` JSON-Datei oder eine Datei, die die zu überschreibenden Umgebungsvariablen enthält.
2. Verwenden Sie das `--env-vars` Argument, um die in Ihren Vorlagen definierten Werte zu überschreiben.

Die `--env-vars` Option unterstützt zwei Dateiformate. Das Dateiformat wird anhand des Dateiinhalts automatisch erkannt.

Deklarieren von Umgebungsvariablen mit JSON

Um Umgebungsvariablen zu deklarieren, die global für alle Ressourcen gelten, geben Sie ein `Parameters` Objekt wie das Folgende an:

```
{
    "Parameters": {
        "TABLE_NAME": "localtable",
        "BUCKET_NAME": "amzn-s3-demo-bucket",
        "STAGE": "dev"
    }
}
```

Um unterschiedliche Umgebungsvariablen für jede Ressource zu deklarieren, geben Sie Objekte für jede Ressource wie folgt an:

```
{
    "MyFunction1": {
        "TABLE_NAME": "localtable",
        "BUCKET_NAME": "amzn-s3-demo-bucket",
    },
    "MyFunction2": {
        "TABLE_NAME": "localtable",
        "STAGE": "dev"
    }
}
```

Bei der Angabe von Objekten für jede Ressource können Sie die folgenden Bezeichner verwenden, die in der Reihenfolge von höchster bis niedrigster Priorität aufgeführt sind:

1. `logical_id`
2. `function_id`
3. `function_name`
4. Vollständige Pfadkennung

Sie können beide oben genannten Methoden zum Deklarieren von Umgebungsvariablen zusammen in einer einzigen Datei verwenden. Dabei haben Umgebungsvariablen, die Sie für bestimmte Ressourcen bereitgestellt haben, Vorrang vor globalen Umgebungsvariablen.

Speichern Sie Ihre Umgebungsvariablen in einer JSON-Datei, z. B. `env.json`

Deklarieren von Umgebungsvariablen mit .env-Dateien

Sie können auch eine `.env` Datei verwenden, um Umgebungsvariablen zu deklarieren. In einer `.env` Datei deklarierte Variablen gelten global für alle Funktionen und entsprechen dem `Parameters` Objekt im JSON-Format.

```
TABLE_NAME=localtable
BUCKET_NAME=amzn-s3-demo-bucket
STAGE=dev
```

Das `.env` Format unterstützt Kommentare (Zeilen, die mit beginnen#) und Werte in Anführungszeichen.

> **Note**
>
> Das `.env` Format unterstützt nur globale Umgebungsvariablen. Verwenden Sie das JSON-Format, um funktionsspezifische Umgebungsvariablen zu deklarieren.

Werte von Umgebungsvariablen überschreiben

Um Umgebungsvariablen mit denen zu überschreiben, die in Ihrer Umgebungsvariablendatei definiert sind, verwenden Sie das `--env-vars` Argument mit den start-api Befehlen invoke oder. Beispiel:

```
# Using a JSON file
```

```
sam local invoke --env-vars env.json

# Using a .env file
sam local invoke --env-vars .env
```

Layer

Wenn Ihre Anwendung Ebenen enthält, finden Sie Informationen zum Debuggen von Problemen mit Layern auf Ihrem lokalen Host unterSteigern Sie die Effizienz mithilfe von Lambda-Schichten mit AWS SAM.

Weitere Informationen

Ein praktisches Beispiel für das lokale Aufrufen von Funktionen finden Sie in Modul 2 — Lokal ausführen in The Complete AWS SAM Workshop.

Lokal ausgeführtes API Gateway mit AWS SAM

Die lokale Ausführung von Amazon API Gateway kann eine Vielzahl von Vorteilen haben. Wenn Sie API Gateway lokal ausführen, können Sie beispielsweise API-Endpunkte vor der Bereitstellung in der AWS Cloud lokal testen. Wenn Sie zuerst lokal testen, können Sie häufig weniger Tests und Entwicklungen in der Cloud durchführen, was zur Kostensenkung beitragen kann. Darüber hinaus erleichtert die lokale Ausführung das Debuggen.

Verwenden Sie den Befehl, um eine lokale Instanz von API Gateway zu starten, mit der Sie die HTTP-Anforderungs-/Antwortfunktionalität testen können. `Sam lokale Start-API` AWS SAMCLI Diese Funktion bietet Hot-Reloading, sodass Sie Ihre Funktionen schnell weiterentwickeln und überarbeiten können.

> **Note**
>
> Beim erneuten Laden im laufenden Betrieb werden nur die Dateien aktualisiert, die sich geändert haben, und der Status der Anwendung bleibt unverändert. Im Gegensatz dazu wird beim Live-Reloading die gesamte Anwendung aktualisiert und der Status der Anwendung geht verloren.

Anweisungen zur Verwendung des `sam local start-api` Befehls finden Sie unter. Einführung in das Testen mit dieselbe lokale Start-API

AWS SAM Verwendet standardmäßig AWS Lambda Proxyintegrationen und unterstützt sowohl `HttpApi` `Api` Ressourcentypen als auch. Weitere Informationen zu Proxyintegrationen für `HttpApi` Ressourcentypen finden Sie unter Arbeiten mit AWS Lambda Proxyintegrationen für HTTP APIs im API Gateway Developer Guide. Weitere Informationen zu Proxyintegrationen mit `Api` Ressourcentypen finden Sie unter Grundlegendes zur API Gateway Lambda-Proxyintegration im API Gateway Developer Guide.

Beispiel:

```
$ sam local start-api
```

AWS SAM findet automatisch alle Funktionen in Ihrer AWS SAM Vorlage, für die `Api` Ereignisquellen `HttpApi` definiert wurden. Dann hängt es die Funktion an den definierten HTTP-Pfaden ein.

Im folgenden `Api` Beispiel wird die `Ratings` Funktion `ratings.py:handler()` bei For-Anfragen eingebunden`/ratings`: `GET`

```
Ratings:
  Type: AWS::Serverless::Function
  Properties:
    Handler: ratings.handler
    Runtime: python3.9
    Events:
      Api:
        Type: Api
        Properties:
          Path: /ratings
          Method: get
```

Hier ist ein Beispiel für eine `Api` Antwort:

```
// Example of a Proxy Integration response
exports.handler = (event, context, callback) => {
    callback(null, {
        statusCode: 200,
        headers: { "x-custom-header" : "my custom header value" },
        body: "hello world"
    });
}
```

Wenn Sie den Code Ihrer Funktion ändern, führen Sie den `sam build` Befehl for aus, `sam local start-api` um Ihre Änderungen zu erkennen.

Umgebungsvariablendatei

Gehen Sie wie folgt vor, um Umgebungsvariablen lokal zu deklarieren, die die in Ihren Vorlagen definierten Werte überschreiben:

1. Erstellen Sie eine JSON-Datei, die die zu überschreibenden Umgebungsvariablen enthält.
2. Verwenden Sie das `--env-vars` Argument, um die in Ihren Vorlagen definierten Werte zu überschreiben.

Umgebungsvariablen deklarieren

Um Umgebungsvariablen zu deklarieren, die global für alle Ressourcen gelten, geben Sie ein `Parameters` Objekt wie das Folgende an:

```
{
    "Parameters": {
        "TABLE_NAME": "localtable",
        "BUCKET_NAME": "amzn-s3-demo-bucket",
        "STAGE": "dev"
    }
}
```

Um unterschiedliche Umgebungsvariablen für jede Ressource zu deklarieren, geben Sie Objekte für jede Ressource wie folgt an:

```
{
    "MyFunction1": {
        "TABLE_NAME": "localtable",
        "BUCKET_NAME": "amzn-s3-demo-bucket",
    },
    "MyFunction2": {
        "TABLE_NAME": "localtable",
        "STAGE": "dev"
    }
}
```

Bei der Angabe von Objekten für jede Ressource können Sie die folgenden Bezeichner verwenden, die in der Reihenfolge von höchster bis niedrigster Priorität aufgeführt sind:

1. `logical_id`
2. `function_id`
3. `function_name`
4. Vollständige Pfadkennung

Sie können beide oben genannten Methoden zum Deklarieren von Umgebungsvariablen zusammen in einer einzigen Datei verwenden. Dabei haben Umgebungsvariablen, die Sie für bestimmte Ressourcen bereitgestellt haben, Vorrang vor globalen Umgebungsvariablen.

Speichern Sie Ihre Umgebungsvariablen in einer JSON-Datei, z. B. `env.json`

Werte von Umgebungsvariablen überschreiben

Um Umgebungsvariablen mit den in Ihrer JSON-Datei definierten Variablen zu überschreiben, verwenden Sie das `--env-vars` Argument mit den start-api Befehlen invoke oder. Beispiel:

```
$ sam local start-api --env-vars env.json
```

Layer

Wenn Ihre Anwendung Ebenen enthält, finden Sie Informationen zum Debuggen von Problemen mit Layern auf Ihrem lokalen Host unterSteigern Sie die Effizienz mithilfe von Lambda-Schichten mit AWS SAM.

Einführung in Cloud-Tests mit sam remote test-event

Verwenden Sie den AWS Serverless Application Model Befehl Command Line Interface (AWS SAM CLI)`sam remote test-event`, um auf gemeinsam nutzbare Testereignisse für Ihre AWS Lambda Funktionen zuzugreifen und diese zu verwalten.

Weitere Informationen zu gemeinsam nutzbaren Testereignissen finden Sie unter Gemeinsam nutzbare Testereignisse im AWS Lambda Entwicklerhandbuch.

Topics

- Richten Sie das ein AWS SAMCLI, um es zu verwenden sam remote test-event

- Verwenden Sie den Befehl sam remote test-event
- Testereignisse verwenden, die gemeinsam genutzt werden können
- Verwaltung gemeinsam nutzbarer Testereignisse

Voraussetzungen

Um das zu verwenden`sam remote test-event`, installieren Sie das, AWS SAMCLI indem Sie wie folgt vorgehen:

- AWS SAM Voraussetzungen.
- Installieren Sie das AWS SAMCLI.

Wenn Sie die Version bereits AWS SAM CLI installiert haben, empfehlen wir, auf die neueste Version der AWS SAMCLI Version zu aktualisieren. Weitere Informationen hierzu finden Sie unter Aktualisierung des AWS SAMCLI.

Vor der Verwendung empfehlen wir`sam remote test-event`, sich mit folgenden Grundkenntnissen vertraut zu machen:

- Konfiguration der AWS SAMCLI.
- Erstellen Sie Ihre Bewerbung in AWS SAM.
- Einführung in das Bauen mit AWS SAM.
- Einführung in die Bereitstellung mit AWS SAM.
- Einführung in die Verwendung von sam sync to sync to AWS Cloud.

Richten Sie das ein AWS SAMCLI, um es zu verwenden sam remote test-event

Führen Sie die folgenden Einrichtungsschritte aus, um den Befehl zu verwenden: AWS SAM CLI `sam remote test-event`

1. Configure the AWS SAM CLI to use your AWS-Konto — Gemeinsam nutzbare Testereignisse für Lambda können von Benutzern innerhalb desselben aufgerufen und verwaltet werden. AWS-Konto Informationen zur Konfiguration für AWS SAM CLI die Verwendung Ihres finden Sie AWS-Konto unter. Konfiguration der AWS SAMCLI

2. Berechtigungen für gemeinsam nutzbare Testereignisse konfigurieren — Um auf gemeinsam nutzbare Testereignisse zuzugreifen und diese zu verwalten, benötigen Sie die entsprechenden Berechtigungen. Weitere Informationen finden Sie unter Gemeinsam nutzbare Testereignisse im Entwicklerhandbuch.AWS Lambda

Verwenden Sie den Befehl sam remote test-event

Der AWS SAM CLI `sam remote test-event` Befehl enthält die folgenden Unterbefehle, mit denen Sie auf Ihre gemeinsam nutzbaren Testereignisse zugreifen und diese verwalten können:

- `delete`— Löscht ein gemeinsam nutzbares Testereignis aus der EventBridge Amazon-Schemaregistry.
- `get`— Ruft ein gemeinsam nutzbares Testereignis aus der EventBridge Schemaregistry ab.
- `list`— Listet die vorhandenen gemeinsam nutzbaren Testereignisse für eine Funktion aus der EventBridge Schemaregistry auf.
- `put`— Speichert ein Ereignis aus einer lokalen Datei in der EventBridge Schemaregistrierung.

Um diese Unterbefehle mithilfe von aufzulisten AWS SAM CLI, führen Sie den folgenden Befehl aus:

```
$ sam remote test-event --help
```

Löschen von gemeinsam nutzbaren Testereignissen

Sie können ein gemeinsam nutzbares Testereignis löschen, indem Sie den `delete` Unterbefehl zusammen mit dem folgenden Befehl verwenden:

- Geben Sie den Namen des gemeinsam nutzbaren Testereignisses ein, das gelöscht werden soll.
- Geben Sie eine akzeptable ID der Lambda-Funktion an, die dem Ereignis zugeordnet ist.
- Wenn Sie die logische ID der Lambda-Funktion angeben, müssen Sie auch den AWS CloudFormation Stacknamen angeben, der der Lambda-Funktion zugeordnet ist.

Im Folgenden wird ein Beispiel gezeigt:

```
$ sam remote test-event delete HelloWorldFunction --stack-name sam-app --name demo-
event
```

Eine Liste der Optionen, die mit dem `delete` Unterbefehl verwendet werden können, finden Sie unter. sam remote test-event delete Sie können Folgendes auch über den folgenden Befehl ausführen: AWS SAM CLI

```
$ sam remote test-event delete --help
```

Testereignisse abrufen, die gemeinsam genutzt werden können

Sie können ein gemeinsam nutzbares Testereignis aus der EventBridge Schemaregistrierung abrufen, indem Sie den `get` Unterbefehl zusammen mit dem folgenden Befehl verwenden:

- Geben Sie den Namen des gemeinsam nutzbaren Testereignisses ein, das abgerufen werden soll.
- Geben Sie eine akzeptable ID der Lambda-Funktion an, die dem Ereignis zugeordnet ist.
- Wenn Sie die logische ID der Lambda-Funktion angeben, müssen Sie auch den AWS CloudFormation Stacknamen angeben, der der Lambda-Funktion zugeordnet ist.

Im folgenden Beispiel wird ein gemeinsam nutzbares Testereignis benanntdemo-event, das der `HelloWorldFunction` Lambda-Funktion des `sam-app` Stacks zugeordnet ist. Mit diesem Befehl wird das Ereignis auf Ihrer Konsole gedruckt.

```
$ sam remote test-event get HelloWorldFunction --stack-name sam-app --name demo-event
```

Um ein gemeinsam nutzbares Testereignis abzurufen und es auf Ihrem lokalen Computer zu speichern, verwenden Sie die `--output-file` Option und geben Sie einen Dateipfad und einen Namen an. Im Folgenden finden Sie ein Beispiel, das `demo-event` wie `demo-event.json` im aktuellen Arbeitsverzeichnis gespeichert wird:

```
$ sam remote test-event get HelloWorldFunction --stack-name sam-app --name demo-event
 --output-file demo-event.json
```

Eine Liste der Optionen, die mit dem `get` Unterbefehl verwendet werden können, finden Sie untersam remote test-event get. Sie können Folgendes auch über den folgenden Befehl ausführen: AWS SAM CLI

```
$ sam remote test-event get --help
```

Testereignisse auflisten, die gemeinsam genutzt werden können

Sie können alle gemeinsam nutzbaren Testereignisse für eine bestimmte Lambda-Funktion aus der Schemaregistrierung auflisten. Verwenden Sie den `list` Unterbefehl zusammen mit dem Folgenden:

- Geben Sie eine akzeptable ID der Lambda-Funktion an, die den Ereignissen zugeordnet ist.
- Wenn Sie die logische ID der Lambda-Funktion angeben, müssen Sie auch den CloudFormation Stacknamen angeben, der der Lambda-Funktion zugeordnet ist.

Im Folgenden finden Sie ein Beispiel, das eine Liste aller gemeinsam nutzbaren Testereignisse abruft, die mit der `HelloWorldFunction` Lambda-Funktion des Stacks verknüpft sind: `sam-app`

```
$ sam remote test-event list HelloWorldFunction --stack-name sam-app
```

Eine Liste der Optionen, die mit dem `list` Unterbefehl verwendet werden können, finden Sie unter. sam remote test-event list Sie können Folgendes auch über den folgenden Befehl ausführen: AWS SAM CLI

```
$ sam remote test-event list --help
```

Testereignisse speichern, die gemeinsam genutzt werden können

Sie können gemeinsam nutzbare Testereignisse in der EventBridge Schemaregistrierung speichern. Verwenden Sie den `put` Unterbefehl zusammen mit dem Folgenden:

- Geben Sie eine akzeptable ID der Lambda-Funktion an, die dem gemeinsam nutzbaren Testereignis zugeordnet ist.
- Geben Sie einen Namen für das gemeinsam nutzbare Testereignis an.
- Geben Sie den Dateipfad und den Namen des lokalen Ereignisses an, das hochgeladen werden soll.

Das folgende Beispiel speichert das lokale `demo-event.json` Ereignis unter `demo-event` und verknüpft es mit der `HelloWorldFunction` Lambda-Funktion des `sam-app` Stacks:

```
$ sam remote test-event put HelloWorldFunction --stack-name sam-app --name demo-event
 --file demo-event.json
```

Wenn ein gemeinsam nutzbares Testereignis mit demselben Namen in der EventBridge Schemaregistrierung vorhanden ist, AWS SAM CLI wird es nicht überschrieben. Um zu überschreiben, fügen Sie die `--force` Option zu Ihrem Befehl hinzu.

Eine Liste der Optionen, die Sie mit dem `put` Unterbefehl verwenden können, finden Sie unter. sam remote test-event put Sie können Folgendes auch über den folgenden Befehl ausführen: AWS SAM CLI

```
$ sam remote test-event put --help
```

Testereignisse verwenden, die gemeinsam genutzt werden können

Verwenden Sie gemeinsam nutzbare Testereignisse, um Ihre Lambda-Funktionen AWS Cloud mit dem `sam remote invoke` Befehl zu testen. Weitere Informationen hierzu finden Sie unter Übergeben Sie gemeinsam nutzbare Testereignisse an eine Lambda-Funktion in der Cloud.

Verwaltung gemeinsam nutzbarer Testereignisse

Dieses Thema enthält Beispiele dafür, wie Sie gemeinsam nutzbare Testereignisse verwalten und verwenden können.

Holen Sie sich ein gemeinsam nutzbares Testereignis, ändern Sie es und verwenden Sie es

Sie können ein gemeinsam nutzbares Testereignis aus der EventBridge Schemaregistrierung abrufen, es lokal ändern und das lokale Testereignis mit Ihrer Lambda-Funktion in der verwenden. AWS Cloud Im Folgenden wird ein Beispiel gezeigt:

1. Das gemeinsam nutzbare Testereignis abrufen — Verwenden Sie den `sam remote test-event get` Unterbefehl, um ein gemeinsam nutzbares Testereignis für eine bestimmte Lambda-Funktion abzurufen und lokal zu speichern:

   ```
   $ sam remote test-event get HelloWorldFunction --stack-name sam-app --name demo-event
    --output-file demo-event.json
   ```

2. Das gemeinsam nutzbare Testereignis ändern — Verwenden Sie einen Texteditor Ihrer Wahl, um das gemeinsam nutzbare Testereignis zu ändern.
3. Das gemeinsam nutzbare Testereignis verwenden — Verwenden Sie den `sam remote invoke` Befehl und geben Sie den Dateipfad und den Namen des Ereignisses mit: `--event-file`

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --event-file demo-
event.json
```

Holen Sie sich ein gemeinsam nutzbares Testereignis, ändern Sie es, laden Sie es hoch und verwenden Sie es

Sie können ein gemeinsam nutzbares Testereignis aus der EventBridge Schemaregistrierung abrufen, es lokal ändern und hochladen. Anschließend können Sie das gemeinsam nutzbare Testereignis direkt an Ihre Lambda-Funktion in übergeben. AWS Cloud Im Folgenden wird ein Beispiel gezeigt:

1. Das gemeinsam nutzbare Testereignis abrufen — Verwenden Sie den `sam remote test-event get` Unterbefehl, um ein gemeinsam nutzbares Testereignis für eine bestimmte Lambda-Funktion abzurufen und lokal zu speichern:

   ```
   $ sam remote test-event get HelloWorldFunction --stack-name sam-app --name demo-event
    --output-file demo-event.json
   ```

2. Das gemeinsam nutzbare Testereignis ändern — Verwenden Sie einen Texteditor Ihrer Wahl, um das gemeinsam nutzbare Testereignis zu ändern.

3. Das gemeinsam nutzbare Testereignis hochladen — Verwenden Sie den `sam remote test-event put` Unterbefehl, um das gemeinsam nutzbare Testereignis hochzuladen und in der Schemaregistrierung zu speichern. EventBridge In diesem Beispiel verwenden wir die `--force` Option, um eine ältere Version unseres gemeinsam nutzbaren Tests zu überschreiben:

   ```
   $ sam remote test-event put HelloWorldFunction --stack-name sam-app --name demo-event
    --file demo-event.json --force
   ```

4. Übergeben Sie das gemeinsam nutzbare Testereignis an Ihre Lambda-Funktion — Verwenden Sie den `sam remote invoke` Befehl, um das gemeinsam nutzbare Testereignis direkt an Ihre Lambda-Funktion zu übergeben in: AWS Cloud

   ```
   $ sam remote invoke HelloWorldFunction --stack-name sam-app --test-event-name demo-
   event
   ```

Einführung in das Testen in der Cloud mit sam remote invoke

Verwenden Sie den AWS Serverless Application Model Befehl Command Line Interface (AWS SAM CLI)`sam remote invoke`, um mit unterstützten AWS Ressourcen in der zu interagieren AWS Cloud. Sie können `sam remote invoke` es verwenden, um die folgenden Ressourcen aufzurufen:

- Amazon Kinesis Data Streams — Senden Sie Datensätze an Kinesis Data Streams Streams-Anwendungen.
- AWS Lambda— Rufen Sie Ereignisse auf und übergeben Sie sie an Ihre Lambda-Funktionen.
- Amazon Simple Queue Service (Amazon SQS) — Nachrichten an Amazon SQS SQS-Warteschlangen senden.
- AWS Step Functions— Ruft Step Functions Functions-Zustandsmaschinen auf, um die Ausführung zu starten.

Eine Einführung in die finden Sie AWS SAMCLI unter Was ist der AWS SAM CLI?

Ein Beispiel für die Verwendung `sam remote invoke` während eines typischen Entwicklungsworkflows finden Sie unterSchritt 5: Interagieren Sie mit Ihrer Funktion im AWS Cloud.

Topics

- Verwenden Sie den Befehl sam remote invoke
- Verwenden von Sam Remote Invoke-Befehlsoptionen
- Konfigurieren Sie Ihre Projektkonfigurationsdatei
- Beispiele
- Weiterführende Links

Voraussetzungen

Um das zu verwendensam `remote invoke`, installieren Sie das, AWS SAMCLI indem Sie wie folgt vorgehen:

- AWS SAM Voraussetzungen.
- Installieren Sie das AWS SAMCLI.

Wir empfehlen außerdem, auf die neueste Version von zu aktualisieren AWS SAMCLI. Weitere Informationen hierzu finden Sie unter Aktualisierung des AWS SAMCLI.

Vor der Verwendung empfehlen wir`sam remote invoke`, sich mit folgenden Grundkenntnissen vertraut zu machen:

- Konfiguration der AWS SAMCLI.
- Erstellen Sie Ihre Bewerbung in AWS SAM.
- Einführung in das Bauen mit AWS SAM.
- Einführung in die Bereitstellung mit AWS SAM.
- Einführung in die Verwendung von sam sync to sync to AWS Cloud.

Verwenden Sie den Befehl sam remote invoke

Bevor Sie diesen Befehl verwenden können, muss Ihre Ressource auf dem AWS Cloud bereitgestellt werden.

Verwenden Sie die folgende Befehlsstruktur und führen Sie sie im Stammverzeichnis Ihres Projekts aus:

```
$ sam remote invoke <arguments> <options>
```

> **Note**
>
> Auf dieser Seite werden Optionen angezeigt, die an der Befehlszeile bereitgestellt werden. Sie können Optionen auch in der Konfigurationsdatei Ihres Projekts konfigurieren, anstatt sie an der Befehlszeile zu übergeben. Weitere Informationen hierzu finden Sie unter Konfigurieren Sie die Projekteinstellungen.

Eine Beschreibung der `sam remote invoke` Argumente und Optionen finden Sie untersam remote invoke.

Verwendung mit Kinesis Data Streams

Sie können Datensätze an eine Kinesis Data Streams Streams-Anwendung senden. Sie AWS SAM CLI sendet Ihren Datensatz und gibt eine Shard-ID und eine Sequenznummer zurück. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam remote invoke KinesisStream --stack-name kinesis-example --event hello-world

 Putting record to Kinesis data stream KinesisStream

 Auto converting value 'hello-world' into JSON '"hello-world"'. If you don't want auto-
conversion, please provide
 a JSON string as event

 {
   "ShardId": "shardId-000000000000",
   "SequenceNumber": "49646251411914806775980850790050483811301135051202232322"
 }%
```

Um einen Datensatz zu senden

1. Geben Sie einen Ressourcen-ID-Wert als Argument für Ihre Kinesis Data Streams Streams-Anwendung an. Informationen zu gültigen Ressourcen finden Sie IDs unter Ressourcen-ID.
2. Stellen Sie den Datensatz als Ereignis bereit, das an Ihre Kinesis Data Streams Streams-Anwendung gesendet werden soll. Sie können das Ereignis mit der `--event` Option in der Befehlszeile oder mithilfe von aus einer Datei angeben. `--event-file` Wenn Sie kein Ereignis angeben, AWS SAM CLI sendet das ein leeres Ereignis.

Verwendung mit Lambda-Funktionen

Sie können eine Lambda-Funktion in der Cloud aufrufen und ein leeres Ereignis übergeben oder ein Ereignis in der Befehlszeile oder aus einer Datei bereitstellen. Das ruft AWS SAM CLI Ihre Lambda-Funktion auf und gibt ihre Antwort zurück. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app

Invoking Lambda Function HelloWorldFunction
START RequestId: d5ef494b-5f45-4086-86fd-d7322fa1a1f9 Version: $LATEST
END RequestId: d5ef494b-5f45-4086-86fd-d7322fa1a1f9
REPORT RequestId: d5ef494b-5f45-4086-86fd-d7322fa1a1f9  Duration: 6.62 ms       Billed
 Duration: 7 ms     Memory Size: 128 MB     Max Memory Used: 67 MB  Init Duration:
 164.06 ms
{"statusCode":200,"body":"{\"message\":\"hello world\"}"}%
```

Um eine Lambda-Funktion aufzurufen

1. Geben Sie einen Ressourcen-ID-Wert als Argument für Ihre Lambda-Funktion an. Informationen zu gültigen Ressourcen finden Sie IDs unter Ressourcen-ID.
2. Geben Sie ein Ereignis an, das an Ihre Lambda-Funktion gesendet werden soll. Sie können das Ereignis in der Befehlszeile mit der `--event` Option oder aus einer Datei mit `--event-file` angeben. Wenn Sie kein Ereignis angeben, AWS SAM CLI sendet das ein leeres Ereignis.

Mit Response-Streaming konfigurierte Lambda-Funktionen

Der `sam remote invoke` Befehl unterstützt Lambda-Funktionen, die für das Streamen von Antworten konfiguriert sind. Sie können eine Lambda-Funktion so konfigurieren, dass sie Antworten streamt, indem Sie die `FunctionUrlConfig` Eigenschaft in Ihren AWS SAM Vorlagen verwenden. Wenn Sie es verwendensam `remote invoke`, erkennt das AWS SAMCLI automatisch Ihre Lambda-Konfiguration und wird mit Antwort-Streaming aufgerufen.

Ein Beispiel finden Sie unter Rufen Sie eine Lambda-Funktion auf, die für das Streamen von Antworten konfiguriert ist.

Übergeben Sie gemeinsam nutzbare Testereignisse an eine Lambda-Funktion in der Cloud

Gemeinsam nutzbare Testereignisse sind Testereignisse, die Sie mit anderen teilen können. AWS-Konto Weitere Informationen finden Sie unter Gemeinsam nutzbare Testereignisse im AWS Lambda Entwicklerhandbuch.

Auf gemeinsam nutzbare Testereignisse zugreifen und diese verwalten

Sie können den AWS SAM CLI `sam remote test-event` Befehl verwenden, um auf gemeinsam nutzbare Testereignisse zuzugreifen und diese zu verwalten. Sie können ihn beispielsweise für Folgendes verwendensam `remote test-event`:

- Rufen Sie gemeinsam nutzbare Testereignisse aus der EventBridge Amazon-Schemaregistry ab.
- Ändern Sie gemeinsam nutzbare Testereignisse lokal und laden Sie sie in die EventBridge Schemaregistrierung hoch.
- Löschen Sie gemeinsam nutzbare Testereignisse aus der EventBridge Schemaregistrierung.

Weitere Informationen hierzu finden Sie unter Einführung in Cloud-Tests mit sam remote test-event.

Übergeben Sie ein gemeinsam nutzbares Testereignis an eine Lambda-Funktion in der Cloud

Um ein gemeinsam nutzbares Testereignis aus der EventBridge Schemaregistrierung an Ihre Lambda-Funktion in der Cloud zu übergeben, verwenden Sie die `--test-event-name` Option und geben Sie den Namen des gemeinsam nutzbaren Testereignisses an. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --test-event-name demo-event
```

Wenn Sie das gemeinsam nutzbare Testereignis lokal speichern, können Sie die `--event-file` Option verwenden und den Dateipfad und den Namen des lokalen Testereignisses angeben. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --event-file demo-event.json
```

Using with Amazon SQS

Sie können Nachrichten an Amazon SQS SQS-Warteschlangen senden. Das AWS SAM CLI gibt Folgendes zurück:

- Nachrichten-ID
- MD5 des Nachrichtentexts
- Metadaten der Antwort

Im Folgenden wird ein Beispiel gezeigt:

```
$ sam remote invoke MySqsQueue --stack-name sqs-example -event hello

Sending message to SQS queue MySqsQueue

{
  "MD5OfMessageBody": "5d41402abc4b2a76b9719d911017c592",
  "MessageId": "05c7af65-9ae8-4014-ae28-809d6d8ec652"
}%
```

So senden Sie eine Nachricht

1. Geben Sie einen Ressourcen-ID-Wert als Argument für die Amazon SQS SQS-Warteschlange an. Informationen zu gültigen Ressourcen finden Sie IDs unter Ressourcen-ID.
2. Geben Sie ein Ereignis an, das an Ihre Amazon SQS SQS-Warteschlange gesendet werden soll. Sie können das Ereignis in der Befehlszeile mit der `--event` Option oder aus einer Datei mithilfe `--event-file` von. Wenn Sie kein Ereignis angeben, AWS SAM CLI sendet das ein leeres Ereignis.

Mit Step-Funktionen verwenden

Sie können eine Step Functions Functions-Zustandsmaschine aufrufen, um die Ausführung zu starten. Der AWS SAM CLI wartet, bis der State-Machine-Workflow abgeschlossen ist, und gibt eine Ausgabe des letzten Schritts der Ausführung zurück. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam remote invoke HelloWorldStateMachine --stack-name state-machine-example --
event '{"is_developer": true}'

Invoking Step Function HelloWorldStateMachine

"Hello Developer World"%
```

Um eine Zustandsmaschine aufzurufen

1. Geben Sie einen Ressourcen-ID-Wert als Argument für die Step Functions Functions-Zustandsmaschine an. Informationen zu einer gültigen Ressource IDs finden Sie unter Ressourcen-ID.
2. Geben Sie ein Ereignis an, das an Ihre Zustandsmaschine gesendet werden soll. Sie können das Ereignis mit der `--event` Option in der Befehlszeile oder mithilfe von einer Datei bereitstellen`--event-file`. Wenn Sie kein Ereignis angeben, AWS SAM CLI sendet das ein leeres Ereignis.

Verwenden von Sam Remote Invoke-Befehlsoptionen

Dieser Abschnitt behandelt einige der wichtigsten Optionen, die Sie mit dem `sam remote invoke` Befehl verwenden können. Eine vollständige Liste der Optionen finden Sie untersam remote invoke.

Übergeben Sie ein Ereignis an Ihre Ressource

Verwenden Sie die folgenden Optionen, um Ereignisse an Ihre Ressourcen in der Cloud weiterzuleiten:

- `--event`— Übergibt ein Ereignis an der Befehlszeile.
- `--event-file`— Übergibt ein Ereignis aus einer Datei.

Beispiele für Lambda

Wird verwendet**`--event`**, um ein Ereignis an der Befehlszeile als Zeichenkettenwert zu übergeben:

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --event '{"message":
 "hello!"}'

Invoking Lambda Function HelloWorldFunction


START RequestId: b992292d-1fac-4aa2-922a-c9dc5c6fceab Version: $LATEST
END RequestId: b992292d-1fac-4aa2-922a-c9dc5c6fceab
REPORT RequestId: b992292d-1fac-4aa2-922a-c9dc5c6fceab  Duration: 16.41 ms      Billed
 Duration: 17 ms  Memory Size: 128 MB     Max Memory Used: 67 MB  Init Duration: 185.96
 ms
{"statusCode":200,"body":"{\"message\":\"hello!\"}"}%
```

Wird verwendet**`--event-file`**, um ein Ereignis aus einer Datei zu übergeben und den Pfad zur Datei anzugeben:

```
$ cat event.json

{"message": "hello from file"}%

$ sam remote invoke HelloWorldFunction --stack-name sam-app --event-file event.json


Invoking Lambda Function HelloWorldFunction


START RequestId: 3bc71f7d-153a-4b1e-8c9a-901d91b1bec9 Version: $LATEST
END RequestId: 3bc71f7d-153a-4b1e-8c9a-901d91b1bec9
REPORT RequestId: 3bc71f7d-153a-4b1e-8c9a-901d91b1bec9  Duration: 21.15 ms      Billed
 Duration: 22 ms  Memory Size: 128 MB     Max Memory Used: 67 MB
```

```
{"statusCode":200,"body":"{\"message\":\"hello from file\"}"}%
```

Übergeben Sie ein Ereignis mit**`stdin`**:

```
$ cat event.json

{"message": "hello from file"}%

$ cat event.json | sam remote invoke HelloWorldFunction --stack-name sam-app --event-
file -

Reading event from stdin (you can also pass it from file with --event-file)

Invoking Lambda Function HelloWorldFunction

START RequestId: 85ecc902-8ad0-4a2b-a8c8-9bb4f65f5a7a Version: $LATEST
END RequestId: 85ecc902-8ad0-4a2b-a8c8-9bb4f65f5a7a
REPORT RequestId: 85ecc902-8ad0-4a2b-a8c8-9bb4f65f5a7a  Duration: 1.36 ms       Billed
 Duration: 2 ms   Memory Size: 128 MB       Max Memory Used: 67 MB
{"statusCode":200,"body":"{\"message\":\"hello from file\"}"}%
```

Konfigurieren Sie die AWS SAMCLI Antwortausgabe

Wenn Sie eine unterstützte Ressource mit aufrufen`sam remote invoke`, gibt sie eine Antwort AWS SAMCLI zurück, die Folgendes enthält:

- Metadaten anfordern — Mit der Anfrage verknüpfte Metadaten. Dazu gehören eine Anforderungs-ID und eine Startzeit der Anfrage.
- Ressourcenantwort — Die Antwort Ihrer Ressource nach dem Aufruf in der Cloud.

Sie können die `--output` Option verwenden, um die AWS SAM CLI Ausgabeantwort zu konfigurieren. Die folgenden Optionswerte sind verfügbar:

- `json`— Metadaten und Ressourcenantwort werden in einer JSON Struktur zurückgegeben. Die Antwort enthält die vollständige SDK Ausgabe.
- `text`— Metadaten werden in Textstruktur zurückgegeben. Die Ressourcenantwort wird im Ausgabeformat der Ressource zurückgegeben.

Das Folgende ist ein Beispiel für eine `json` Ausgabe:

```
$ sam remote invoke --stack-name sam-app --output json

Invoking Lambda Function HelloWorldFunction



{
  "ResponseMetadata": {
    "RequestId": "3bdf9a30-776d-4a90-94a6-4cccc0fc7b41",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "date": "Mon, 19 Jun 2023 17:15:46 GMT",
      "content-type": "application/json",
      "content-length": "57",
      "connection": "keep-alive",
      "x-amzn-requestid": "3bdf9a30-776d-4a90-94a6-4cccc0fc7b41",
      "x-amzn-remapped-content-length": "0",
      "x-amz-executed-version": "$LATEST",
      "x-amz-log-result":
 "U1RBUlQgUmVxdWVzdElkOiAzYmRmOWEzMC03NzZkLTRhOTAtOTRhNi00Y2NjYzBmYzdiNDEgVmVyc2lvbjogJExBVEVTV
      "x-amzn-trace-id":
 "root=1-64908d42-17dab270273fcc6b527dd6b8;sampled=0;lineage=2301f8dc:0"
    },
    "RetryAttempts": 0
  },
  "StatusCode": 200,
  "LogResult":
 "U1RBUlQgUmVxdWVzdElkOiAzYmRmOWEzMC03NzZkLTRhOTAtOTRhNi00Y2NjYzBmYzdiNDEgVmVyc2lvbjogJExBVEVTV
  "ExecutedVersion": "$LATEST",
  "Payload": "{\"statusCode\":200,\"body\":\"{\\\"message\\\":\\\"hello world\\\"}\"}"
}%
```

Wenn Sie eine `json` Ausgabe angeben, wird die gesamte Antwort zurückgegeben`stdout`. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam remote invoke --stack-name sam-app --output json 1> stdout.log

Invoking Lambda Function HelloWorldFunction



$ cat stdout.log

{
```

```
  "ResponseMetadata": {
    "RequestId": "d30d280f-8188-4372-bc94-ce0f1603b6bb",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "date": "Mon, 19 Jun 2023 17:35:56 GMT",
      "content-type": "application/json",
      "content-length": "57",
      "connection": "keep-alive",
      "x-amzn-requestid": "d30d280f-8188-4372-bc94-ce0f1603b6bb",
      "x-amzn-remapped-content-length": "0",
      "x-amz-executed-version": "$LATEST",
      "x-amz-log-result":
 "U1RBUlQgUmVxdWVzdElkOiBkMzBkMjgwZi04MTg4LTQzNzItYmM5NC1jZTBmMTYwM2I2YmIgVmVyc2lvbjogJExBVEVTV
      "x-amzn-trace-id":
 "root=1-649091fc-771473c7778689627a6122b7;sampled=0;lineage=2301f8dc:0"
    },
    "RetryAttempts": 0
  },
  "StatusCode": 200,
  "LogResult":
 "U1RBUlQgUmVxdWVzdElkOiBkMzBkMjgwZi04MTg4LTQzNzItYmM5NC1jZTBmMTYwM2I2YmIgVmVyc2lvbjogJExBVEVTV
  "ExecutedVersion": "$LATEST",
  "Payload": "{\"statusCode\":200,\"body\":\"{\\\"message\\\":\\\"hello world\\\"}\"}"
}%
```

Im Folgenden finden Sie ein Beispiel für eine `text` Ausgabe:

```
$ sam remote invoke --stack-name sam-app --output text

Invoking Lambda Function HelloWorldFunction


START RequestId: 4dbacc43-1ec6-47c2-982b-9dc4620144d6 Version: $LATEST
END RequestId: 4dbacc43-1ec6-47c2-982b-9dc4620144d6
REPORT RequestId: 4dbacc43-1ec6-47c2-982b-9dc4620144d6  Duration: 9.13 ms       Billed
 Duration: 10 ms  Memory Size: 128 MB     Max Memory Used: 67 MB  Init Duration: 165.50
 ms
{"statusCode":200,"body":"{\"message\":\"hello world\"}"}%
```

Wenn Sie eine `text` Ausgabe angeben, wird die Laufzeitausgabe der Lambda-Funktion (z. B. Logs) zurückgegeben. `stderr` Die Nutzlast der Lambda-Funktion wird zurückgegeben. `stdout` Im Folgenden wird ein Beispiel gezeigt:

```
$ sam remote invoke --stack-name sam-app --output text 2> stderr.log

{"statusCode":200,"body":"{\"message\":\"hello world\"}"}%



$ cat stderr.log

Invoking Lambda Function HelloWorldFunction
START RequestId: 82273c3b-aa3a-4d16-8f1c-1d2ad3ace891 Version: $LATEST
END RequestId: 82273c3b-aa3a-4d16-8f1c-1d2ad3ace891
REPORT RequestId: 82273c3b-aa3a-4d16-8f1c-1d2ad3ace891  Duration: 40.62 ms      Billed
 Duration: 41 ms  Memory Size: 128 MB     Max Memory Used: 68 MB

$ sam remote invoke --stack-name sam-app --output text 1> stdout.log

Invoking Lambda Function HelloWorldFunction


START RequestId: 74acaa9f-5b80-4a5c-b3b8-ffaccb84cbbd Version: $LATEST
END RequestId: 74acaa9f-5b80-4a5c-b3b8-ffaccb84cbbd
REPORT RequestId: 74acaa9f-5b80-4a5c-b3b8-ffaccb84cbbd  Duration: 2.31 ms       Billed
 Duration: 3 ms   Memory Size: 128 MB     Max Memory Used: 67 MB

$ cat stdout.log

{"statusCode":200,"body":"{\"message\":\"hello world\"}"}%
```

Passen Sie die Parameter an Boto3

Denn `sam remote invoke` AWS SAM CLI verwendet das AWS SDK for Python (Boto3), um mit Ihren Ressourcen in der Cloud zu interagieren. Sie können die `--parameter` Option verwenden, um Parameter anzupassen. Boto3 Eine Liste der unterstützten Parameter, die Sie anpassen können, finden Sie unter`--parameter`.

Beispiele

Rufen Sie eine Lambda-Funktion auf, um Parameterwerte zu validieren und Berechtigungen zu überprüfen:

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --
parameter InvocationType="DryRun"
```

Verwenden Sie die **`--parameter`** Option mehrmals in einem einzigen Befehl, um mehrere Parameter bereitzustellen:

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --
parameter InvocationType="Event" --parameter LogType="None"
```

Andere Optionen

Eine vollständige Liste der `sam remote invoke` Optionen finden Sie untersam remote invoke.

Konfigurieren Sie Ihre Projektkonfigurationsdatei

Um `sam remote invoke` in Ihrer Konfigurationsdatei zu konfigurieren, verwenden Sie `remote_invoke` in Ihrer Tabelle. Im Folgenden finden Sie ein Beispiel für eine `samconfig.toml` Datei, in der Standardwerte für den `sam remote invoke` Befehl konfiguriert werden.

```
...
version =0.1

[default]
...
[default.remote_invoke.parameters]
stack_name = "cloud-app"
event = '{"message": "Hello!"}'
```

Beispiele

Ein grundlegendes Anwendungsbeispiel finden Sie unter AWS Lambda Funktionen mit AWS SAM Remote testen im AWS Compute-Blog. `sam remote invoke`

Beispiele für Kinesis Data Streams

Grundlegende Beispiele

Senden Sie einen Datensatz aus einer Datei an eine Kinesis Data Streams Streams-Anwendung. Die Kinesis Data Streams Streams-Anwendung wird identifiziert, indem ein ARN für die Ressourcen-ID bereitgestellt wird:

```
$ sam remote invoke arn:aws:kinesis:us-west-2:01234567890:stream/kinesis-example-
KinesisStream-BgnLcAey4xUQ --event-file event.json
```

Senden Sie ein in der Befehlszeile bereitgestelltes Ereignis an eine Kinesis Data Streams Streams-Anwendung:

```
$ sam remote invoke KinesisStream --stack-name kinesis-example --event hello-world

Putting record to Kinesis data stream KinesisStream

Auto converting value 'hello-world' into JSON '"hello-world"'. If you don't want auto-
conversion, please provide
a JSON string as event

{
  "ShardId": "shardId-000000000000",
  "SequenceNumber": "49646251411914806775980903986194508740483329854174920706"
}%
```

Rufen Sie die physische ID der Kinesis Data Streams Streams-Anwendung ab. Geben Sie dann ein Ereignis in der Befehlszeile ein:

```
$ sam list resources --stack-name kinesis-example --output json

[
  {
    "LogicalResourceId": "KinesisStream",
    "PhysicalResourceId": "kinesis-example-KinesisStream-ZgnLcQey4xUQ"
  }
]

$ sam remote invoke kinesis-example-KinesisStream-ZgnLcQey4xUQ --event hello

Putting record to Kinesis data stream KinesisStream

Auto converting value 'hello' into JSON '"hello"'. If you don't want auto-conversion,
 please provide a JSON
string as event

{
  "ShardId": "shardId-000000000000",
  "SequenceNumber": "49646251411914806775980904340716841045751814812900261890"
}%
```

Geben Sie in der Befehlszeile eine JSON-Zeichenfolge als Ereignis an:

```
$ sam remote invoke KinesisStream --stack-name kinesis-example --event '{"method":
 "GET", "body": ""}'

Putting record to Kinesis data stream KinesisStream

{
  "ShardId": "shardId-000000000000",
  "SequenceNumber": "49646251411914806775980904492868617924990209230536441858"
}%
```

Senden Sie ein leeres Ereignis an die Kinesis Data Streams Streams-Anwendung:

```
$ sam remote invoke KinesisStream --stack-name kinesis-example

Putting record to Kinesis data stream KinesisStream

{
  "ShardId": "shardId-000000000000",
  "SequenceNumber": "49646251411914806775980904866469008589597168190416224258"
}%
```

Gibt die AWS SAM CLI Antwort im JSON-Format zurück:

```
$ sam remote invoke KinesisStream --stack-name kinesis-example --event '{"hello":
 "world"}' --output json

Putting record to Kinesis data stream KinesisStream

{
  "ShardId": "shardId-000000000000",
  "SequenceNumber": "49646251411914806775980905078409420803696667195489648642",
  "ResponseMetadata": {
    "RequestId": "ebbbd307-3e9f-4431-b67c-f0715e9e353e",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "x-amzn-requestid": "ebbbd307-3e9f-4431-b67c-f0715e9e353e",
      "x-amz-id-2": "Q3yBcgTwtPaQTV26IKclbECmZikUYOzKY+CzcxA84ZHgCkc5T2N/
ITWg6RPOQcWw8Gn0tNPcEJBEHyVVqboJAPgCritqsvCu",
      "date": "Thu, 09 Nov 2023 18:13:10 GMT",
      "content-type": "application/x-amz-json-1.1",
      "content-length": "110"
    },
```

```
    "RetryAttempts": 0
  }
}%
```

Geben Sie die JSON-Ausgabe an stdout zurück:

```
$ sam remote invoke KinesisStream --stack-name kinesis-example --event '{"hello":
 "world"}' --output json 1> stdout.log

Putting record to Kinesis data stream KinesisStream


$ cat stdout.log
{
  "ShardId": "shardId-000000000000",
  "SequenceNumber": "49646251411914806775980906397777867595039988349006774274",
  "ResponseMetadata": {
    "RequestId": "f4290006-d84b-b1cd-a9ee-28306eeb2939",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "x-amzn-requestid": "f4290006-d84b-b1cd-a9ee-28306eeb2939",
      "x-amz-id-2": "npCqz
+IBKpoL4sQ1ClbUmxuJlbeA24Fx1UgpIrS6mm2NoIeV2qdZSN5AhNurdssykXajBrXaC9anMhj2eG/h7Hnbf
+bPuotU",
      "date": "Thu, 09 Nov 2023 18:33:26 GMT",
      "content-type": "application/x-amz-json-1.1",
      "content-length": "110"
    },
    "RetryAttempts": 0
  }
}%
```

Beispiele für Lambda

Grundlegende Beispiele

Rufen Sie eine Lambda-Funktion auf, indem Sie den ARN als Ressourcen-ID angeben:

```
$ sam remote invoke arn:aws:lambda:us-west-2:012345678910:function:sam-app-
HelloWorldFunction-ohRFEn2RuAvp
```

Rufen Sie eine Lambda-Funktion auf, indem Sie die logische ID als Ressourcen-ID angeben:

Sie müssen auch den CloudFormation Stack-Namen mithilfe der `--stack-name` Option angeben. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app
```

Wenn Ihre Anwendung eine einzelne Lambda-Funktion enthält, müssen Sie deren logische ID nicht angeben. Sie können nur die `--stack-name` Option angeben. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam remote invoke --stack-name sam-app
```

Rufen Sie eine Lambda-Funktion auf, indem Sie die physische ID als Ressourcen-ID angeben:

Die physische ID wird erstellt, wenn Sie sie bereitstellen mit. CloudFormation

```
$ sam remote invoke sam-app-HelloWorldFunction-TZvxQRFNv0k4
```

Rufen Sie eine Lambda-Funktion eines untergeordneten Stacks auf:

In diesem Beispiel enthält unsere Anwendung die folgende Verzeichnisstruktur:

```
lambda-example
### childstack
#   ### function
#   #   ### __init__.py
#   #   ### app.py
#   #   ### requirements.txt
#   ### template.yaml
### events
#   ### event.json
### samconfig.toml
### template.yaml
```

Um die Lambda-Funktion von our aufzurufen`childstack`, führen wir Folgendes aus:

```
$ sam remote invoke ChildStack/HelloWorldFunction --stack-name lambda-example

Invoking Lambda Function HelloWorldFunction

START RequestId: 207a864b-e67c-4307-8478-365b004d4bcd Version: $LATEST
```

```
END RequestId: 207a864b-e67c-4307-8478-365b004d4bcd
REPORT RequestId: 207a864b-e67c-4307-8478-365b004d4bcd  Duration: 1.27 ms       Billed
 Duration: 2 ms   Memory Size: 128 MB     Max Memory Used: 36 MB  Init Duration: 111.07
 ms
{"statusCode": 200, "body": "{\"message\": \"Hello\", \"received_event\": {}}"}%
```

Rufen Sie eine Lambda-Funktion auf, die für das Streamen von Antworten konfiguriert ist

In diesem Beispiel verwenden wir die, um eine neue serverlose Anwendung AWS SAMCLI zu initialisieren, die eine Lambda-Funktion enthält, die so konfiguriert ist, dass sie ihre Antwort streamt. Wir stellen unsere Anwendung in der Cloud bereit AWS Cloud und verwenden `sam remote invoke` sie, um mit unserer Funktion in der Cloud zu interagieren.

Wir beginnen damit, den `sam init` Befehl auszuführen, um eine neue serverlose Anwendung zu erstellen. Wir wählen die Lambda Response Streaming-Schnellstartvorlage aus und geben unserer Anwendung lambda-streaming-nodejs-appeinen Namen.

```
$ sam init

 You can preselect a particular runtime or package type when using the `sam init`
 experience.
 Call `sam init --help` to learn more.

 Which template source would you like to use?
        1 - AWS Quick Start Templates
        2 - Custom Template Location
 Choice: 1

 Choose an AWS Quick Start application template
        1 - Hello World Example
        ...
        9 - Lambda Response Streaming
        ...
        15 - Machine Learning
 Template: 9

 Which runtime would you like to use?
        1 - go (provided.al2)
        2 - nodejs18.x
        3 - nodejs16.x
 Runtime: 2

 Based on your selections, the only Package type available is Zip.
```

```
 We will proceed to selecting the Package type as Zip.

 Based on your selections, the only dependency manager available is npm.
 We will proceed copying the template using npm.

 Would you like to enable X-Ray tracing on the function(s) in your application?  [y/
N]: ENTER

 Would you like to enable monitoring using CloudWatch Application Insights?
 For more info, please view https://docs.aws.amazon.com/AmazonCloudWatch/latest/
monitoring/cloudwatch-application-insights.html [y/N]: ENTER

 Project name [sam-app]: lambda-streaming-nodejs-app

     -----------------------
     Generating application:
     -----------------------
     Name: lambda-streaming-nodejs-app
     Runtime: nodejs18.x
     Architectures: x86_64
     Dependency Manager: npm
     Application Template: response-streaming
     Output Directory: .
     Configuration file: lambda-streaming-nodejs-app/samconfig.toml

     Next steps can be found in the README file at lambda-streaming-nodejs-app/
README.md


 Commands you can use next
 =========================
 [*] Create pipeline: cd lambda-streaming-nodejs-app && sam pipeline init --bootstrap
 [*] Validate SAM template: cd lambda-streaming-nodejs-app && sam validate
 [*] Test Function in the Cloud: cd lambda-streaming-nodejs-app && sam sync --stack-
name {stack-name} --watch
```

Das AWS SAMCLI erstellt unser Projekt mit der folgenden Struktur:

```
lambda-streaming-nodejs-app
 ### README.md
 ### __tests__
 #   ### unit
 #       ### index.test.js
```

```
### package.json
### samconfig.toml
### src
#   ### index.js
### template.yaml
```

Das Folgende ist ein Beispiel für unseren Lambda-Funktionscode:

```
exports.handler = awslambda.streamifyResponse(
   async (event, responseStream, context) => {
     const httpResponseMetadata = {
       statusCode: 200,
       headers: {
         "Content-Type": "text/html",
         "X-Custom-Header": "Example-Custom-Header"
       }
     };

     responseStream = awslambda.HttpResponseStream.from(responseStream,
 httpResponseMetadata);
     // It's recommended to use a `pipeline` over the `write` method for more complex
 use cases.
     // Learn more: https://docs.aws.amazon.com/lambda/latest/dg/configuration-
response-streaming.html
     responseStream.write("<html>");
     responseStream.write("<p>First write!</p>");

     responseStream.write("<h1>Streaming h1</h1>");
     await new Promise(r => setTimeout(r, 1000));
     responseStream.write("<h2>Streaming h2</h2>");
     await new Promise(r => setTimeout(r, 1000));
     responseStream.write("<h3>Streaming h3</h3>");
     await new Promise(r => setTimeout(r, 1000));

     // Long strings will be streamed
     const loremIpsum1 = "Lorem ipsum dolor sit amet, consectetur adipiscing elit.
 Quisque vitae mi tincidunt tellus ultricies dignissim id et diam. Morbi pharetra eu
 nisi et finibus. Vivamus diam nulla, vulputate et nisl cursus, pellentesque vehicula
 libero. Cras imperdiet lorem ante, non posuere dolor sollicitudin a. Vestibulum ipsum
 lacus, blandit nec augue id, lobortis dictum urna. Vestibulum ante ipsum primis in
 faucibus orci luctus et ultrices posuere cubilia curae; Morbi auctor orci eget tellus
 aliquam, non maximus massa porta. In diam ante, pulvinar aliquam nisl non, elementum
```

```
 hendrerit sapien. Vestibulum massa nunc, mattis non congue vitae, placerat in quam.
 Nam vulputate lectus metus, et dignissim erat varius a.";
     responseStream.write(`<p>${loremIpsum1}</p>`);
     await new Promise(r => setTimeout(r, 1000));

     responseStream.write("<p>DONE!</p>");
     responseStream.write("</html>");
     responseStream.end();
   }
 );
```

Das Folgende ist ein Beispiel für unsere `template.yaml` Datei. Das Antwort-Streaming für unsere Lambda-Funktion wird mithilfe der `FunctionUrlConfig` Eigenschaft konfiguriert.

```
AWSTemplateFormatVersion: '2010-09-09'
 Transform: AWS::Serverless-2016-10-31

 Description: >
   Sample SAM Template for lambda-streaming-nodejs-app

 Resources:
   StreamingFunction:
     Type: AWS::Serverless::Function
     Properties:
       CodeUri: src/
       Handler: index.handler
       Runtime: nodejs18.x
       Architectures:
         - x86_64
       Timeout: 10
       FunctionUrlConfig:
         AuthType: AWS_IAM
         InvokeMode: RESPONSE_STREAM

 Outputs:
   StreamingFunction:
     Description: "Streaming Lambda Function ARN"
     Value: !GetAtt StreamingFunction.Arn
   StreamingFunctionURL:
     Description: "Streaming Lambda Function URL"
     Value: !GetAtt StreamingFunctionUrl.FunctionUrl
```

In der Regel können Sie `sam build` und verwendensam `deploy --guided`, um eine Produktionsanwendung zu erstellen und bereitzustellen. In diesem Beispiel gehen wir von einer Entwicklungsumgebung aus und verwenden den `sam sync` Befehl, um unsere Anwendung zu erstellen und bereitzustellen.

> **Note**
>
> Der `sam sync` Befehl wird für Entwicklungsumgebungen empfohlen. Weitere Informationen hierzu finden Sie unter Einführung in die Verwendung von sam sync to sync to AWS Cloud.

Vor der Ausführung überprüfen wirsam `sync`, ob unser Projekt in unserer `samconfig.toml` Datei korrekt konfiguriert ist. Am wichtigsten ist, dass wir die Werte für `stack_name` und überprüfen`watch`. Da diese Werte in unserer Konfigurationsdatei angegeben sind, müssen wir sie nicht in der Befehlszeile angeben.

```
version = 0.1

 [default]
 [default.global.parameters]
 stack_name = "lambda-streaming-nodejs-app"

 [default.build.parameters]
 cached = true
 parallel = true

 [default.validate.parameters]
 lint = true

 [default.deploy.parameters]
 capabilities = "CAPABILITY_IAM"
 confirm_changeset = true
 resolve_s3 = true
 s3_prefix = "lambda-streaming-nodejs-app"
 region = "us-west-2"
 image_repositories = []

 [default.package.parameters]
 resolve_s3 = true

 [default.sync.parameters]
```

```
watch = true

[default.local_start_api.parameters]
warm_containers = "EAGER"

[default.local_start_lambda.parameters]
warm_containers = "EAGER"
```

Als Nächstes erstellen und implementieren wir unsere Anwendung. `sam sync` Da die `--watch` Option in unserer Konfigurationsdatei konfiguriert ist, AWS SAMCLI wird sie unsere Anwendung erstellen, unsere Anwendung bereitstellen und auf Änderungen achten.

```
$ sam sync

The SAM CLI will use the AWS Lambda, Amazon API Gateway, and AWS StepFunctions APIs to
upload your code
without

performing a CloudFormation deployment. This will cause drift in your CloudFormation
stack.
**The sync command should only be used against a development stack**.



Queued infra sync. Waiting for in progress code syncs to complete...

Starting infra sync.

Building codeuri:

/Users/.../lambda-streaming-nodejs-app/src runtime: nodejs18.x metadata: {}
architecture: x86_64 functions: StreamingFunction
package.json file not found. Continuing the build without dependencies.

Running NodejsNpmBuilder:CopySource


Build Succeeded

Successfully packaged artifacts and wrote output template to file /var/
folders/45/5ct135bx3fn2551_ptl5g6_80000gr/T/tmpavrzdhgp.
Execute the following command to deploy the packaged template
```

```
 sam deploy --template-file /var/folders/45/5ct135bx3fn2551_ptl5g6_80000gr/T/
tmpavrzdhgp --stack-name <YOUR STACK NAME>


        Deploying with following values
        ===============================
        Stack name                   : lambda-streaming-nodejs-app
        Region                       : us-west-2
        Disable rollback             : False
        Deployment s3 bucket         : aws-sam-cli-managed-default-samclisam-s3-demo-
bucket-1a4x26zbcdkqr
        Capabilities                 : ["CAPABILITY_NAMED_IAM",
 "CAPABILITY_AUTO_EXPAND"]
        Parameter overrides          : {}
        Signing Profiles             : null

 Initiating deployment
 =====================


 2023-06-20 12:11:16 - Waiting for stack create/update to complete

 CloudFormation events from stack operations (refresh every 0.5 seconds)
 ---------------------------------------------------------------------------------------------
 ResourceStatus            ResourceType              LogicalResourceId
 ResourceStatusReason
 ---------------------------------------------------------------------------------------------
 CREATE_IN_PROGRESS        AWS::CloudFormation::St   lambda-streaming-
 Transformation
                           ack                       nodejs-app
 succeeded
 CREATE_IN_PROGRESS        AWS::IAM::Role            StreamingFunctionRole     -

 CREATE_IN_PROGRESS        AWS::CloudFormation::St   AwsSamAutoDependencyLay   -

                           ack                       erNestedStack

 CREATE_IN_PROGRESS        AWS::IAM::Role            StreamingFunctionRole     Resource
 creation

 Initiated
 CREATE_IN_PROGRESS        AWS::CloudFormation::St   AwsSamAutoDependencyLay   Resource
 creation
```

```
                           ack                         erNestedStack
 Initiated
 CREATE_COMPLETE           AWS::IAM::Role              StreamingFunctionRole     -

 CREATE_COMPLETE           AWS::CloudFormation::St     AwsSamAutoDependencyLay   -

                           ack                         erNestedStack

 CREATE_IN_PROGRESS        AWS::Lambda::Function       StreamingFunction         -

 CREATE_IN_PROGRESS        AWS::Lambda::Function       StreamingFunction         Resource
 creation

 Initiated
 CREATE_COMPLETE           AWS::Lambda::Function       StreamingFunction         -

 CREATE_IN_PROGRESS        AWS::Lambda::Url            StreamingFunctionUrl      -

 CREATE_IN_PROGRESS        AWS::Lambda::Url            StreamingFunctionUrl      Resource
 creation

 Initiated
 CREATE_COMPLETE           AWS::Lambda::Url            StreamingFunctionUrl      -

 CREATE_COMPLETE           AWS::CloudFormation::St     lambda-streaming-         -

                           ack                         nodejs-app

 -----------------------------------------------------------------------------------------

 CloudFormation outputs from deployed stack
 -----------------------------------------------------------------------------------------
 Outputs

 -----------------------------------------------------------------------------------------
 Key                 StreamingFunction

 Description         Streaming Lambda Function ARN

 Value               arn:aws:lambda:us-west-2:012345678910:function:lambda-streaming-
nodejs-app-
 StreamingFunction-gUmhO833A0vZ
```

```
 Key                 StreamingFunctionURL

 Description         Streaming Lambda Function URL

 Value               https://wxgkcc2dyntgtrwhf2dgdcvylu0rnnof.lambda-url.us-
west-2.on.aws/
 -----------------------------------------------------------------------------------------------------------------



 Stack creation succeeded. Sync infra completed.



 Infra sync completed.
```

Jetzt, da unsere Funktion in der Cloud bereitgestellt ist, können wir `sam remote invoke` sie verwenden, um mit unserer Funktion zu interagieren. Die erkennt AWS SAMCLI automatisch, dass unsere Funktion für Antwort-Streaming konfiguriert ist, und beginnt sofort mit der Ausgabe einer gestreamten Antwort unserer Funktion in Echtzeit.

```
$ sam remote invoke StreamingFunction

 Invoking Lambda Function StreamingFunction

 {"statusCode":200,"headers":{"Content-Type":"text/html","X-Custom-Header":"Example-
Custom-Header"}}<html><p>First write!</p><h1>Streaming h1</h1><h2>Streaming h2</
h2><h3>Streaming h3</h3><p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
 Quisque vitae mi tincidunt tellus ultricies dignissim id et diam. Morbi pharetra eu
 nisi et finibus. Vivamus diam nulla, vulputate et nisl cursus, pellentesque vehicula
 libero. Cras imperdiet lorem ante, non posuere dolor sollicitudin a. Vestibulum ipsum
 lacus, blandit nec augue id, lobortis dictum urna. Vestibulum ante ipsum primis in
 faucibus orci luctus et ultrices posuere cubilia curae; Morbi auctor orci eget tellus
 aliquam, non maximus massa porta. In diam ante, pulvinar aliquam nisl non, elementum
 hendrerit sapien. Vestibulum massa nunc, mattis non congue vitae, placerat in quam.
 Nam vulputate lectus metus, et dignissim erat varius a.</p><p>DONE!</p></html>START
 RequestId: 1e4cdf04-60de-4769-b3a2-c1481982deb4 Version: $LATEST
 END RequestId: 1e4cdf04-60de-4769-b3a2-c1481982deb4
 REPORT RequestId: 1e4cdf04-60de-4769-b3a2-c1481982deb4  Duration: 4088.66 ms
 Billed Duration: 4089 ms        Memory Size: 128 MB     Max Memory Used: 68 MB  Init
 Duration: 168.45 ms
```

Wenn wir unseren Funktionscode ändern, erkennt der unsere Änderungen AWS SAMCLI sofort und implementiert sie sofort. Hier ist ein Beispiel für die AWS SAMCLI Ausgabe, nachdem Änderungen an unserem Funktionscode vorgenommen wurden:

```
Syncing Lambda Function StreamingFunction...

 Building codeuri:

 /Users/.../lambda-streaming-nodejs-app/src runtime: nodejs18.x metadata: {}
 architecture:
 x86_64 functions: StreamingFunction

 package.json file not found. Continuing the build without dependencies.

 Running NodejsNpmBuilder:CopySource

 Finished syncing Lambda Function StreamingFunction.

 Syncing Layer StreamingFunctione9cfe924DepLayer...

 SyncFlow [Layer StreamingFunctione9cfe924DepLayer]: Skipping resource update as the

 content didn't change

 Finished syncing Layer StreamingFunctione9cfe924DepLayer.
```

Wir können es jetzt `sam remote invoke` wieder verwenden, um mit unserer Funktion in der Cloud zu interagieren und unsere Änderungen zu testen.

SQS-Beispiele

Grundlegende Beispiele

Rufen Sie eine Amazon SQS SQS-Warteschlange auf, indem Sie den ARN als Ressourcen-ID angeben:

```
$ sam remote invoke arn:aws:sqs:us-west-2:01234567890:sqs-example-4DonhBsjsW1b --
event '{"hello": "world"}' --output json

Sending message to SQS queue MySqsQueue

{
```

```
  "MD5OfMessageBody": "49dfdd54b01cbcd2d2ab5e9e5ee6b9b9",
  "MessageId": "4f464cdd-15ef-4b57-bd72-3ad225d80adc",
  "ResponseMetadata": {
    "RequestId": "95d39377-8323-5ef0-9223-ceb198bd09bd",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "x-amzn-requestid": "95d39377-8323-5ef0-9223-ceb198bd09bd",
      "date": "Wed, 08 Nov 2023 23:27:26 GMT",
      "content-type": "application/x-amz-json-1.0",
      "content-length": "106",
      "connection": "keep-alive"
    },
    "RetryAttempts": 0
  }
}%
```

Beispiele Step Functions

Grundlegende Beispiele

Rufen Sie eine Zustandsmaschine auf, indem Sie ihre physische ID als Ressourcen-ID angeben:

Zuerst verwenden wir, `sam list resources` um unsere physische ID zu erhalten:

```
$ sam list resources --stack-name state-machine-example --output json

[
  {
    "LogicalResourceId": "HelloWorldStateMachine",
    "PhysicalResourceId": "arn:aws:states:us-
west-2:513423067560:stateMachine:HelloWorldStateMachine-z69tFEUx0F66"
  },
  {
    "LogicalResourceId": "HelloWorldStateMachineRole",
    "PhysicalResourceId": "simple-state-machine-HelloWorldStateMachineRole-
PduA0BDGuFXw"
  }
]
```

Als Nächstes rufen wir unsere Zustandsmaschine auf, wobei wir die physische ID als Ressourcen-ID verwenden. Wir übergeben ein Ereignis an der Befehlszeile mit der `--event` Option:

```
$ sam remote invoke arn:aws:states:us-
west-2:01234567890:stateMachine:HelloWorldStateMachine-z69tFEUx0F66 --
event '{"is_developer": true}'

Invoking Step Function arn:aws:states:us-
west-2:01234567890:stateMachine:HelloWorldStateMachine-z69tFEUx0F66
"Hello Developer World"%
```

Rufen Sie eine Zustandsmaschine auf, indem Sie ein leeres Ereignis übergeben:

```
$ sam remote invoke HelloWorldStateMachine --stack-name state-machine-example

Invoking Step Function HelloWorldStateMachine

"Hello World"%
```

Weiterführende Links

Dokumentation zu `sam remote invoke` und zur Verwendung von finden Sie im Folgenden: AWS SAMCLI

- sam remote invoke
- AWS SAMCLIProblembehandlung

Automatisieren Sie lokale Integrationstests mit AWS SAM

Sie können Einführung in das Testen mit sam local invoke damit zwar Code manuell testen, AWS SAM aber Sie können Ihren Code auch mithilfe automatisierter Integrationstests testen. Integrationstests helfen Ihnen dabei, Probleme früh im Entwicklungszyklus zu erkennen, die Qualität Ihres Codes zu verbessern, Zeit zu sparen und gleichzeitig Kosten zu senken.

Um automatisierte Integrationstests zu erstellen AWS SAM, führen Sie zunächst Tests mit lokalen Lambda-Funktionen durch, bevor Sie sie in der AWS Cloud bereitstellen. Der Einführung in das Testen mit sam local start-lambda Befehl startet einen lokalen Endpunkt, der den Lambda-Aufrufendpunkt emuliert. Sie können ihn von Ihren automatisierten Tests aus aufrufen. Da dieser Endpunkt den Lambda-Aufrufendpunkt emuliert, können Sie Tests einmal schreiben und sie dann (ohne Änderungen) für die lokale Lambda-Funktion oder für eine bereitgestellte Lambda-Funktion

ausführen. Sie können dieselben Tests auch für einen bereitgestellten AWS SAM Stack in Ihrer Pipeline ausführen. CI/CD

So funktioniert der Prozess:

1. Starten Sie den lokalen Lambda-Endpunkt.

   Starten Sie den lokalen Lambda-Endpunkt, indem Sie den folgenden Befehl in dem Verzeichnis ausführen, das Ihre AWS SAM Vorlage enthält:

   ```
   sam local start-lambda
   ```

   Mit diesem Befehl wird ein lokaler Endpunkt gestartet`http://127.0.0.1:3001`, der emuliert AWS Lambda. Sie können Ihre automatisierten Tests für diesen lokalen Lambda-Endpunkt ausführen. Wenn Sie diesen Endpunkt mit dem AWS CLI oder SDK aufrufen, führt er lokal die in der Anfrage angegebene Lambda-Funktion aus und gibt eine Antwort zurück.
2. Führen Sie einen Integrationstest für den lokalen Lambda-Endpunkt durch.

   In Ihrem Integrationstest können Sie das AWS SDK verwenden, um Ihre Lambda-Funktion mit Testdaten aufzurufen, auf eine Antwort zu warten und zu überprüfen, ob die Antwort Ihren Erwartungen entspricht. Um den Integrationstest lokal auszuführen, sollten Sie das AWS SDK so konfigurieren, dass es einen Lambda Invoke-API-Aufruf sendet, um den lokalen Lambda-Endpunkt aufzurufen, den Sie im vorherigen Schritt gestartet haben.

   Das Folgende ist ein Python-Beispiel (die AWS SDKs vier anderen Sprachen haben ähnliche Konfigurationen):

   ```
   import boto3
   import botocore

   # Set "running_locally" flag if you are running the integration test locally
   running_locally = True

   if running_locally:

       # Create Lambda SDK client to connect to appropriate Lambda endpoint
       lambda_client = boto3.client('lambda',
           region_name="us-west-2",
           endpoint_url="http://127.0.0.1:3001",
           use_ssl=False,
   ```

```
        verify=False,
        config=botocore.client.Config(
            signature_version=botocore.UNSIGNED,
            read_timeout=15,
            retries={'max_attempts': 0},
        )
    )
else:
    lambda_client = boto3.client('lambda')


# Invoke your Lambda function as you normally usually do. The function will run
# locally if it is configured to do so
response = lambda_client.invoke(FunctionName="HelloWorldFunction")

# Verify the response
assert response == "Hello World"
```

Sie können diesen Code verwenden, um bereitgestellte Lambda-Funktionen zu testen, indem Sie `running_locally` auf `False` setzen. Dadurch wird das AWS SDK für die Verbindung AWS Lambda in der AWS Cloud eingerichtet.

Generieren Sie Beispielereignis-Payloads mit AWS SAM

Um Ihre Lambda-Funktionen zu testen, können Sie Beispielereignis-Payloads generieren und anpassen, die die Daten imitieren, die Ihre Lambda-Funktionen erhalten, wenn sie von anderen Diensten ausgelöst werden. AWS Dazu gehören Dienste wie API Gateway CloudFormation, Amazon S3 und mehr.

Durch das Generieren von Nutzlasten für Beispielereignisse können Sie das Verhalten Ihrer Lambda-Funktion mit einer Vielzahl verschiedener Eingaben testen, ohne in einer Live-Umgebung arbeiten zu müssen. Dieser Ansatz spart auch Zeit im Vergleich zur manuellen Erstellung von Beispielen für AWS Serviceereignisse zum Testen von Funktionen.

Eine vollständige Liste der Dienste, für die Sie Beispielereignis-Payloads generieren können, finden Sie mit diesem Befehl:

```
sam local generate-event --help
```

Eine Liste der Optionen, die Sie für einen bestimmten Dienst verwenden können, finden Sie mit diesem Befehl:

```
sam local generate-event [SERVICE] --help
```

Beispiele:

```
#Generates the event from S3 when a new object is created
sam local generate-event s3 put

# Generates the event from S3 when an object is deleted
sam local generate-event s3 delete
```

Testen und Debuggen langlebiger Funktionen

Das lokale Testen und Debuggen langlebiger Funktionen funktioniert ähnlich wie reguläre Lambda-Funktionen, mit automatischer Unterstützung für Checkpointing und Replay. Dieses Handbuch behandelt gängige Testszenarien und Techniken zur Fehlerbehebung.

Lokaler Test-Workflow

Wenn langlebige Funktionen lokal getestet werden, unterscheidet sich der Workflow von regulären Lambda-Funktionen:

Arbeitsablauf beim Testen dauerhafter Funktionen

1. Erstellen Sie Ihre Anwendung:

   ```
   $ sam build
   ```

2. Rufen Sie Ihre dauerhafte Funktion auf:

   ```
   $ sam local invoke MyDurableFunction --durable-execution-name test
   ```

3. Überprüfen Sie bei Bedarf den Ausführungsverlauf:

   ```
   $ sam local execution history execution-id
   ```

Allgemeine Testszenarien

Testen des Checkpointing-Verhaltens

Um zu testen, ob Ihre Funktion ordnungsgemäß funktioniert, gehen Sie wie folgt vor:

```
# Example Python durable function
def handler(event, context):
    # This will create a checkpoint
    context.wait(10)  # Wait 10 seconds

    # Function resumes here after wait
    return {"message": "Function resumed after wait"}
```

Wenn Sie diese Funktion lokal aufrufen, wird die Wartezeit automatisch behandelt.

Callback-Szenarien testen

Für Funktionen, die auf externe Rückrufe warten:

1. Starten Sie Ihre dauerhafte Funktion, die auf einen Rückruf wartet
2. Löse den Rückruf in einem anderen Terminal auf:

   ```
   $ sam local callback succeed callback-id
   ```

3. Beobachten Sie, wie die Funktion die Ausführung fortsetzt

Fehlerbehebung

Dauerhafte Funktion wird nicht ordnungsgemäß ausgeführt

Problem: Die Funktion verhält sich nicht wie eine dauerhafte Funktion.

Lösungen:

- Vergewissern Sie `DurableConfig` sich, dass dies in Ihrer SAM-Vorlage festgelegt ist
- Stellen Sie sicher, dass Ihr Funktionscode SDK-Methoden für dauerhafte Funktionen verwendet `context.wait()` (z. B.
- Überprüfe, ob du eine unterstützte Runtime (TypeScript, JavaScript, Python) verwendest

Der Ausführungsverlauf kann nicht abgerufen werden

Problem: Der `local execution history` Befehl gibt keine Ergebnisse zurück.

Lösungen:

- Stellen Sie sicher, dass die Ausführungs-ID korrekt ist
- Vergewissern Sie sich, dass die Funktion mindestens einmal aufgerufen wurde

Callback-Befehle funktionieren nicht

Problem: Callback-Befehle lösen ausstehende Operationen nicht auf.

Lösungen:

- Stellen Sie sicher, dass die Rückruf-ID korrekt ist
- Stellen Sie sicher, dass die Funktion tatsächlich auf einen Rückruf wartet
- Stellen Sie sicher, dass Sie die richtige Callback-Befehlssyntax verwenden

Tipps zum Debuggen

- Ausführungshistorie verwenden — Sehen Sie sich den Ausführungsverlauf an, um den Ablauf Ihrer dauerhaften Funktion nachzuvollziehen
- Testen Sie inkrementell — Beginnen Sie mit einfachen Warteoperationen, bevor Sie komplexe Logik hinzufügen
- Ausführliche Protokollierung verwenden — Aktivieren Sie die detaillierte Protokollierung, um den Ausführungsablauf nachzuverfolgen

Weitere Informationen

Weitere Informationen zum Testen und Debuggen finden Sie unter:

- Einführung in das Testen mit sam local invoke- Dokumentation zum lokalen Aufrufen
- sam local execution history- Ausführungshistorie

Debuggen Sie Ihre serverlose Anwendung mit AWS SAM

Nach dem Testen Ihrer Anwendung sind Sie bereit, alle gefundenen Probleme zu debuggen. Mit der AWS SAM Befehlszeilenschnittstelle (CLI) können Sie Ihre serverlose Anwendung lokal testen und debuggen, bevor Sie sie in die Cloud hochladen. AWS Durch das Debuggen Ihrer Anwendung werden Probleme oder Fehler in Ihrer Anwendung identifiziert und behoben.

Sie können AWS SAM es verwenden, um ein schrittweises Debugging durchzuführen. Dabei handelt es sich um eine Methode, bei der Code zeilenweise oder Anweisung nach der anderen ausgeführt wird. Wenn Sie lokal eine Lambda-Funktion im Debug-Modus innerhalb von aufrufen AWS SAMCLI, können Sie ihr anschließend einen Debugger anhängen. Mit dem Debugger können Sie Ihren Code Zeile für Zeile durchgehen, die Werte verschiedener Variablen einsehen und Probleme beheben, genauso wie Sie es mit jeder anderen Anwendung tun würden. Sie können überprüfen, ob sich Ihre Anwendung wie erwartet verhält, Fehler debuggen und alle Probleme beheben, bevor Sie die Schritte zum Verpacken und Bereitstellen Ihrer Anwendung ausführen.

> **Note**
>
> Wenn Ihre Anwendung eine oder mehrere Ebenen umfasst, wird das Layer-Paket heruntergeladen und auf Ihrem lokalen Host zwischengespeichert, wenn Sie Ihre Anwendung lokal ausführen und debuggen. Weitere Informationen finden Sie unter Wie werden Ebenen lokal zwischengespeichert.

Themen

- Lokales Debuggen von Funktionen mit AWS SAM
- Übergeben Sie beim Debuggen mehrere Laufzeitargumente mit AWS SAM
- Validieren Sie Ihre AWS SAM Bewerbungen mit CloudFormation Linter

Lokales Debuggen von Funktionen mit AWS SAM

Sie können eine Vielzahl von AWS Toolkits und Debuggern verwenden AWS SAM , um Ihre serverlosen Anwendungen lokal zu testen und zu debuggen. Durch das schrittweise Debuggen Ihrer Lambda-Funktionen können Sie Probleme in Ihrer Anwendung Zeile oder Anweisung für Zeile in Ihrer lokalen Umgebung identifizieren und beheben.

Zu den Möglichkeiten, wie Sie lokales schrittweises Debugging durchführen können, gehören das Setzen von Breakpoints, das Überprüfen von Variablen und das zeilenweise Ausführen von Funktionscode. Durch lokales schrittweises Debuggen wird die Feedback-Schleife enger, indem es Ihnen ermöglicht, Probleme zu finden und zu beheben, auf die Sie in der Cloud stoßen könnten.

Sie können AWS Toolkits zum Debuggen verwenden, und Sie können es auch im Debug-Modus ausführen. AWS SAM Einzelheiten finden Sie in den Themen in diesem Abschnitt.

Verwenden von AWS Toolkits

AWS Toolkits sind IDE-Plugins (Integrated Development Environment), mit denen Sie viele gängige Debugging-Aufgaben ausführen können, wie z. B. das Setzen von Breakpoints, das Prüfen von Variablen und das zeilenweise Ausführen von Funktionscode. AWS Toolkits erleichtern Ihnen das Entwickeln, Debuggen und Bereitstellen serverloser Anwendungen, die mit erstellt wurden. AWS SAM Sie bieten eine Erfahrung zum Erstellen, Testen, Debuggen, Bereitstellen und Aufrufen von Lambda-Funktionen, die in Ihre IDE integriert sind.

Weitere Informationen zu AWS Toolkits, die Sie zusammen verwenden können, finden Sie im Folgenden AWS SAM:

- AWS Toolkit for Visual Studio Code
- AWS Cloud9
- AWS Toolkit for JetBrains

Es gibt eine Vielzahl von AWS Toolkits, die mit unterschiedlichen Kombinationen von IDEs und Laufzeiten funktionieren. In der folgenden Tabelle sind gängige IDE/runtime Kombinationen aufgeführt, die das schrittweise Debuggen von Anwendungen unterstützen: AWS SAM

IDE	Laufzeit	AWS Toolkit	Anweisungen für das schrittweise Debuggen
Visual Studio Code	• Node.js • Python • .NET • Java • Go	AWS Toolkit for Visual Studio Code	Arbeiten mit AWS-Serverless-Anwendung im Benutzerhandbuch AWS Toolkit for Visual Studio Code

IDE	Laufzeit	AWS Toolkit	Anweisungen für das schrittweise Debuggen
AWS Cloud9	• Node.js • Python	AWS Cloud9, mit aktiviertem AWS Toolkit 1	Arbeiten mit AWS serverlosen Anwendungen mithilfe des AWS Toolkits im AWS Cloud9 Benutzerhandbuch.
WebStorm	Node.js	AWS Toolkit for JetBrains[2]	Ausführen (Aufrufen) oder Debuggen einer lokalen Funktion in der AWS Toolkit for JetBrains
PyCharm	Python	AWS Toolkit for JetBrains[2]	Ausführen (Aufrufen) oder Debuggen einer lokalen Funktion in der AWS Toolkit for JetBrains
Rider	.NET	AWS Toolkit for JetBrains[2]	Ausführen (Aufrufen) oder Debuggen einer lokalen Funktion in der AWS Toolkit for JetBrains
IntelliJ	Java	AWS Toolkit for JetBrains[2]	Ausführen (Aufrufen) oder Debuggen einer lokalen Funktion in der AWS Toolkit for JetBrains

IDE	Laufzeit	AWS Toolkit	Anweisungen für das schrittweise Debuggen
GoLand	Go	AWS Toolkit for JetBrains[2]	Ausführen (Aufrufen) oder Debuggen einer lokalen Funktion in der AWS Toolkit for JetBrains

Hinweise:

1. Um das schrittweise Debuggen von AWS SAM Anwendungen AWS Cloud9 zu verwenden, muss das AWS Toolkit aktiviert sein. Weitere Informationen finden Sie im Benutzerhandbuch unter Aktivieren des AWS Toolkits.AWS Cloud9
2. AWS Toolkit for JetBrains Um die schrittweisen AWS SAM Debug-Anwendungen verwenden zu können, müssen Sie sie zunächst installieren und konfigurieren, indem Sie den Anweisungen unter Installation von folgen. AWS Toolkit for JetBrainsAWS Toolkit for JetBrains

Wird AWS SAM lokal im Debug-Modus ausgeführt

Zusätzlich zur Integration mit AWS Toolkits können Sie die Software auch AWS SAM im Debug-Modus ausführen, um Verbindungen zu Debuggern von Drittanbietern wie ptvsd oder delve herzustellen.

Um AWS SAM im Debug-Modus zu starten, verwenden Sie Befehle oder die Option oder. sam local invoke Sam lokale Start-API `--debug-port -d`

Beispiel:

```
# Invoke a function locally in debug mode on port 5858
sam local invoke -d 5858 <function logical id>

# Start local API Gateway in debug mode on port 5858
sam local start-api -d 5858
```

> **Note**
>
> Wenn Sie verwendensam `local start-api`, macht die lokale API-Gateway-Instanz alle Ihre Lambda-Funktionen verfügbar. Da Sie jedoch einen einzelnen Debug-Port angeben können, können Sie jeweils nur eine Funktion debuggen. Sie müssen Ihre API aufrufen, bevor sie sich an den Port AWS SAMCLI bindet, sodass der Debugger eine Verbindung herstellen kann.

Übergeben Sie beim Debuggen mehrere Laufzeitargumente mit AWS SAM

Sie können sich dafür entscheiden, zusätzliche Laufzeitargumente mit AWS SAM zu übergeben, um Probleme zu untersuchen und Variablen effektiver zu behandeln. Auf diese Weise erhalten Sie mehr Kontrolle und Flexibilität für Ihren Debugging-Prozess, was Ihnen bei benutzerdefinierten Laufzeitkonfigurationen und -umgebungen helfen kann.

Verwenden Sie die Umgebungsvariable, um beim Debuggen Ihrer Funktion zusätzliche Laufzeitargumente zu übergeben. `DEBUGGER_ARGS` Dadurch wird eine Reihe von Argumenten direkt an den Run-Befehl übergeben, mit dem AWS SAMCLI Ihre Funktion gestartet wird.

Wenn Sie beispielsweise einen Debugger wie i KPdb zur Laufzeit Ihrer Python-Funktion laden möchten, können Sie Folgendes als `DEBUGGER_ARGS: -m ikpdb --ikpdb-port=5858 --ikpdb-working-directory=/var/task/ --ikpdb-client-working-directory=/myApp --ikpdb-address=0.0.0.0` übergeben. Dies würde i zur KPdb Laufzeit mit den anderen Argumenten laden, die Sie angegeben haben.

In diesem Fall wäre Ihr vollständiger AWS SAMCLI Befehl:

```
DEBUGGER_ARGS="-m ikpdb --ikpdb-port=5858 --ikpdb-working-directory=/var/task/ --ikpdb-client-working-directory=/myApp --ikpdb-address=0.0.0.0" echo {} | sam local invoke -d 5858 myFunction
```

Sie können Debugger-Argumente an die Funktionen aller Laufzeiten übergeben.

Validieren Sie Ihre AWS SAM Bewerbungen mit CloudFormation Linter

AWS CloudFormation Linter (cfn-lint) ist ein Open-Source-Tool, mit dem Sie Ihre Vorlagen detailliert validieren können. CloudFormation CFN-Lint enthält Regeln, die sich an der Ressourcenspezifikation orientieren. CloudFormation Verwenden Sie cfn-lint, um Ihre Ressourcen mit diesen Regeln zu vergleichen und detaillierte Meldungen zu Fehlern, Warnungen oder Informationsvorschlägen zu erhalten. Alternativ können Sie Ihre eigenen benutzerdefinierten Regeln erstellen, anhand derer validiert werden soll. Weitere Informationen zu cfn-lint finden Sie im Repository unter cfn-lint.CloudFormation GitHub

Sie können cfn-lint verwenden, um Ihre AWS Serverless Application Model (AWS SAM) -Vorlagen über die AWS SAM Befehlszeilenschnittstelle (AWS SAMCLI) zu validieren, indem Sie sie mit der Option ausführen. sam validate --lint

```
sam validate --lint
```

Um das Verhalten von cfn-lint anzupassen, z. B. um benutzerdefinierte Regeln zu erstellen oder Validierungsoptionen anzugeben, können Sie eine Konfigurationsdatei definieren. Weitere Informationen finden Sie unter Config File im cfn-lint-Repository CloudFormation GitHub . Bei der Ausführung wird sam validate --lint das in Ihrer Konfigurationsdatei definierte Verhalten von cfn-lint angewendet.

Beispiele

Führen Sie eine CFN-Lint-Validierung für eine Vorlage durch AWS SAM

```
sam validate --lint --template myTemplate.yaml
```

Weitere Informationen

Weitere Informationen zum sam validate-Befehl finden Sie unter sam validate.

Stellen Sie Ihre Anwendung und Ressourcen bereit mit AWS SAM

Durch die Bereitstellung Ihrer Anwendung werden Ihre AWS Ressourcen in der AWS Cloud bereitgestellt und konfiguriert, sodass Ihre Anwendung in der Cloud ausgeführt wird. AWS SAM verwendet CloudFormationals zugrunde liegenden Bereitstellungsmechanismus. AWS SAM verwendet die Build-Artefakte, die Sie bei der Ausführung des sam build Befehls erstellen, als Standardeingaben für die Bereitstellung Ihrer serverlosen Anwendung.

Mit AWS SAM können Sie Ihre serverlose Anwendung manuell bereitstellen oder Bereitstellungen automatisieren. Um Bereitstellungen zu automatisieren, verwenden Sie AWS SAM Pipelines mit einem CI/CD-System (Continuous Integration and Continuous Deployment) Ihrer Wahl. Ihre Bereitstellungspipeline besteht aus einer automatisierten Abfolge von Schritten, die ausgeführt werden, um eine neue Version Ihrer serverlosen Anwendung zu veröffentlichen.

Die Themen in diesem Abschnitt enthalten Anleitungen sowohl zu automatisierten als auch zu manuellen Bereitstellungen. Um Ihre Anwendung manuell bereitzustellen, verwenden Sie AWS SAMCLI Befehle. Informationen zur Automatisierung von Bereitstellungen finden Sie in den Themen in diesem Abschnitt. Sie bieten insbesondere ausführliche Inhalte zur Automatisierung von Bereitstellungen mithilfe von Pipelines und einem System. CI/CD Dazu gehören die Generierung einer Starter-Pipeline, die Einrichtung der Automatisierung, die Fehlerbehebung bei Bereitstellungen, die Verwendung der OpenID Connect (OIDC) -Benutzerauthentifizierung und das Hochladen lokaler Dateien bei der Bereitstellung.

Themen

- Einführung in die Bereitstellung mit AWS SAM
- Optionen für die Bereitstellung Ihrer Anwendung mit AWS SAM
- Verwendung von CI/CD Systemen und Pipelines für die Bereitstellung mit AWS SAM
- Einführung in die Verwendung von sam sync to sync to AWS Cloud

Einführung in die Bereitstellung mit AWS SAM

Verwenden Sie den AWS Serverless Application Model Befehl Command Line Interface (AWS SAMCLI)`sam deploy`, um Ihre serverlose Anwendung auf dem bereitzustellen. AWS Cloud

- Eine Einführung in das finden Sie AWS SAMCLI unterWas ist der AWS SAM CLI?.

- Eine Liste der `sam deploy` Befehlsoptionen finden Sie untersam deploy.
- Ein Beispiel für die Verwendung `sam deploy` während eines typischen Entwicklungsworkflows finden Sie unterSchritt 3: Stellen Sie Ihre Anwendung auf dem bereit AWS Cloud.

Themen

- Voraussetzungen
- Bereitstellen von Anwendungen mit Sam Deploy
- Best Practices
- Optionen für Sam Deploy
- Fehlerbehebung
- Beispiele
- Weitere Informationen

Voraussetzungen

Um das zu verwenden`sam deploy`, installieren Sie das, AWS SAMCLI indem Sie wie folgt vorgehen:

- AWS SAM Voraussetzungen.
- Installieren Sie das AWS SAMCLI.

Vor der Verwendung empfehlen wir`sam deploy`, sich mit folgenden Grundkenntnissen vertraut zu machen:

- Konfiguration der AWS SAMCLI.
- Erstellen Sie Ihre Bewerbung in AWS SAM.
- Einführung in das Bauen mit AWS SAM.

Bereitstellen von Anwendungen mit Sam Deploy

Wenn Sie eine serverlose Anwendung zum ersten Mal bereitstellen, verwenden Sie die `--guided` Option. Sie AWS SAMCLI führt Sie durch einen interaktiven Ablauf zur Konfiguration der Bereitstellungseinstellungen Ihrer Anwendung.

Um eine Anwendung mithilfe des interaktiven Ablaufs bereitzustellen

1. Gehen Sie zum Stammverzeichnis Ihres Projekts. Dies ist derselbe Speicherort wie Ihre AWS SAM Vorlage.

   ```
   $ cd sam-app
   ```

2. Führen Sie den folgenden Befehl aus:

   ```
   $ sam deploy --guided
   ```

3. Während des interaktiven Ablaufs werden Sie AWS SAMCLI aufgefordert, Optionen zur Konfiguration der Bereitstellungseinstellungen Ihrer Anwendung anzugeben.

   Klammern (`[ ]`) stehen für Standardwerte. Lassen Sie Ihre Antwort leer, um den Standardwert auszuwählen. Standardwerte werden aus den folgenden Konfigurationsdateien abgerufen:

   - `~/.aws/config`— Ihre allgemeinen AWS Kontoeinstellungen.
   - `~/.aws/credentials`— Ihre AWS Kontoanmeldeinformationen.
   - *`<project>`*`/samconfig.toml`— Die Konfigurationsdatei Ihres Projekts.

   Geben Sie Werte an, indem Sie die AWS SAMCLI Eingabeaufforderungen beantworten. Sie können beispielsweise Werte **y** für Ja, **n** für Nein oder Zeichenfolgen eingeben.

   Der AWS SAMCLI schreibt Ihre Antworten in die `samconfig.toml` Datei Ihres Projekts. Für nachfolgende Bereitstellungen können Sie die Bereitstellung mithilfe dieser konfigurierten Werte verwenden. `sam deploy` Um diese Werte neu zu konfigurieren, verwenden Sie Ihre `sam deploy --guided` Konfigurationsdateien erneut oder ändern Sie sie direkt.

   Im Folgenden finden Sie ein Beispiel für eine Ausgabe:

   ```
   sam-app $ sam deploy --guided

   Configuring SAM deploy
   ======================

           Looking for config file [samconfig.toml] :  Found
           Reading default arguments  :  Success

           Setting default arguments for 'sam deploy'
   ```

```
        =========================================
        Stack Name [sam-app]: ENTER
        AWS Region [us-west-2]: ENTER
        #Shows you resources changes to be deployed and require a 'Y' to initiate
 deploy
        Confirm changes before deploy [Y/n]: ENTER
        #SAM needs permission to be able to create roles to connect to the
 resources in your template
        Allow SAM CLI IAM role creation [Y/n]: ENTER
        #Preserves the state of previously provisioned resources when an operation
 fails
        Disable rollback [y/N]: ENTER
        HelloWorldFunction may not have authorization defined, Is this okay? [y/
N]: y
        Save arguments to configuration file [Y/n]: ENTER
        SAM configuration file [samconfig.toml]: ENTER
        SAM configuration environment [default]: ENTER
```

4. Als Nächstes AWS SAMCLI stellt der Ihre Anwendung auf dem AWS Cloud bereit. Während der Bereitstellung wird der Fortschritt in Ihrer Befehlszeile angezeigt. Im Folgenden sind die wichtigsten Phasen der Bereitstellung aufgeführt:

   - Bei Anwendungen mit AWS Lambda Funktionen, die als ZIP-Dateiarchiv verpackt sind, wird das Paket AWS SAMCLI komprimiert und in einen Amazon Simple Storage Service (Amazon S3) -Bucket hochgeladen. Falls erforderlich, AWS SAMCLI wird ein neuer Bucket erstellt.
   - Bei Anwendungen mit einem Lambda-Funktionspaket als Container-Image wird das Image in Amazon Elastic Container Registry (Amazon ECR) AWS SAMCLI hochgeladen. Falls erforderlich, AWS SAMCLI wird ein neues Repository erstellt.
   - Das AWS SAMCLI erstellt einen AWS CloudFormation Änderungssatz und stellt Ihre Anwendung CloudFormation als Stack bereit.
   - Das AWS SAMCLI ändert Ihre bereitgestellte AWS SAM Vorlage mit dem neuen `CodeUri` Wert für Ihre Lambda-Funktionen.

   Im Folgenden finden Sie ein Beispiel für die AWS SAMCLI Bereitstellungsausgabe:

```
        Looking for resources needed for deployment:

        Managed S3 bucket: aws-sam-cli-managed-default-samclisam-s3-demo-
bucket-1a4x26zbcdkqr
```

```
        A different default S3 bucket can be set in samconfig.toml and auto
 resolution of buckets turned off by setting resolve_s3=False

        Parameter "stack_name=sam-app" in [default.deploy.parameters] is defined as
 a global parameter [default.global.parameters].
        This parameter will be only saved under [default.global.parameters] in /
Users/.../sam-app/samconfig.toml.

        Saved arguments to config file
        Running 'sam deploy' for future deployments will use the parameters saved
 above.
        The above parameters can be changed by modifying samconfig.toml
        Learn more about samconfig.toml syntax at
        https://docs.aws.amazon.com/serverless-application-model/latest/
developerguide/serverless-sam-cli-config.html

        Uploading to sam-app-zip/da3c598813f1c2151579b73ad788cac8  262144 / 619839
 (42.29%)Uploading to sam-app-zip/da3c598813f1c2151579b73ad788cac8  524288 / 619839
  (84.58%)Uploading to sam-app-zip/da3c598813f1c2151579b73ad788cac8  619839 /
 619839  (100.00%)

        Deploying with following values
        ===============================
        Stack name                   : sam-app
        Region                       : us-west-2
        Confirm changeset            : True
        Disable rollback             : False
        Deployment s3 bucket         : aws-sam-cli-managed-default-samclisam-s3-
demo-bucket-1a4x26zbcdkqr
        Capabilities                 : ["CAPABILITY_IAM"]
        Parameter overrides          : {}
        Signing Profiles             : {}

Initiating deployment
=====================

        Uploading to sam-app-zip/be84c20f868068e4dc4a2c11966edf2d.template  1212 /
 1212  (100.00%)


Waiting for changeset to be created..

CloudFormation stack changeset
-----------------------------------------------------------------------------------------
```

```
Operation                 LogicalResourceId         ResourceType
 Replacement
-------------------------------------------------------------------------------------
+ Add                     HelloWorldFunctionHell    AWS::Lambda::Permissio    N/A

                          oWorldPermissionProd      n

+ Add                     HelloWorldFunctionRole    AWS::IAM::Role            N/A

+ Add                     HelloWorldFunction        AWS::Lambda::Function     N/A

+ Add                     ServerlessRestApiDeplo    AWS::ApiGateway::Deplo    N/A

                          yment47fc2d5f9d           yment

+ Add                     ServerlessRestApiProdS    AWS::ApiGateway::Stage    N/A

                          tage

+ Add                     ServerlessRestApi         AWS::ApiGateway::RestA    N/A

                                                    pi

-------------------------------------------------------------------------------------


Changeset created successfully. arn:aws:cloudformation:us-
west-2:012345678910:changeSet/samcli-deploy1680559234/d9f58a77-98bc-41cd-
b9f4-433a5a450d7a


Previewing CloudFormation changeset before deployment
======================================================
Deploy this changeset? [y/N]: y

2023-04-03 12:00:50 - Waiting for stack create/update to complete

CloudFormation events from stack operations (refresh every 5.0 seconds)
-------------------------------------------------------------------------------------
ResourceStatus            ResourceType              LogicalResourceId
 ResourceStatusReason
-------------------------------------------------------------------------------------
CREATE_IN_PROGRESS        AWS::IAM::Role            HelloWorldFunctionRole    -
```

```
CREATE_IN_PROGRESS       AWS::IAM::Role           HelloWorldFunctionRole   Resource
 creation

 Initiated
CREATE_COMPLETE          AWS::IAM::Role           HelloWorldFunctionRole   -

CREATE_IN_PROGRESS       AWS::Lambda::Function    HelloWorldFunction       -

CREATE_IN_PROGRESS       AWS::Lambda::Function    HelloWorldFunction       Resource
 creation

 Initiated
CREATE_COMPLETE          AWS::Lambda::Function    HelloWorldFunction       -

CREATE_IN_PROGRESS       AWS::ApiGateway::RestA   ServerlessRestApi        -

                         pi

CREATE_IN_PROGRESS       AWS::ApiGateway::RestA   ServerlessRestApi        Resource
 creation
                         pi
 Initiated
CREATE_COMPLETE          AWS::ApiGateway::RestA   ServerlessRestApi        -

                         pi

CREATE_IN_PROGRESS       AWS::Lambda::Permissio   HelloWorldFunctionHell   -

                         n                        oWorldPermissionProd

CREATE_IN_PROGRESS       AWS::ApiGateway::Deplo   ServerlessRestApiDeplo   -

                         yment                    yment47fc2d5f9d

CREATE_IN_PROGRESS       AWS::Lambda::Permissio   HelloWorldFunctionHell   Resource
 creation
                         n                        oWorldPermissionProd
 Initiated
CREATE_IN_PROGRESS       AWS::ApiGateway::Deplo   ServerlessRestApiDeplo   Resource
 creation
                         yment                    yment47fc2d5f9d
 Initiated
CREATE_COMPLETE          AWS::ApiGateway::Deplo   ServerlessRestApiDeplo   -
```

```
                         yment                     yment47fc2d5f9d

CREATE_IN_PROGRESS       AWS::ApiGateway::Stage    ServerlessRestApiProdS   -

                                                   tage

CREATE_IN_PROGRESS       AWS::ApiGateway::Stage    ServerlessRestApiProdS   Resource
 creation
                                                   tage
 Initiated
CREATE_COMPLETE          AWS::ApiGateway::Stage    ServerlessRestApiProdS   -

                                                   tage

CREATE_COMPLETE          AWS::Lambda::Permissio    HelloWorldFunctionHell   -

                         n                         oWorldPermissionProd

CREATE_COMPLETE          AWS::CloudFormation::S    sam-app-zip              -

                         tack

-------------------------------------------------------------------------------------------

CloudFormation outputs from deployed stack
-------------------------------------------------------------------------------------------
Outputs

-------------------------------------------------------------------------------------------
Key                 HelloWorldFunctionIamRole

Description         Implicit IAM Role created for Hello World function

Value               arn:aws:iam::012345678910:role/sam-app-zip-

HelloWorldFunctionRole-11ZOGSCG28H0M


Key                 HelloWorldApi

Description         API Gateway endpoint URL for Prod stage for Hello World
 function
Value               https://njzfhdmls0.execute-api.us-west-2.amazonaws.com/Prod/
hello/
```

```
Key                 HelloWorldFunction

Description         Hello World Lambda Function ARN

Value               arn:aws:lambda:us-west-2:012345678910:function:sam-app-

HelloWorldFunction-XPqNX4TBu7qn

-----------------------------------------------------------------------------------------------


Successfully created/updated stack - sam-app-zip in us-west-2
```

5. Gehen Sie wie folgt vor, um Ihre bereitgestellte Anwendung anzuzeigen:

   1. Öffnen Sie die CloudFormation Konsole direkt mit der URL https://console.aws.amazon.com/cloudformation.
   2. Wählen Sie Stacks aus.
   3. Identifizieren Sie Ihren Stack anhand des Anwendungsnamens und wählen Sie ihn aus.

Überprüfen Sie die Änderungen vor der Bereitstellung

Sie können den so konfigurieren AWS SAMCLI, dass Ihr CloudFormation Änderungssatz angezeigt wird und Sie vor der Bereitstellung um Bestätigung gebeten werden.

Um Änderungen vor der Bereitstellung zu bestätigen

1. Geben Sie `sam deploy --guided` währenddessen ein, **Y** um die Änderungen vor der Bereitstellung zu bestätigen.

   ```
   #Shows you resources changes to be deployed and require a 'Y' to initiate deploy
   Confirm changes before deploy [Y/n]: Y
   ```

   Alternativ können Sie Ihre `samconfig.toml` Datei wie folgt ändern:

   ```
   [default.deploy]
   [default.deploy.parameters]
   confirm_changeset = true
   ```

2. Während der Bereitstellung AWS SAMCLI werden Sie aufgefordert, die Änderungen vor der Bereitstellung zu bestätigen. Im Folgenden wird ein Beispiel gezeigt:

```
Waiting for changeset to be created..

CloudFormation stack changeset
-----------------------------------------------------------------------------------------
Operation                LogicalResourceId        ResourceType
 Replacement
-----------------------------------------------------------------------------------------
+ Add                    HelloWorldFunctionHell   AWS::Lambda::Permissio   N/A

                         oWorldPermissionProd     n

+ Add                    HelloWorldFunctionRole   AWS::IAM::Role           N/A

+ Add                    HelloWorldFunction       AWS::Lambda::Function    N/A

+ Add                    ServerlessRestApiDeplo   AWS::ApiGateway::Deplo   N/A

                         yment47fc2d5f9d          yment

+ Add                    ServerlessRestApiProdS   AWS::ApiGateway::Stage   N/A

                         tage

+ Add                    ServerlessRestApi        AWS::ApiGateway::RestA   N/A

                                                  pi

-----------------------------------------------------------------------------------------

Changeset created successfully. arn:aws:cloudformation:us-
west-2:012345678910:changeSet/samcli-deploy1680559234/d9f58a77-98bc-41cd-
b9f4-433a5a450d7a
Previewing CloudFormation changeset before deployment
======================================================
Deploy this changeset? [y/N]: y
```

Geben Sie während der Bereitstellung zusätzliche Parameter an

Sie können zusätzliche Parameterwerte angeben, die bei der Bereitstellung konfiguriert werden sollen. Sie tun dies, indem Sie Ihre AWS SAM Vorlage ändern und Ihren Parameterwert während der Bereitstellung konfigurieren.

Um zusätzliche Parameter anzugeben

1. Ändern Sie den `Parameters` Abschnitt Ihrer AWS SAM Vorlage. Im Folgenden wird ein Beispiel gezeigt:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Globals:
...
Parameters:
  DomainName:
    Type: String
    Default: example
    Description: Domain name
```

2. Führen Sie `sam deploy --guided`. Im Folgenden finden Sie ein Beispiel für eine Ausgabe:

```
sam-app $ sam deploy --guided

Configuring SAM deploy
======================

        Looking for config file [samconfig.toml] :  Found
        Reading default arguments  :  Success

        Setting default arguments for 'sam deploy'
        =========================================
        Stack Name [sam-app-zip]: ENTER
        AWS Region [us-west-2]: ENTER
        Parameter DomainName [example]: ENTER
```

Konfigurieren Sie die Codesignatur für Ihre Lambda-Funktionen

Sie können die Codesignatur für Ihre Lambda-Funktionen bei der Bereitstellung konfigurieren. Dazu ändern Sie Ihre AWS SAM Vorlage und konfigurieren die Codesignatur während der Bereitstellung.

Um die Codesignatur zu konfigurieren

1. Geben Sie `CodeSigningConfigArn` dies in Ihrer AWS SAM Vorlage an. Im Folgenden wird ein Beispiel gezeigt:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: hello_world/
      Handler: app.lambda_handler
      Runtime: python3.7
      CodeSigningConfigArn: arn:aws:lambda:us-east-1:111122223333:code-signing-
config:csc-12e12345db1234567
```

2. Führen Sie `sam deploy --guided`. Sie AWS SAMCLI werden aufgefordert, die Codesignatur zu konfigurieren. Im Folgenden finden Sie ein Beispiel für eine Ausgabe:

```
#Found code signing configurations in your function definitions
Do you want to sign your code? [Y/n]: ENTER
#Please provide signing profile details for the following functions & layers
#Signing profile details for function 'HelloWorld'
Signing Profile Name:
Signing Profile Owner Account ID (optional):
#Signing profile details for layer 'MyLayer', which is used by functions
 {'HelloWorld'}
Signing Profile Name:
Signing Profile Owner Account ID (optional):
```

Best Practices

- Bei der Verwendung `sam deploy` werden die AWS SAMCLI Build-Artefakte Ihrer Anwendung bereitgestellt, die sich im `.aws-sam` Verzeichnis befinden. Wenn Sie Änderungen an den

Originaldateien Ihrer Anwendung vornehmen, führen Sie vor der Bereitstellung den Befehl aus, `sam build` um das `.aws-sam` Verzeichnis zu aktualisieren.

- Wenn Sie eine Anwendung zum ersten Mal bereitstellen, verwenden Sie diese Option, `sam deploy --guided` um die Bereitstellungseinstellungen zu konfigurieren. Für nachfolgende Bereitstellungen können Sie die Bereitstellung mit Ihren konfigurierten Einstellungen verwendensam deploy.

Optionen für Sam Deploy

Die folgenden Optionen werden häufig für verwendetsam deploy. Eine Liste aller Optionen finden Sie untersam deploy.

Verwenden Sie den geführten interaktiven Ablauf, um Ihre Anwendung bereitzustellen

Verwenden Sie die `--guided` Option, um die Bereitstellungseinstellungen Ihrer Anwendung über einen interaktiven Ablauf zu konfigurieren. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam deploy --guided
```

Die Bereitstellungseinstellungen Ihrer Anwendung werden in der `samconfig.toml` Datei Ihres Projekts gespeichert. Weitere Informationen hierzu finden Sie unter Konfigurieren Sie die Projekteinstellungen.

Fehlerbehebung

Informationen zur Fehlerbehebung bei AWS SAMCLI finden Sie unterAWS SAMCLIProblembehandlung.

Beispiele

Stellen Sie eine Hello World-Anwendung bereit, die eine Lambda-Funktion enthält, die als ZIP-Dateiarchiv verpackt ist

Ein Beispiel finden Sie Schritt 3: Stellen Sie Ihre Anwendung auf dem bereit AWS Cloud im Tutorial zur Hello World-Anwendung.

Stellen Sie eine Hello World-Anwendung bereit, die eine Lambda-Funktion enthält, die als Container-Image verpackt ist

Zunächst erstellen wir `sam init` damit unsere Hello World-Anwendung. Während des interaktiven Ablaufs wählen wir die `Python3.9` Laufzeit und den `Image` Pakettyp aus.

```
$ sam init
...
Which template source would you like to use?
        1 - AWS Quick Start Templates
        2 - Custom Template Location
Choice: 1

Choose an AWS Quick Start application template
        1 - Hello World Example
        2 - Multi-step workflow
        ...
Template: 1

Use the most popular runtime and package type? (Python and zip) [y/N]:  ENTER

Which runtime would you like to use?
        1 - aot.dotnet7 (provided.al2)
        ...
        15 - nodejs12.x
        16 - python3.9
        17 - python3.8
        ...
Runtime: 16

What package type would you like to use?
        1 - Zip
        2 - Image
Package type: 2

Based on your selections, the only dependency manager available is pip.
We will proceed copying the template using pip.
...
Project name [sam-app]: ENTER

    -----------------------
    Generating application:
    -----------------------
```

```
    Name: sam-app
    Base Image: amazon/python3.9-base
    Architectures: x86_64
    Dependency Manager: pip
    Output Directory: .
    Configuration file: sam-app/samconfig.toml

    Next steps can be found in the README file at sam-app/README.md
...
```

Als Nächstes `cd` gehen wir in das Stammverzeichnis unseres Projekts und führen es aus`sam build`. Das AWS SAMCLI erstellt unsere Lambda-Funktion lokal mitDocker.

```
sam-app $ sam build
Building codeuri: /Users/.../sam-app runtime: None metadata: {'Dockerfile':
 'Dockerfile', 'DockerContext': '/Users/.../sam-app/hello_world', 'DockerTag':
 'python3.9-v1'} architecture: x86_64 functions: HelloWorldFunction
Building image for HelloWorldFunction function
Setting DockerBuildArgs: {} for HelloWorldFunction function
Step 1/5 : FROM public.ecr.aws/lambda/python:3.9
 ---> 0a5e3da309aa
Step 2/5 : COPY requirements.txt ./
 ---> abc4e82e85f9
Step 3/5 : RUN python3.9 -m pip install -r requirements.txt -t .
 ---> [Warning] The requested image's platform (linux/amd64) does not match the
 detected host platform (linux/arm64/v8) and no specific platform was requested
 ---> Running in 43845e7aa22d
Collecting requests
  Downloading requests-2.28.2-py3-none-any.whl (62 kB)
     ######################################## 62.8/62.8 KB 829.5 kB/s eta 0:00:00
Collecting idna<4,>=2.5
  Downloading idna-3.4-py3-none-any.whl (61 kB)
     ######################################## 61.5/61.5 KB 2.4 MB/s eta 0:00:00
Collecting charset-normalizer<4,>=2
  Downloading charset_normalizer-3.1.0-cp39-cp39-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl (199 kB)
     ######################################## 199.2/199.2 KB 2.1 MB/s eta 0:00:00
Collecting certifi>=2017.4.17
  Downloading certifi-2022.12.7-py3-none-any.whl (155 kB)
     ######################################## 155.3/155.3 KB 10.2 MB/s eta 0:00:00
Collecting urllib3<1.27,>=1.21.1
  Downloading urllib3-1.26.15-py2.py3-none-any.whl (140 kB)
     ######################################## 140.9/140.9 KB 9.1 MB/s eta 0:00:00
```

```
Installing collected packages: urllib3, idna, charset-normalizer, certifi, requests
Successfully installed certifi-2022.12.7 charset-normalizer-3.1.0 idna-3.4
 requests-2.28.2 urllib3-1.26.15
Removing intermediate container 43845e7aa22d
 ---> cab8ace899ce
Step 4/5 : COPY app.py ./
 ---> 4146f3cd69f2
Step 5/5 : CMD ["app.lambda_handler"]
 ---> [Warning] The requested image's platform (linux/amd64) does not match the
 detected host platform (linux/arm64/v8) and no specific platform was requested
 ---> Running in f4131ddffb31
Removing intermediate container f4131ddffb31
 ---> d2f5180b2154
Successfully built d2f5180b2154
Successfully tagged helloworldfunction:python3.9-v1


Build Succeeded

Built Artifacts  : .aws-sam/build
Built Template   : .aws-sam/build/template.yaml

Commands you can use next
=========================
[*] Validate SAM template: sam validate
[*] Invoke Function: sam local invoke
[*] Test Function in the Cloud: sam sync --stack-name {{stack-name}} --watch
[*] Deploy: sam deploy --guided
```

Als Nächstes starten wir, `sam deploy --guided` um unsere Anwendung bereitzustellen. Das AWS SAMCLI führt uns durch die Konfiguration unserer Bereitstellungseinstellungen. Anschließend AWS SAMCLI stellt der unsere Anwendung auf dem AWS Cloud bereit.

```
sam-app $ sam deploy --guided

Configuring SAM deploy
======================

        Looking for config file [samconfig.toml] :  Found
        Reading default arguments  :  Success

        Setting default arguments for 'sam deploy'
        =========================================
```

```
        Stack Name [sam-app]: ENTER
        AWS Region [us-west-2]: ENTER
        #Shows you resources changes to be deployed and require a 'Y' to initiate
 deploy
        Confirm changes before deploy [Y/n]: ENTER
        #SAM needs permission to be able to create roles to connect to the resources in
 your template
        Allow SAM CLI IAM role creation [Y/n]: ENTER
        #Preserves the state of previously provisioned resources when an operation
 fails
        Disable rollback [y/N]: ENTER
        HelloWorldFunction may not have authorization defined, Is this okay? [y/N]: y
        Save arguments to configuration file [Y/n]: ENTER
        SAM configuration file [samconfig.toml]: ENTER
        SAM configuration environment [default]: ENTER

        Looking for resources needed for deployment:

        Managed S3 bucket: aws-sam-cli-managed-default-samclisam-s3-demo-
bucket-1a4x26zbcdkqr
        A different default S3 bucket can be set in samconfig.toml and auto resolution
 of buckets turned off by setting resolve_s3=False

        Parameter "stack_name=sam-app" in [default.deploy.parameters] is defined as a
 global parameter [default.global.parameters].
        This parameter will be only saved under [default.global.parameters] in /
Users/.../sam-app/samconfig.toml.

        Saved arguments to config file
        Running 'sam deploy' for future deployments will use the parameters saved
 above.
        The above parameters can be changed by modifying samconfig.toml
        Learn more about samconfig.toml syntax at
        https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/
serverless-sam-cli-config.html

e95fc5e75742: Pushed
d8df51e7bdd7: Pushed
b1d0d7e0b34a: Pushed
0071317b94d8: Pushed
d98f98baf147: Pushed
2d244e0816c6: Pushed
eb2eeb1ebe42: Pushed
a5ca065a3279: Pushed
```

```
fe9e144829c9: Pushed
helloworldfunction-d2f5180b2154-python3.9-v1: digest:
 sha256:cceb71401b47dc3007a7a1e1f2e0baf162999e0e6841d15954745ecc0c447533 size: 2206

        Deploying with following values
        ===============================
        Stack name                   : sam-app
        Region                       : us-west-2
        Confirm changeset            : True
        Disable rollback             : False
        Deployment image repository  :
                                       {
                                           "HelloWorldFunction":
 "012345678910.dkr.ecr.us-west-2.amazonaws.com/samapp7427b055/
helloworldfunction19d43fc4repo"
                                       }
        Deployment s3 bucket         : aws-sam-cli-managed-default-samclisam-s3-demo-
bucket-1a4x26zbcdkqr
        Capabilities                 : ["CAPABILITY_IAM"]
        Parameter overrides          : {}
        Signing Profiles             : {}

Initiating deployment
=====================

HelloWorldFunction may not have authorization defined.
        Uploading to sam-app/682ad27c7cf7a17c7f77a1688b0844f2.template  1328 / 1328
 (100.00%)


Waiting for changeset to be created..

CloudFormation stack changeset
-----------------------------------------------------------------------------------------------
Operation                LogicalResourceId        ResourceType             Replacement

-----------------------------------------------------------------------------------------------
+ Add                    HelloWorldFunctionHell   AWS::Lambda::Permissio   N/A

                         oWorldPermissionProd     n

+ Add                    HelloWorldFunctionRole   AWS::IAM::Role           N/A
```

```
+ Add                    HelloWorldFunction       AWS::Lambda::Function    N/A

+ Add                    ServerlessRestApiDeplo   AWS::ApiGateway::Deplo   N/A

                         yment47fc2d5f9d          yment

+ Add                    ServerlessRestApiProdS   AWS::ApiGateway::Stage   N/A

                         tage

+ Add                    ServerlessRestApi        AWS::ApiGateway::RestA   N/A

                                                  pi

-----------------------------------------------------------------------------------------------


Changeset created successfully. arn:aws:cloudformation:us-
west-2:012345678910:changeSet/samcli-deploy1680634124/0ffd4faf-2e2b-487e-
b9e0-9116e8299ac4


Previewing CloudFormation changeset before deployment
======================================================
Deploy this changeset? [y/N]: y

2023-04-04 08:49:15 - Waiting for stack create/update to complete

CloudFormation events from stack operations (refresh every 5.0 seconds)
-----------------------------------------------------------------------------------------------
ResourceStatus           ResourceType             LogicalResourceId
 ResourceStatusReason
-----------------------------------------------------------------------------------------------
CREATE_IN_PROGRESS       AWS::CloudFormation::S   sam-app                  User
 Initiated
                         tack

CREATE_IN_PROGRESS       AWS::IAM::Role           HelloWorldFunctionRole   -

CREATE_IN_PROGRESS       AWS::IAM::Role           HelloWorldFunctionRole   Resource
 creation
                                                                           Initiated
```

```
CREATE_COMPLETE          AWS::IAM::Role           HelloWorldFunctionRole   -

CREATE_IN_PROGRESS       AWS::Lambda::Function    HelloWorldFunction       -

CREATE_IN_PROGRESS       AWS::Lambda::Function    HelloWorldFunction       Resource
 creation                                                                  Initiated

CREATE_COMPLETE          AWS::Lambda::Function    HelloWorldFunction       -

CREATE_IN_PROGRESS       AWS::ApiGateway::RestA   ServerlessRestApi        -
                         pi

CREATE_IN_PROGRESS       AWS::ApiGateway::RestA   ServerlessRestApi        Resource
 creation                pi                                               Initiated

CREATE_COMPLETE          AWS::ApiGateway::RestA   ServerlessRestApi        -
                         pi

CREATE_IN_PROGRESS       AWS::Lambda::Permissio   HelloWorldFunctionHell   -
                         n                        oWorldPermissionProd

CREATE_IN_PROGRESS       AWS::ApiGateway::Deplo   ServerlessRestApiDeplo   -
                         yment                    yment47fc2d5f9d

CREATE_IN_PROGRESS       AWS::Lambda::Permissio   HelloWorldFunctionHell   Resource
 creation                n                        oWorldPermissionProd     Initiated

CREATE_IN_PROGRESS       AWS::ApiGateway::Deplo   ServerlessRestApiDeplo   Resource
 creation                yment                    yment47fc2d5f9d          Initiated

CREATE_COMPLETE          AWS::ApiGateway::Deplo   ServerlessRestApiDeplo   -
                         yment                    yment47fc2d5f9d

CREATE_IN_PROGRESS       AWS::ApiGateway::Stage   ServerlessRestApiProdS   -
```

```
                                                tage

CREATE_IN_PROGRESS       AWS::ApiGateway::Stage   ServerlessRestApiProdS   Resource
 creation
                                                tage                     Initiated

CREATE_COMPLETE          AWS::ApiGateway::Stage   ServerlessRestApiProdS   -

                                                tage

CREATE_COMPLETE          AWS::Lambda::Permissio   HelloWorldFunctionHell   -

                         n                        oWorldPermissionProd

CREATE_COMPLETE          AWS::CloudFormation::S   sam-app                  -

                         tack

-------------------------------------------------------------------------------------------------

CloudFormation outputs from deployed stack
-------------------------------------------------------------------------------------------------
Outputs

-------------------------------------------------------------------------------------------------
Key                 HelloWorldFunctionIamRole

Description         Implicit IAM Role created for Hello World function

Value               arn:aws:iam::012345678910:role/sam-app-HelloWorldFunctionRole-
JFML1JOKHJ71

Key                 HelloWorldApi

Description         API Gateway endpoint URL for Prod stage for Hello World function

Value               https://endlwiqqod.execute-api.us-west-2.amazonaws.com/Prod/hello/


Key                 HelloWorldFunction

Description         Hello World Lambda Function ARN
```

```
Value                arn:aws:lambda:us-west-2:012345678910:function:sam-app-
HelloWorldFunction-
kyg6Y2iNRUPg

------------------------------------------------------------------------------------------------


Successfully created/updated stack - sam-app in us-west-2
```

Weitere Informationen

Weitere Informationen zur Verwendung des AWS SAMCLI `sam deploy` Befehls finden Sie in den folgenden Abschnitten:

- Der komplette AWS SAM Workshop: Modul 3 — Manuelles Bereitstellen — Erfahren Sie, wie Sie eine serverlose Anwendung mit dem AWS SAMCLI erstellen, paketieren und bereitstellen.

Optionen für die Bereitstellung Ihrer Anwendung mit AWS SAM

Mit AWS SAM können Sie Ihre Anwendung manuell bereitstellen und Bereitstellungen automatisieren. Verwenden Sie die AWS SAMCLI, um Ihre Anwendung manuell bereitzustellen. Verwenden Sie zur Automatisierung der Bereitstellung Pipelines und ein CI/CD-System (Continuous Integration and Continuous Deployment). Die Themen in diesem Abschnitt enthalten Informationen zu beiden Ansätzen.

Themen

- Wie benutzt man den AWS SAMCLI für die manuelle Bereitstellung
- Bereitstellen mit CI/CD Systemen und Pipelines
- Schrittweise Bereitstellungen
- Problembehandlung bei Bereitstellungen mit dem AWS SAMCLI
- Weitere Informationen

Wie benutzt man den AWS SAMCLI für die manuelle Bereitstellung

Nachdem Sie Ihre serverlose Anwendung lokal entwickelt und getestet haben, können Sie Ihre Anwendung mithilfe des sam deploy Befehls bereitstellen.

Geben Sie das AWS SAM Kennzeichen an, damit Sie mit Anweisungen durch die --guided Bereitstellung geführt werden. Wenn Sie dieses Flag angeben, komprimiert der sam deploy Befehl Ihre Anwendungsartefakte und lädt sie entweder in Amazon Simple Storage Service (Amazon S3) (für ZIP-Dateiarchive) oder in Amazon Elastic Container Registry (Amazon ECR) (für Container-Images) hoch. Der Befehl stellt dann Ihre Anwendung in der Cloud bereit. AWS

Beispiel:

```
# Deploy an application using prompts:
sam deploy --guided
```

Bereitstellen mit CI/CD Systemen und Pipelines

AWS SAM unterstützt Sie bei der Automatisierung der Bereitstellung mithilfe von Pipelines und einem CI/CD-System (Continuous Integration and Continuous Deployment). AWS SAM kann verwendet werden, um Pipelines zu erstellen und CI/CD Aufgaben für serverlose Anwendungen zu vereinfachen. Mehrere CI/CD Systeme unterstützen AWS SAM Build-Container-Images und bieten AWS SAM außerdem eine Reihe von Standard-Pipeline-Vorlagen für mehrere CI/CD Systeme, in denen die bewährten Bereitstellungsmethoden zusammengefasst AWS sind.

Weitere Informationen finden Sie unter Verwendung von CI/CD Systemen und Pipelines für die Bereitstellung mit AWS SAM.

Schrittweise Bereitstellungen

Wenn Sie Ihre AWS SAM Anwendung schrittweise und nicht alle auf einmal bereitstellen möchten, können Sie Bereitstellungskonfigurationen angeben, die Folgendes AWS CodeDeploy bieten: Weitere Informationen finden Sie CodeDeployim AWS CodeDeploy Benutzerhandbuch unter Arbeiten mit Bereitstellungskonfigurationen.

Informationen zur Konfiguration Ihrer AWS SAM Anwendung für die schrittweise Bereitstellung finden Sie unterSchrittweise Bereitstellung serverloser Anwendungen mit AWS SAM.

Problembehandlung bei Bereitstellungen mit dem AWS SAMCLI

AWS SAMCLIFehler: „Sicherheitseinschränkungen nicht erfüllt“

Beim Ausführen sam deploy --guided wird Ihnen die Frage angezeigt`HelloWorldFunction may not have authorization defined, Is this okay? [y/N]`. Wenn Sie auf diese Aufforderung mit **`N`** (der Standardantwort) antworten, wird der folgende Fehler angezeigt:

```
Error: Security Constraints Not Satisfied
```

Die Aufforderung informiert Sie darüber, dass für die Anwendung, die Sie bereitstellen möchten, möglicherweise eine Amazon API Gateway Gateway-API ohne Autorisierung konfiguriert ist. Wenn Sie **N** auf diese Aufforderung antworten, sagen Sie, dass dies nicht in Ordnung ist.

Um dieses Problem zu beheben, haben Sie die folgenden Optionen:

- Konfigurieren Sie Ihre Anwendung mit Autorisierung. Informationen zur Konfiguration der Autorisierung finden Sie unterKontrollieren Sie den API-Zugriff mit Ihrer AWS SAM Vorlage.
- Beantworten Sie diese Frage mit, **Y** um anzugeben, dass Sie mit der Bereitstellung einer Anwendung einverstanden sind, für die eine API-Gateway-API ohne Autorisierung konfiguriert ist.

Weitere Informationen

Praktische Beispiele für die Bereitstellung serverloser Anwendungen finden Sie im Folgenden in The Complete AWS SAM Workshop:

- Modul 3 — Manuelles Bereitstellen — Erfahren Sie, wie Sie eine serverlose Anwendung mithilfe von erstellen, verpacken und bereitstellen. AWS SAMCLI
- Modul 4 — CI/CD — Erfahren Sie, wie Sie die Erstellungs-, Paketierungs- und Bereitstellungsphasen automatisieren können, indem Sie eine CI/CD-Pipeline (Continuous Integration and Delivery) einrichten.

Verwendung von CI/CD Systemen und Pipelines für die Bereitstellung mit AWS SAM

AWS SAM unterstützt Unternehmen bei der Erstellung von Pipelines für ihre bevorzugten CI/CD Systeme, sodass sie CI/CD mit minimalem Aufwand die Vorteile nutzen können, wie z. B. die Beschleunigung der Bereitstellungshäufigkeit, die Verkürzung der Vorlaufzeit für Änderungen und die Reduzierung von Bereitstellungsfehlern.

AWS SAM vereinfacht CI/CD Aufgaben für serverlose Anwendungen mithilfe von Build-Container-Images. Die bereitgestellten Images beinhalten die Tools AWS SAMCLI und Build-Tools für eine

Reihe unterstützter AWS Lambda Laufzeiten. AWS SAM Dies macht es einfacher, serverlose Anwendungen mithilfe von zu erstellen und zu paketieren. AWS SAMCLI Diese Images machen es den Teams auch leichter, ihre eigenen Images für CI/CD Systeme zu erstellen und zu verwalten. Weitere Informationen zum AWS SAM Erstellen von Container-Images finden Sie unterBildrepositorien für AWS SAM.

Das AWS SAM Erstellen von Container-Images wird von mehreren CI/CD Systemen unterstützt. Welches CI/CD System Sie verwenden sollten, hängt von mehreren Faktoren ab. Dazu gehört, ob Ihre Anwendung eine einzelne Laufzeit oder mehrere Laufzeiten verwendet oder ob Sie Ihre Anwendung in einem Container-Image oder direkt auf einem Hostcomputer, entweder einer virtuellen Maschine (VM) oder einem Bare-Metal-Host, erstellen möchten.

AWS SAM bietet außerdem eine Reihe von Standard-Pipeline-Vorlagen für mehrere CI/CD Systeme, in denen die bewährten Bereitstellungsmethoden zusammengefasst AWS sind. Diese Standard-Pipeline-Vorlagen verwenden standardmäßige JSON/YAML Pipeline-Konfigurationsformate, und die integrierten Best Practices helfen bei der Durchführung von Bereitstellungen mit mehreren Konten und Regionen und stellen sicher, dass Pipelines keine unbeabsichtigten Änderungen an der Infrastruktur vornehmen können.

Sie haben zwei Hauptoptionen für die Bereitstellung Ihrer serverlosen Anwendungen: 1) Ändern Sie Ihre bestehende Pipeline-Konfiguration, sodass AWS SAMCLI Befehle verwendet werden, oder 2) Generieren Sie eine CI/CD Beispiel-Pipeline-Konfiguration, die Sie als Ausgangspunkt für Ihre eigene Anwendung verwenden können. AWS SAM

Themen

- Was ist eine Pipeline?
- So werden lokale Dateien bei der Bereitstellung AWS SAM hochgeladen
- Generieren Sie eine CI/CD Starter-Pipeline mit AWS SAM
- So passen Sie Starter-Pipelines an mit AWS SAM
- Automatisieren Sie die Bereitstellung Ihrer AWS SAM Anwendung
- So verwenden Sie die OIDC-Authentifizierung mit Pipelines AWS SAM

Was ist eine Pipeline?

Eine Pipeline ist eine automatisierte Abfolge von Schritten, die ausgeführt werden, um eine neue Version einer Anwendung zu veröffentlichen. Mit AWS SAM können Sie viele gängige CI/CD

Systeme für die Bereitstellung Ihrer Anwendungen verwenden, darunter Jenkins AWS CodePipeline, GitLab CI/CD und Actions. GitHub

Pipeline-Vorlagen enthalten bewährte AWS Bereitstellungsmethoden, die Sie bei Bereitstellungen mit mehreren Konten und Regionen unterstützen. AWS Umgebungen wie Entwicklung und Produktion existieren in der Regel in unterschiedlichen Konten. AWS Auf diese Weise können Entwicklungsteams sichere Bereitstellungspipelines konfigurieren, ohne unbeabsichtigte Änderungen an der Infrastruktur vorzunehmen.

Sie können auch Ihre eigenen benutzerdefinierten Pipeline-Vorlagen bereitstellen, um die Pipelines zwischen den Entwicklungsteams zu standardisieren.

So werden lokale Dateien bei der Bereitstellung AWS SAM hochgeladen

Wenn Sie Ihre Anwendung auf dem bereitstellen AWS Cloud, AWS CloudFormation müssen Ihre lokalen Dateien zunächst in einen zugänglichen AWS Service wie Amazon Simple Storage Service (Amazon S3) hochgeladen werden. Dies schließt lokale Dateien ein, auf die Ihre AWS SAM Vorlage verweist. Um diese Anforderung zu erfüllen, AWS SAMCLI geht der wie folgt vor, wenn Sie den `sam package` Befehl `sam deploy` oder verwenden:

1. Lädt Ihre lokalen Dateien automatisch in einen AWS Dienst hoch, auf den zugegriffen werden kann.
2. Aktualisiert Ihre Anwendungsvorlage automatisch, sodass sie auf den neuen Dateipfad verweist.

Topics

- Demo: Verwenden Sie AWS SAMCLI den Lambda-Funktionscode zum Hochladen
- Unterstützte Anwendungsfälle
- Weitere Informationen

Demo: Verwenden Sie AWS SAMCLI den Lambda-Funktionscode zum Hochladen

In dieser Demo initialisieren wir die Hello World-Beispielanwendung mit einem Pakettyp .zip für unsere Lambda-Funktion. Wir verwenden den AWS SAMCLI, um unseren Lambda-Funktionscode automatisch auf Amazon S3 hochzuladen und auf seinen neuen Pfad in unserer Anwendungsvorlage zu verweisen.

Zuerst starten wir, `sam init` um unsere Hello World-Anwendung zu initialisieren.

```
$ sam init
...
Which template source would you like to use?
        1 - AWS Quick Start Templates
        2 - Custom Template Location
Choice: 1

Choose an AWS Quick Start application template
        1 - Hello World Example
        2 - Multi-step workflow
        ...
Template: 1

Use the most popular runtime and package type? (Python and zip) [y/N]: y

Would you like to enable X-Ray tracing on the function(s) in your application?  [y/
N]: ENTER

Would you like to enable monitoring using CloudWatch Application Insights?
For more info, please view https://docs.aws.amazon.com/AmazonCloudWatch/latest/
monitoring/cloudwatch-application-insights.html [y/N]: ENTER

Project name [sam-app]: demo

    -----------------------
    Generating application:
    -----------------------
    Name: demo
    Runtime: python3.9
    Architectures: x86_64
    Dependency Manager: pip
    Application Template: hello-world
    Output Directory: .
    Configuration file: demo/samconfig.toml

...
```

Unser Lambda-Funktionscode ist im `hello_world` Unterverzeichnis unseres Projekts organisiert.

```
demo
### README.md
### hello_world
#   ### __init__.py
```

```
#   ### app.py
#   ### requirements.txt
### template.yaml
### tests
```

In unserer AWS SAM Vorlage verweisen wir mithilfe der `CodeUri` Eigenschaft auf den lokalen Pfad zu unserem Lambda-Funktionscode.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function # More info about Function Resource:
 https://github.com/awslabs/serverless-application-model/blob/master/
versions/2016-10-31.md#awsserverlessfunction
    Properties:
      CodeUri: hello_world/
      Handler: app.lambda_handler
      Runtime: python3.9
      ...
```

Als Nächstes erstellen wir unsere `sam build` Anwendung und bereiten sie auf die Bereitstellung vor.

```
$ sam build
Starting Build use cache
Manifest file is changed (new hash: 3298f13049d19cffaa37ca931dd4d421) or dependency
 folder (.aws-sam/deps/7896875f-9bcc-4350-8adb-2c1d543627a1) is missing for
 (HelloWorldFunction), downloading dependencies and copying/building source
Building codeuri: /Users/.../demo/hello_world runtime: python3.9 metadata: {}
 architecture: x86_64 functions: HelloWorldFunction
Running PythonPipBuilder:CleanUp
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource
Running PythonPipBuilder:CopySource

Build Succeeded

Built Artifacts  : .aws-sam/build
Built Template   : .aws-sam/build/template.yaml
...
```

Als Nächstes starten wir, `sam deploy --guided` um unsere Anwendung bereitzustellen.

```
$ sam deploy --guided

Configuring SAM deploy
======================

        Looking for config file [samconfig.toml] :  Found
        Reading default arguments  :  Success

        Setting default arguments for 'sam deploy'
        =========================================
        Stack Name [demo]: ENTER
        AWS Region [us-west-2]: ENTER
        #Shows you resources changes to be deployed and require a 'Y' to initiate
 deploy
        Confirm changes before deploy [Y/n]: n
        #SAM needs permission to be able to create roles to connect to the resources in
 your template
        Allow SAM CLI IAM role creation [Y/n]: ENTER
        #Preserves the state of previously provisioned resources when an operation
 fails
        Disable rollback [y/N]: ENTER
        HelloWorldFunction may not have authorization defined, Is this okay? [y/N]: y
        Save arguments to configuration file [Y/n]: ENTER
        SAM configuration file [samconfig.toml]: ENTER
        SAM configuration environment [default]: ENTER

        Looking for resources needed for deployment:
        ...
        Saved arguments to config file
        Running 'sam deploy' for future deployments will use the parameters saved
 above.
        The above parameters can be changed by modifying samconfig.toml
        Learn more about samconfig.toml syntax at
        https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/
serverless-sam-cli-config.html

File with same data already exists at demo/da3c598813f1c2151579b73ad788cac8, skipping
 upload

        Deploying with following values
        ===============================
        Stack name                   : demo
```

```
        Region                      : us-west-2
        Confirm changeset           : False
        Disable rollback            : False
        Deployment s3 bucket        : aws-sam-cli-managed-default-samclisam-s3-demo-
bucket-1a4x26zbcdkqr
        Capabilities                : ["CAPABILITY_IAM"]
        Parameter overrides         : {}
        Signing Profiles            : {}

Initiating deployment
=====================
...
Waiting for changeset to be created..
CloudFormation stack changeset
---------------------------------------------------------------------------------------------
Operation                LogicalResourceId        ResourceType             Replacement

---------------------------------------------------------------------------------------------
+ Add                    HelloWorldFunctionHell   AWS::Lambda::Permissio   N/A

                         oWorldPermissionProd     n

+ Add                    HelloWorldFunctionRole   AWS::IAM::Role           N/A

...
---------------------------------------------------------------------------------------------

Changeset created successfully. arn:aws:cloudformation:us-
west-2:012345678910:changeSet/samcli-deploy1680906292/1164338d-72e7-4593-a372-
f2b3e67f542f

2023-04-07 12:24:58 - Waiting for stack create/update to complete

CloudFormation events from stack operations (refresh every 5.0 seconds)
---------------------------------------------------------------------------------------------
ResourceStatus           ResourceType             LogicalResourceId
 ResourceStatusReason
---------------------------------------------------------------------------------------------
CREATE_IN_PROGRESS       AWS::IAM::Role           HelloWorldFunctionRole   -

CREATE_IN_PROGRESS       AWS::IAM::Role           HelloWorldFunctionRole   Resource
 creation
                                                                           Initiated
```

```
...
---------------------------------------------------------------------------------------------------
CloudFormation outputs from deployed stack
---------------------------------------------------------------------------------------------------
Outputs

---------------------------------------------------------------------------------------------------
Key                 HelloWorldFunctionIamRole

Description         Implicit IAM Role created for Hello World function

Value               arn:aws:iam::012345678910:role/demo-HelloWorldFunctionRole-
VQ4CU7UY7S2K

Key                 HelloWorldApi

Description         API Gateway endpoint URL for Prod stage for Hello World function

Value               https://satnon55e9.execute-api.us-west-2.amazonaws.com/Prod/hello/


Key                 HelloWorldFunction

Description         Hello World Lambda Function ARN

Value               arn:aws:lambda:us-west-2:012345678910:function:demo-

HelloWorldFunction-G14inKTmSQvK

---------------------------------------------------------------------------------------------------
Successfully created/updated stack - demo in us-west-2
```

Während der Bereitstellung lädt der AWS SAMCLI automatisch unseren Lambda-Funktionscode auf Amazon S3 hoch und aktualisiert unsere Vorlage. Unsere geänderte Vorlage in der CloudFormation Konsole spiegelt den Amazon S3 S3-Bucket-Pfad wider.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
```

```
      CodeUri: s3://aws-sam-cli-managed-default-samclisam-s3-demo-bucket-1a4x26zbcdkqr/
demo/da3c598813f1c2151579b73ad788cac8
      Handler: app.lambda_handler
      ...
```

Unterstützte Anwendungsfälle

Das AWS SAMCLI kann diesen Vorgang für eine Reihe von Dateitypen, CloudFormation Ressourcentypen und CloudFormation Makros automatisch vereinfachen.

Dateitypen

Anwendungsdateien und Docker Bilder werden unterstützt.

CloudFormation Ressourcentypen

Im Folgenden finden Sie eine Liste der unterstützten Ressourcentypen und ihrer Eigenschaften:

Ressource	Eigenschaften
`AWS::ApiGateway::RestApi`	`BodyS3Location`
`AWS::ApiGatewayV2::Api`	`BodyS3Location`

`AWS::AppSync:FunctionConfiguration`	`CodeS3Location`
	`RequestMappingTemplateS3Location`
	`ResponseMappingTemplateS3Location`

`AWS::AppSync::GraphQLSchema`	`DefinitionS3Location`

`AWS::AppSync::Resolver`	`CodeS3Location`
	`RequestMappingTemplateS3Location`
	`ResponseMappingTemplateS3Location`

Ressource	Eigenschaften
AWS::CloudFormation::ModuleVersion	ModulePackage
AWS::CloudFormation::ResourceVersion	SchemaHandlerPackage
AWS::ECR::Repository	RepositoryName
AWS::ElasticBeanstalk::ApplicationVersion	SourceBundle
AWS::Glue::Job	Command.ScriptLocation
AWS::Lambda::Function	Code Code.ImageUri
AWS::Lambda::LayerVersion	Content
AWS::Serverless::Api	DefinitionUri
AWS::Serverless::Function	CodeUri ImageUri
AWS::Serverless::GraphQLApi	SchemaUri Function.CodeUri Resolver.CodeUri
AWS::Serverless::HttpApi	DefinitionUri
AWS::Serverless::LayerVersion	ContentUri
AWS::Serverless::StateMachine	DefinitionUri

Ressource	Eigenschaften
`AWS::StepFunctions::StateMachine`	`DefinitionS3Location`

CloudFormation Makros

Dateien, auf die mit dem `AWS::Include` Transform-Makro verwiesen wird, werden unterstützt.

Weitere Informationen

Weitere Informationen zur `AWS::Include` Transformation finden Sie unter AWS::Include Transformation im AWS CloudFormation Benutzerhandbuch.

Ein Beispiel für die Verwendung der `AWS::Include` Transformation in einer AWS SAM Vorlage finden Sie im API-Gateway-HTTP-API-zu-SQS-Muster auf Serverless Land.

Generieren Sie eine CI/CD Starter-Pipeline mit AWS SAM

Wenn Sie bereit sind, die Bereitstellung zu automatisieren, können Sie eine der AWS SAM Start-Pipeline-Vorlagen verwenden, um eine Bereitstellungspipeline für das CI/CD System zu generieren, das Sie verwenden möchten. Ihre Bereitstellungspipeline ist das, was Sie konfigurieren und verwenden, um die Bereitstellung Ihrer serverlosen Anwendung zu automatisieren. Eine Vorlage für die Starter-Pipeline ist vorkonfiguriert, damit Sie Ihre Bereitstellungspipeline für Ihre serverlose Anwendung schnell einrichten können.

Mit einer Vorlage für eine Starter-Pipeline können Sie mithilfe des Befehls Pipelines innerhalb von Minuten generieren. sam pipeline init

Die Vorlagen für die Starter-Pipeline verwenden die vertraute JSON/YAML Syntax des CI/CD Systems und beinhalten bewährte Methoden wie die Verwaltung von Artefakten über mehrere Konten und Regionen hinweg und die Verwendung der Mindestanzahl an Berechtigungen, die für die Bereitstellung der Anwendung erforderlich sind. Derzeit unterstützt die AWS SAM CLI die Generierung von CI/CD Starter-Pipeline-Konfigurationen für Jenkins AWS CodePipeline, GitLab CI/CD, GitHub Actions und Bitbucket-Pipelines.

Hier sind die wichtigsten Aufgaben, die Sie ausführen müssen, um eine Starter-Pipeline-Konfiguration zu generieren:

1. Infrastrukturressourcen erstellen — Ihre Pipeline benötigt bestimmte AWS Ressourcen, z. B. den IAM-Benutzer und Rollen mit den erforderlichen Berechtigungen, einen Amazon S3 S3-Bucket und optional ein Amazon ECR-Repository.
2. Connect dein Git-Repository mit deinem CI/CD System — Dein CI/CD System muss wissen, welches Git-Repository die Ausführung der Pipeline auslöst. Beachten Sie, dass dieser Schritt möglicherweise nicht erforderlich ist, je nachdem, welche Kombination aus Git-Repository und CI/CD System Sie verwenden.
3. Generieren Sie Ihre Pipeline-Konfiguration — In diesem Schritt wird eine Starter-Pipeline-Konfiguration generiert, die zwei Bereitstellungsphasen umfasst.
4. Übernehmen Sie Ihre Pipeline-Konfiguration in Ihr Git-Repository — Dieser Schritt ist notwendig, um sicherzustellen, dass Ihr CI/CD System Ihre Pipeline-Konfiguration kennt. Er wird ausgeführt, wenn die Änderungen festgeschrieben wurden.

Nachdem Sie die Starter-Pipeline-Konfiguration generiert und in Ihr Git-Repository übernommen haben, wird Ihre Pipeline automatisch ausgeführt, wenn jemand eine Codeänderung an dieses Repository festschreibt.

Die Reihenfolge dieser Schritte und die Einzelheiten der einzelnen Schritte variieren je nach CI/CD-System:

- Wenn Sie verwenden, finden Sie weitere Informationen AWS CodePipeline unter. Generierung der Starter-Pipeline für AWS CodePipeline in AWS SAM
- Wenn Sie Jenkins-, GitLab CI/CD-, GitHub Actions- oder Bitbucket-Pipelines verwenden, finden Sie weitere Informationen unter. Wird verwendet AWS SAM , um Starter-Pipelines für Jenkins, GitLab CI/CD, Actions und Bitbucket-Pipelines zu generieren GitHub

Generierung der Starter-Pipeline für AWS CodePipeline in AWS SAM

Um eine Starter-Pipeline-Konfiguration für zu generieren AWS CodePipeline, führen Sie die folgenden Aufgaben in dieser Reihenfolge aus:

1. Infrastrukturressourcen erstellen
2. Generieren Sie die Pipeline-Konfiguration
3. Übergeben Sie Ihre Pipeline-Konfiguration auf Git
4. Connect dein Git-Repository mit deinem CI/CD System

> **Note**
>
> Das folgende Verfahren verwendet zwei AWS SAMCLI Befehle, `sam pipeline bootstrap` und`sam pipeline init`. Der Grund dafür, dass es zwei Befehle gibt, ist der Anwendungsfall, in dem Administratoren (d. h. Benutzer, die eine Berechtigung zum Einrichten von AWS Infrastrukturressourcen wie IAM-Benutzern und -Rollen benötigen) mehr Rechte haben als Entwickler (d. h. Benutzer, die lediglich die Erlaubnis benötigen, einzelne Pipelines einzurichten, aber nicht die erforderlichen AWS Infrastrukturressourcen).

Schritt 1: Infrastrukturressourcen erstellen

Verwendete Pipelines AWS SAM benötigen bestimmte AWS Ressourcen, z. B. einen IAM-Benutzer und Rollen mit den erforderlichen Berechtigungen, einen Amazon S3 S3-Bucket und optional ein Amazon ECR-Repository. Sie benötigen für jede Bereitstellungsphase der Pipeline eine Reihe von Infrastrukturressourcen.

Sie können den folgenden Befehl ausführen, um bei dieser Einrichtung zu helfen:

```
sam pipeline bootstrap
```

> **Note**
>
> Führen Sie den vorherigen Befehl für jede Bereitstellungsphase Ihrer Pipeline aus.

Schritt 2: Generieren Sie die Pipeline-Konfiguration

Führen Sie den folgenden Befehl aus, um die Pipeline-Konfiguration zu generieren:

```
sam pipeline init
```

Schritt 3: Übernehmen Sie Ihre Pipeline-Konfiguration in das Git-Repository

Dieser Schritt ist notwendig, um sicherzustellen, dass Ihr CI/CD System Ihre Pipeline-Konfiguration kennt. Er wird ausgeführt, wenn die Änderungen übernommen wurden.

Schritt 4: Connect dein Git-Repository mit deinem CI/CD System

Denn AWS CodePipeline Sie können jetzt die Verbindung herstellen, indem Sie den folgenden Befehl ausführen:

```
sam deploy -t codepipeline.yaml --stack-name <pipeline-stack-name> --
capabilities=CAPABILITY_IAM --region <region-X>
```

Wenn du Bitbucket GitHub verwendest, stelle nach dem vorherigen Ausführen des sam deploy Befehls die Verbindung her, indem du die Schritte unter So stellst du eine Verbindung her, die du im Thema Eine ausstehende Verbindung aktualisieren im Benutzerhandbuch für die Developer Tools-Konsole findest. Speichere außerdem eine Kopie der `CodeStarConnectionArn` aus der Ausgabe des sam deploy Befehls, da du sie benötigst, wenn du sie AWS CodePipeline mit einem anderen Zweig als `main` verwenden möchtest.

Konfiguration anderer Zweige

AWS CodePipeline Verwendet standardmäßig den `main` Zweig mit AWS SAM. Wenn Sie einen anderen Zweig als verwenden möchten`main`, müssen Sie den sam deploy Befehl erneut ausführen. Beachten Sie, dass Sie je nachdem, welches Git-Repository Sie verwenden, möglicherweise auch Folgendes angeben müssen`CodeStarConnectionArn`:

```
# For GitHub and Bitbucket
sam deploy -t codepipeline.yaml --stack-name <feature-pipeline-stack-name> --
capabilities=CAPABILITY_IAM --parameter-overrides="FeatureGitBranch=<branch-name>
 CodeStarConnectionArn=<codestar-connection-arn>"

# For AWS CodeCommit
sam deploy -t codepipeline.yaml --stack-name <feature-pipeline-stack-name> --
capabilities=CAPABILITY_IAM --parameter-overrides="FeatureGitBranch=<branch-name>"
```

Weitere Informationen

Ein praktisches Beispiel für die Einrichtung einer CI/CD Pipeline finden Sie unter CI/CD mit AWS CodePipeline in The Complete Workshop. AWS SAM

Wird verwendet AWS SAM , um Starter-Pipelines für Jenkins, GitLab CI/CD, Actions und Bitbucket-Pipelines zu generieren GitHub

Um eine Starter-Pipeline-Konfiguration für Jenkins-, GitLab CI/CD-, GitHub Actions- oder Bitbucket-Pipelines zu generieren, führe die folgenden Aufgaben in dieser Reihenfolge aus:

1. Infrastrukturressourcen erstellen
2. Connect dein Git-Repository mit deinem CI/CD System
3. Erstellen Sie Objekte mit Anmeldeinformationen
4. Generieren Sie die Pipeline-Konfiguration
5. Übergeben Sie Ihre Pipeline-Konfiguration in das Git-Repository

> **Note**
>
> Das folgende Verfahren verwendet zwei AWS SAMCLI Befehle, `sam pipeline bootstrap` und`sam pipeline init`. Der Grund dafür, dass es zwei Befehle gibt, ist der Anwendungsfall, in dem Administratoren (d. h. Benutzer, die eine Berechtigung zum Einrichten von AWS Infrastrukturressourcen wie IAM-Benutzern und -Rollen benötigen) mehr Rechte haben als Entwickler (d. h. Benutzer, die lediglich die Erlaubnis benötigen, einzelne Pipelines einzurichten, aber nicht die erforderlichen AWS Infrastrukturressourcen).

Schritt 1: Infrastrukturressourcen erstellen

Verwendete Pipelines AWS SAM benötigen bestimmte AWS Ressourcen, z. B. einen IAM-Benutzer und Rollen mit den erforderlichen Berechtigungen, einen Amazon S3 S3-Bucket und optional ein Amazon ECR-Repository. Sie benötigen für jede Bereitstellungsphase der Pipeline eine Reihe von Infrastrukturressourcen.

Sie können den folgenden Befehl ausführen, um bei dieser Einrichtung zu helfen:

```
sam pipeline bootstrap
```

> **Note**
>
> Führen Sie den vorherigen Befehl für jede Bereitstellungsphase Ihrer Pipeline aus.

Sie müssen die AWS Anmeldeinformationen (Schlüssel-ID und geheimer Schlüssel) für die Pipeline-Benutzer für jede Bereitstellungsphase Ihrer Pipeline erfassen, da sie für nachfolgende Schritte benötigt werden.

Schritt 2: Connect dein Git-Repository mit deinem CI/CD System

Das Verbinden Ihres Git-Repositorys mit Ihrem CI/CD System ist notwendig, damit das CI/CD System auf den Quellcode Ihrer Anwendung für Builds und Deployments zugreifen kann.

> **Note**
>
> Sie können diesen Schritt überspringen, wenn Sie eine der folgenden Kombinationen verwenden, da die Verbindung automatisch für Sie hergestellt wird:
>
> 1. GitHub Aktionen mit dem GitHub Repository
> 2. GitLab CI/CD mit Repository GitLab
> 3. Bitbucket-Pipelines mit einem Bitbucket-Repository

Um dein Git-Repository mit deinem CI/CD System zu verbinden, führe einen der folgenden Schritte aus:

- Wenn Sie Jenkins verwenden, lesen Sie in der Jenkins-Dokumentation unter „Hinzufügen einer Branch-Quelle“ nach.
- Wenn du GitLab CI/CD und ein anderes Git-Repository als verwendest GitLab, lies die GitLabDokumentation unter „Ein externes Repository verbinden“.

Schritt 3: Objekte mit Anmeldeinformationen erstellen

Jedes CI/CD System hat seine eigene Methode zur Verwaltung der Anmeldeinformationen, die das CI/CD System für den Zugriff auf Ihr Git-Repository benötigt.

Gehen Sie wie folgt vor, um die erforderlichen Objekte mit Anmeldeinformationen zu erstellen:

- Wenn Sie Jenkins verwenden, erstellen Sie ein einzelnes „Credential“, das sowohl die Schlüssel-ID als auch den geheimen Schlüssel speichert. Folgen Sie den Anweisungen im AWS SAM Blog „Eine Jenkins-Pipeline mit erstellen“ im Abschnitt „Jenkins konfigurieren“. Für den nächsten Schritt benötigen Sie die „Credential ID“.
- Wenn Sie GitLab CI/CD verwenden, erstellen Sie zwei „geschützte Variablen“, jeweils eine für die Schlüssel-ID und den geheimen Schlüssel. Folgen Sie den Anweisungen in der GitLab Dokumentation — für den nächsten Schritt benötigen Sie zwei „variable Schlüssel“.

- Wenn Sie GitHub Aktionen verwenden, erstellen Sie zwei „verschlüsselte Geheimnisse“, jeweils einen für Schlüssel und geheimen Schlüssel. Folgen Sie den Anweisungen in der GitHubDokumentation. Für den nächsten Schritt benötigen Sie zwei „geheime Namen“.
- Wenn du Bitbucket-Pipelines verwendest, erstelle zwei „sichere Variablen“, jeweils eine für die Schlüssel-ID und den geheimen Schlüssel. Folge den Anweisungen in den Abschnitten Variablen und Geheimnisse — für den nächsten Schritt benötigst du zwei „geheime Namen“.

Schritt 4: Generieren Sie die Pipeline-Konfiguration

Führen Sie den folgenden Befehl aus, um die Pipeline-Konfiguration zu generieren. Sie müssen das Anmeldeinformationsobjekt eingeben, das Sie im vorherigen Schritt erstellt haben:

```
sam pipeline init
```

Schritt 5: Übernehmen Sie Ihre Pipeline-Konfiguration in das Git-Repository

Dieser Schritt ist notwendig, um sicherzustellen, dass Ihr CI/CD System Ihre Pipeline-Konfiguration kennt. Er wird ausgeführt, wenn die Änderungen übernommen wurden.

Weitere Informationen

Ein praktisches Beispiel für die Einrichtung einer CI/CD Pipeline mit GitHub Actions finden Sie unter CI/CD with GitHub in The Complete Workshop. AWS SAM

So passen Sie Starter-Pipelines an mit AWS SAM

Als CI/CD Administrator möchten Sie möglicherweise eine Vorlage für die Starter-Pipeline und die zugehörigen geführten Eingabeaufforderungen anpassen, sodass Entwickler in Ihrer Organisation Pipeline-Konfigurationen erstellen können.

The AWS SAMCLI verwendet Cookiecutter-Vorlagen bei der Erstellung von Starter-Vorlagen. Einzelheiten zu Ausstechvorlagen finden Sie unter Cookiecutter.

Mit dem Befehl können Sie auch die Eingabeaufforderungen anpassen, die Benutzern beim Erstellen von Pipeline-Konfigurationen AWS SAMCLI angezeigt werden. `sam pipeline init` Gehen Sie wie folgt vor, um Benutzeraufforderungen anzupassen:

1. **`questions.json`**Datei erstellen — Die **`questions.json`** Datei muss sich im Stammverzeichnis des Projekt-Repositorys befinden. Dies ist das gleiche Verzeichnis wie die Datei.

`cookiecutter.json` Das Schema für die `questions.json` Datei finden Sie unter questions.json.schema. Eine Beispieldatei finden Sie unter questions.json. `questions.json`

2. Zuordnen von Frageschlüsseln zu Cookiecutter-Namen — Jedes Objekt in der `questions.json` Datei benötigt einen Schlüssel, der einem Namen in der Cookiecutter-Vorlage entspricht. Dieser Schlüsselabgleich ist die Art und Weise, wie der AWS SAMCLI Maps-Benutzer Antworten auf die Cookie-Cutter-Vorlage eingibt. Beispiele für diese Tastenkombination finden Sie im Beispieldateien Abschnitt weiter unten in diesem Thema.
3. **`metadata.json`**Datei erstellen — Geben Sie die Anzahl der Stufen an, die die Pipeline in der `metadata.json` Datei enthalten soll. Die Anzahl der Stufen gibt dem `sam pipeline init` Befehl an, für wie viele Stufen Informationen angefordert werden sollen, oder, im Fall der `--bootstrap` Option, für wie viele Stufen Infrastrukturressourcen erstellt werden sollen. Eine `metadata.json` Beispieldatei, die eine Pipeline mit zwei Stufen deklariert, finden Sie unter metadata.json.

Beispielprojekte

Hier sind Beispielprojekte, die jeweils eine Cookiecutter-Vorlage, eine `questions.json` Datei und eine `metadata.json` Datei enthalten:

- Jenkins-Beispiel: Zweistufige Jenkins-Pipeline-Vorlage
- CodePipeline Beispiel: Zweistufige Pipeline-Vorlage CodePipeline

Beispieldateien

Die folgenden Dateien zeigen, wie Fragen in der `questions.json` Datei mit Einträgen in der Cookiecutter-Vorlagendatei verknüpft sind. Beachten Sie, dass es sich bei diesen Beispielen um Dateifragmente und nicht um vollständige Dateien handelt. Beispiele für vollständige Dateien finden Sie im Beispielprojekte Abschnitt weiter oben in diesem Thema.

Beispiel **`questions.json`**:

```
{
  "questions": [{
    "key": "intro",
    "question": "\nThis template configures a pipeline that deploys a serverless
 application to a testing and a production stage.\n",
    "kind": "info"
```

```
  }, {
    "key": "pipeline_user_jenkins_credential_id",
    "question": "What is the Jenkins credential ID (via Jenkins plugin \"aws-
credentials\") for pipeline user access key?",
    "isRequired": true
  }, {
    "key": "sam_template",
    "question": "What is the template file path?",
    "default": "template.yaml"
  }, {
    ...
```

Beispiel **`cookiecutter.json`**:

```
{
  "outputDir": "aws-sam-pipeline",
  "pipeline_user_jenkins_credential_id": "",
  "sam_template": "",
    ...
```

Beispiel **`Jenkinsfile`**:

```
pipeline {
  agent any
  environment {
    PIPELINE_USER_CREDENTIAL_ID =
 '{{cookiecutter.pipeline_user_jenkins_credential_id}}'
    SAM_TEMPLATE = '{{cookiecutter.sam_template}}'
    ...
```

Automatisieren Sie die Bereitstellung Ihrer AWS SAM Anwendung

Wie Sie die Bereitstellung Ihrer AWS SAM Anwendung automatisieren, hängt vom verwendeten CI/CD System ab. AWS SAM Aus diesem Grund zeigen Ihnen die Beispiele in diesem Abschnitt, wie Sie verschiedene CI/CD Systeme konfigurieren, um die Erstellung serverloser Anwendungen in einem AWS SAM Build-Container-Image zu automatisieren. Diese Build-Container-Images erleichtern das Erstellen und Verpacken serverloser Anwendungen mithilfe von. AWS SAMCLI

Die Verfahren für Ihre bestehende CI/CD Pipeline zur Bereitstellung serverloser Anwendungen unterscheiden AWS SAM sich geringfügig, je nachdem, welches CI/CD System Sie verwenden.

Die folgenden Themen enthalten Beispiele für die Konfiguration Ihres CI/CD Systems für die Erstellung serverloser Anwendungen in einem AWS SAM Build-Container-Image:

Themen

- Wird AWS CodePipeline zur Bereitstellung mit verwendet AWS SAM
- Verwenden von Bitbucket-Pipelines für die Bereitstellung mit AWS SAM
- Verwenden von Jenkins für die Bereitstellung mit AWS SAM
- Verwendung von GitLab CI/CD für die Bereitstellung mit AWS SAM
- Verwenden von GitHub Aktionen zur Bereitstellung mit AWS SAM

Wird AWS CodePipeline zur Bereitstellung mit verwendet AWS SAM

Um Ihre AWS CodePipelinePipeline so zu konfigurieren, dass die Erstellung und Bereitstellung Ihrer AWS SAM Anwendung automatisiert wird, müssen Ihre CloudFormation Vorlage und `buildspec.yml` Datei Zeilen enthalten, die Folgendes bewirken:

1. Verweisen Sie aus den verfügbaren Images auf ein Build-Container-Image mit der erforderlichen Laufzeit. Im folgenden Beispiel wird das `public.ecr.aws/sam/build-nodejs20.x` Build-Container-Image verwendet.
2. Konfigurieren Sie die Pipeline-Stufen, um die erforderlichen AWS SAM Befehlszeilenschnittstellenbefehle (CLI) auszuführen. Im folgenden Beispiel AWS SAMCLI werden zwei Befehle ausgeführt: sam build und sam deploy (mit den erforderlichen Optionen).

In diesem Beispiel wird vorausgesetzt, dass Sie alle Funktionen und Ebenen in Ihrer AWS SAM Vorlagendatei mit deklariert haben`runtime: nodejs20.x`.

CloudFormation Vorlagenausschnitt:

```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
    Environment:
      ComputeType: BUILD_GENERAL1_SMALL
      Image: public.ecr.aws/sam/build-nodejs20.x
      Type: LINUX_CONTAINER
    ...
```

`buildspec.yml`Schnipsel:

```
version: 0.2
phases:
  build:
    commands:
      - sam build
      - sam deploy --no-confirm-changeset --no-fail-on-empty-changeset
```

Eine Liste der verfügbaren Build-Container-Images von Amazon Elastic Container Registry (Amazon ECR) für verschiedene Laufzeiten finden Sie unter. Bildrepositorien für AWS SAM

Verwenden von Bitbucket-Pipelines für die Bereitstellung mit AWS SAM

Um deine Bitbucket-Pipeline so zu konfigurieren, dass sie den Build und die Bereitstellung deiner AWS SAM Anwendung automatisiert, muss deine `bitbucket-pipelines.yml` Datei Zeilen enthalten, die Folgendes tun:

1. Verweise auf ein Build-Container-Image mit der erforderlichen Laufzeit aus den verfügbaren Images. Im folgenden Beispiel wird das `public.ecr.aws/sam/build-nodejs20.x` Build-Container-Image verwendet.
2. Konfigurieren Sie die Pipeline-Stufen, um die erforderlichen AWS SAM Befehlszeilenschnittstellenbefehle (CLI) auszuführen. Im folgenden Beispiel AWS SAMCLI werden zwei Befehle ausgeführt: sam build und sam deploy (mit den erforderlichen Optionen).

In diesem Beispiel wird vorausgesetzt, dass Sie alle Funktionen und Ebenen in Ihrer AWS SAM Vorlagendatei mit deklariert haben`runtime: nodejs20.x`.

```
image: public.ecr.aws/sam/build-nodejs20.x

pipelines:
  branches:
    main: # branch name
      - step:
          name: Build and Package
          script:
            - sam build
            - sam deploy --no-confirm-changeset --no-fail-on-empty-changeset
```

Eine Liste der verfügbaren Build-Container-Images von Amazon Elastic Container Registry (Amazon ECR) für verschiedene Laufzeiten finden Sie unter. Bildrepositorien für AWS SAM

Verwenden von Jenkins für die Bereitstellung mit AWS SAM

Um Ihre Jenkins-Pipeline so zu konfigurieren, dass sie den Build und die Bereitstellung Ihrer AWS SAM Anwendung automatisiert, `Jenkinsfile` müssen Sie Zeilen enthalten, die Folgendes tun:

1. Verweisen Sie aus den verfügbaren Images auf ein Build-Container-Image mit der erforderlichen Laufzeit. Im folgenden Beispiel wird das `public.ecr.aws/sam/build-nodejs20.x` Build-Container-Image verwendet.
2. Konfigurieren Sie die Pipeline-Stufen, um die erforderlichen AWS SAM Befehlszeilenschnittstellenbefehle (CLI) auszuführen. Im folgenden Beispiel AWS SAMCLI werden zwei Befehle ausgeführt: sam build und sam deploy (mit den erforderlichen Optionen).

In diesem Beispiel wird vorausgesetzt, dass Sie alle Funktionen und Ebenen in Ihrer AWS SAM Vorlagendatei mit deklariert haben`runtime: nodejs20.x`.

```
pipeline {
    agent { docker { image 'public.ecr.aws/sam/build-nodejs20.x' } }
    stages {
        stage('build') {
            steps {
                sh 'sam build'
                sh 'sam deploy --no-confirm-changeset --no-fail-on-empty-changeset'
            }
        }
    }
}
```

Eine Liste der verfügbaren Build-Container-Images von Amazon Elastic Container Registry (Amazon ECR) für verschiedene Laufzeiten finden Sie unter. Bildrepositorien für AWS SAM

Verwendung von GitLab CI/CD für die Bereitstellung mit AWS SAM

Um Ihre GitLabPipeline so zu konfigurieren, dass der Build und die Bereitstellung Ihrer AWS SAM Anwendung automatisiert werden, muss Ihre `gitlab-ci.yml` Datei Zeilen enthalten, die Folgendes tun:

1. Verweisen Sie aus den verfügbaren Images auf ein Build-Container-Image mit der erforderlichen Laufzeit. Im folgenden Beispiel wird das `public.ecr.aws/sam/build-nodejs20.x` Build-Container-Image verwendet.
2. Konfigurieren Sie die Pipeline-Stufen, um die erforderlichen AWS SAM Befehlszeilenschnittstellenbefehle (CLI) auszuführen. Im folgenden Beispiel AWS SAMCLI werden zwei Befehle ausgeführt: sam build und sam deploy (mit den erforderlichen Optionen).

In diesem Beispiel wird vorausgesetzt, dass Sie alle Funktionen und Ebenen in Ihrer AWS SAM Vorlagendatei mit deklariert haben`runtime: nodejs20.x`.

```
image: public.ecr.aws/sam/build-nodejs20.x
deploy:
  script:
    - sam build
    - sam deploy --no-confirm-changeset --no-fail-on-empty-changeset
```

Eine Liste der verfügbaren Build-Container-Images von Amazon Elastic Container Registry (Amazon ECR) für verschiedene Laufzeiten finden Sie unter. Bildrepositorien für AWS SAM

Verwenden von GitHub Aktionen zur Bereitstellung mit AWS SAM

Um Ihre GitHubPipeline so zu konfigurieren, dass der Build und die Bereitstellung Ihrer AWS SAM Anwendung automatisiert werden, müssen Sie zunächst die AWS SAM Befehlszeilenschnittstelle (CLI) auf Ihrem Host installieren. Sie können GitHub Aktionen in Ihrem GitHub Workflow verwenden, um Ihnen bei dieser Einrichtung zu helfen.

Der folgende GitHub Beispiel-Workflow richtet mithilfe einer Reihe von GitHub Aktionen einen Ubuntu-Host ein und führt dann AWS SAMCLI Befehle zum Erstellen und Bereitstellen einer AWS SAM Anwendung aus:

```
on:
  push:
    branches:
      - main
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-python@v3
```

```
      - uses: aws-actions/setup-sam@v2
      - uses: aws-actions/configure-aws-credentials@v1
        with:
          aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
          aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
          aws-region: us-east-2
      - run: sam build --use-container
      - run: sam deploy --no-confirm-changeset --no-fail-on-empty-changeset
```

Eine Liste der verfügbaren Build-Container-Images von Amazon Elastic Container Registry (Amazon ECR) für verschiedene Laufzeiten finden Sie unter. Bildrepositorien für AWS SAM

So verwenden Sie die OIDC-Authentifizierung mit Pipelines AWS SAM

AWS Serverless Application Model (AWS SAM) unterstützt OpenID Connect (OIDC) - Benutzerauthentifizierung für Bitbucket, GitHub Actions sowie GitLab Continuous Integration und Continuous Delivery (CI/CD) platforms. With this support, you can use authorized CI/CDBenutzerkonten von jeder dieser Plattformen) zur Verwaltung deiner serverlosen Anwendungspipelines. Andernfalls müsstest du mehrere AWS Identity and Access Management (IAM-) Benutzer erstellen und verwalten, um den Zugriff auf Pipelines zu kontrollieren. AWS SAM

Richten Sie OIDC mit Pipeline ein AWS SAM

Gehen Sie während des `sam pipeline bootstrap` Konfigurationsprozesses wie folgt vor, um OIDC mit Ihrer Pipeline einzurichten. AWS SAM

1. Wenn Sie aufgefordert werden, einen Identitätsanbieter auszuwählen, wählen Sie OIDC aus.
2. Wählen Sie als Nächstes einen unterstützten OIDC-Anbieter aus.
3. Geben Sie die URL des OIDC-Anbieters ein, beginnend mit. **`https://`**

> **Note**
>
> AWS SAM verweist bei der Generierung des `AWS::IAM::OIDCProvider` Ressourcentyps auf diese URL.

4. Folgen Sie anschließend den Anweisungen und geben Sie die CI/CD Plattforminformationen ein, die für den Zugriff auf die ausgewählte Plattform erforderlich sind. Diese Details variieren je nach Plattform und können Folgendes beinhalten:
   - OIDC-Client-ID.

- Name des Code-Repositorys oder UUID (Universally Unique Identifier).
- Gruppen- oder Organisationsname, der dem Repository zugeordnet ist.
- GitHub Organisation, zu der das Code-Repository gehört.
- GitHub Name des Repositorys.
- Zweig, von dem aus die Bereitstellungen erfolgen.

5. AWS SAM zeigt eine Zusammenfassung der eingegebenen OIDC-Konfiguration an. Geben Sie die Nummer für eine Einstellung ein, um sie zu bearbeiten, oder drücken Sie, Enter um fortzufahren.
6. Wenn Sie aufgefordert werden, die Erstellung der Ressourcen zu bestätigen, die zur Unterstützung der eingegebenen OIDC-Verbindung erforderlich sind, drücken Sie, Y um fortzufahren.

AWS SAM generiert eine `AWS::IAM::OIDCProvider` AWS CloudFormation Ressource mit der angegebenen Konfiguration, die die Rolle der Pipeline-Ausführung übernimmt. Weitere Informationen zu diesem CloudFormation Ressourcentyp finden Sie unter AWS: :IAM:: OIDCProvider im AWS CloudFormation Benutzerhandbuch.

> **Note**
>
> Wenn die Identitätsanbieter-Ressource (IdP) bereits in Ihrer vorhanden ist AWS-Konto, AWS SAM verweist Sie darauf, anstatt eine neue Ressource zu erstellen.

Beispiel

Im Folgenden finden Sie ein Beispiel für die Einrichtung von OIDC mit Pipeline. AWS SAM

```
Select a permissions provider:
    1 - IAM (default)
    2 - OpenID Connect (OIDC)
Choice (1, 2): 2
Select an OIDC provider:
    1 - GitHub Actions
    2 - GitLab
    3 - Bitbucket
Choice (1, 2, 3): 1
Enter the URL of the OIDC provider [https://token.actions.githubusercontent.com]:
Enter the OIDC client ID (sometimes called audience) [sts.amazonaws.com]:
```

```
Enter the GitHub organization that the code repository belongs to. If there is no
 organization enter your username instead: my-org
Enter GitHub repository name: testing
Enter the name of the branch that deployments will occur from [main]:

[3] Reference application build resources
Enter the pipeline execution role ARN if you have previously created one, or we will
 create one for you []:
Enter the CloudFormation execution role ARN if you have previously created one, or we
 will create one for you []:
Please enter the artifact bucket ARN for your Lambda function. If you do not have a
 bucket, we will create one for you []:
Does your application contain any IMAGE type Lambda functions? [y/N]:

[4] Summary
Below is the summary of the answers:
    1 - Account: 123456
    2 - Stage configuration name: dev
    3 - Region: us-east-1
    4 - OIDC identity provider URL: https://token.actions.githubusercontent.com
    5 - OIDC client ID: sts.amazonaws.com
    6 - GitHub organization: my-org
    7 - GitHub repository: testing
    8 - Deployment branch: main
    9 - Pipeline execution role: [to be created]
    10 - CloudFormation execution role: [to be created]
    11 - Artifacts bucket: [to be created]
    12 - ECR image repository: [skipped]
Press enter to confirm the values above, or select an item to edit the value:

This will create the following required resources for the 'dev' configuration:
    - IAM OIDC Identity Provider
    - Pipeline execution role
    - CloudFormation execution role
    - Artifact bucket
Should we proceed with the creation? [y/N]:
```

Weitere Informationen

Weitere Informationen zur Verwendung von OIDC mit AWS SAM Pipeline finden Sie unter. [sam pipeline bootstrap]

Einführung in die Verwendung von sam sync to sync to AWS Cloud

Der AWS Serverless Application Model Befehl Command Line Interface (AWS SAMCLI) `sam sync` bietet Optionen zum schnellen Synchronisieren von lokalen Anwendungsänderungen mit dem AWS Cloud. Verwenden Sie `sam sync` ihn bei der Entwicklung Ihrer Anwendungen für:

1. Automatische Erkennung und Synchronisation lokaler Änderungen mit dem AWS Cloud.
2. Passen Sie an, welche lokalen Änderungen mit dem AWS Cloud synchronisiert werden.
3. Bereiten Sie Ihre Anwendung in der Cloud für Tests und Validierung vor.

Mit können Sie einen schnellen Entwicklungsworkflow erstellen`sam sync`, der die Zeit verkürzt, die benötigt wird, um Ihre lokalen Änderungen zum Testen und Validieren mit der Cloud zu synchronisieren.

> **Note**
>
> Der `sam sync` Befehl wird für Entwicklungsumgebungen empfohlen. Für Produktionsumgebungen empfehlen wir, eine CI/CD-Pipeline (Continuous Integration and Delivery) zu verwenden `sam deploy` oder zu konfigurieren. Weitere Informationen hierzu finden Sie unter Stellen Sie Ihre Anwendung und Ressourcen bereit mit AWS SAM.

Der `sam sync` Befehl ist Teil von. AWS SAM Accelerate AWS SAM Acceleratebietet Tools, mit denen Sie das Entwickeln und Testen serverloser Anwendungen in der AWS Cloud beschleunigen können.

Topics

- Erkennt automatisch lokale Änderungen und synchronisiert sie mit AWS Cloud
- Passen Sie an, welche lokalen Änderungen mit dem synchronisiert werden AWS Cloud
- Bereiten Sie Ihre Anwendung in der Cloud für Tests und Validierung vor
- Optionen für den Befehl sam sync
- Fehlerbehebung
- Beispiele
- Weitere Informationen

Erkennt automatisch lokale Änderungen und synchronisiert sie mit AWS Cloud

Führen Sie `sam sync` mit der `--watch` Option aus, mit der Synchronisierung Ihrer Anwendung mit dem AWS Cloud zu beginnen. Dies macht Folgendes:

1. Erstellen Sie Ihre Anwendung — Dieser Vorgang ähnelt der Verwendung des `sam build` Befehls.
2. Stellen Sie Ihre Anwendung bereit — AWS SAMCLI Der stellt Ihre Anwendung AWS CloudFormation unter Verwendung Ihrer Standardeinstellungen bereit. Die folgenden Standardwerte werden verwendet:
   a. AWS Anmeldeinformationen und allgemeine Konfigurationseinstellungen befinden sich in Ihrem `.aws` Benutzerordner.
   b. Die Einstellungen für die Anwendungsbereitstellung finden Sie in der `samconfig.toml` Datei Ihrer Anwendung.

   Wenn keine Standardwerte gefunden werden können, werden Sie darüber informiert und der Synchronisierungsvorgang AWS SAMCLI wird beendet.
3. Achten Sie auf lokale Änderungen — AWS SAMCLI Der läuft weiter und sucht nach lokalen Änderungen an Ihrer Anwendung. Das bietet die `--watch` Option.

   Diese Option kann standardmäßig aktiviert sein. Standardwerte finden Sie in der `samconfig.toml` Datei Ihrer Anwendung. Im Folgenden sehen Sie ein Beispiel für eine -Datei:

   ```
   ...
   [default.sync]
   [default.sync.parameters]
   watch = true
   ...
   ```
4. Lokale Änderungen mit dem synchronisieren AWS Cloud — Wenn Sie lokale Änderungen vornehmen, AWS SAMCLI erkennt und synchronisiert der diese Änderungen mit der AWS Cloud schnellsten verfügbaren Methode. Je nach Art der Änderung kann Folgendes passieren:
   a. Wenn Ihre aktualisierte Ressource den AWS Dienst unterstützt APIs, verwendet AWS SAMCLI sie ihn, um Ihre Änderungen bereitzustellen. Dies führt zu einer schnellen Synchronisierung, um Ihre Ressource in der zu aktualisieren AWS Cloud.
   b. Wenn Ihre aktualisierte Ressource den AWS Dienst nicht unterstützt APIs, AWS SAMCLI führt sie eine CloudFormation Bereitstellung durch. Dadurch wird Ihre gesamte Anwendung

in der aktualisiert AWS Cloud. Es ist zwar nicht so schnell, verhindert aber, dass Sie eine Bereitstellung manuell initiieren müssen.

Da der `sam sync` Befehl Ihre Anwendung in der automatisch aktualisiert AWS Cloud, wird er nur für Entwicklungsumgebungen empfohlen. Beim Ausführen werden Sie aufgefordert`sam sync`, Folgendes zu bestätigen:

```
**The sync command should only be used against a development stack**.

Confirm that you are synchronizing a development stack.

Enter Y to proceed with the command, or enter N to cancel:
 [Y/n]: ENTER
```

Passen Sie an, welche lokalen Änderungen mit dem synchronisiert werden AWS Cloud

Stellen Sie Optionen bereit, um anzupassen, welche lokalen Änderungen mit dem AWS Cloud synchronisiert werden. Dies kann die Zeit verkürzen, die benötigt wird, um Ihre lokalen Änderungen zum Testen und Validieren in der Cloud zu sehen.

Bieten Sie beispielsweise die `--code` Option an, nur Codeänderungen wie AWS Lambda Funktionscode zu synchronisieren. Wenn Sie sich während der Entwicklung speziell auf Lambda-Code konzentrieren, werden Ihre Änderungen schnell in die Cloud übertragen, um sie zu testen und zu validieren. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam sync --code --watch
```

Verwenden Sie die `--resource-id` Option, um nur Codeänderungen für eine bestimmte Lambda-Funktion oder -Layer zu synchronisieren. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam sync --code --resource-id HelloWorldFunction --resource-id HelloWorldLayer
```

Bereiten Sie Ihre Anwendung in der Cloud für Tests und Validierung vor

Der `sam sync` Befehl findet automatisch die schnellste verfügbare Methode zum Aktualisieren Ihrer Anwendung in der AWS Cloud. Dies kann Ihre Entwicklungs- und Cloud-Test-Workflows beschleunigen. Durch die Nutzung des AWS Service APIs können Sie unterstützte Ressourcen

schnell entwickeln, synchronisieren und testen. Ein praktisches Beispiel finden Sie in Modul 6 — AWS SAM Accelerate in The Complete AWS SAM Workshop.

Optionen für den Befehl sam sync

Im Folgenden sind einige der wichtigsten Optionen aufgeführt, mit denen Sie den `sam sync` Befehl ändern können. Eine Liste aller Optionen finden Sie unterSam Sync.

Führen Sie eine einmalige CloudFormation Bereitstellung durch

Verwenden Sie die `--no-watch` Option, um die automatische Synchronisierung zu deaktivieren. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam sync --no-watch
```

Der AWS SAMCLI führt eine einmalige CloudFormation Bereitstellung durch. Dieser Befehl gruppiert die Aktionen, die mit den `sam deploy` Befehlen `sam build` und ausgeführt werden.

Überspringen Sie die erste CloudFormation Bereitstellung

Sie können anpassen, ob bei jeder Ausführung eine CloudFormation Bereitstellung erforderlich `sam sync` ist.

- Geben Sie `--no-skip-deploy-sync` an, dass bei jeder Ausführung eine CloudFormation Bereitstellung erforderlich `sam sync` ist. Dadurch wird sichergestellt, dass Ihre lokale Infrastruktur synchronisiert ist CloudFormation, sodass Abweichungen vermieden werden. Die Verwendung dieser Option verlängert Ihren Entwicklungs- und Test-Workflow um zusätzliche Zeit.
- Geben Sie `--skip-deploy-sync` an, dass die CloudFormation Bereitstellung optional ist. Das vergleicht AWS SAMCLI Ihre lokale AWS SAM Vorlage mit Ihrer bereitgestellten CloudFormation Vorlage und überspringt die erste CloudFormation Bereitstellung, wenn keine Änderung festgestellt wird. Wenn Sie die CloudFormation Bereitstellung überspringen, können Sie Zeit sparen, wenn Sie lokale Änderungen mit dem synchronisieren. AWS Cloud

  Wenn keine Änderung festgestellt AWS SAMCLI wird, führt der in den folgenden Szenarien dennoch eine CloudFormation Bereitstellung durch:

  - Wenn seit Ihrer letzten CloudFormation Bereitstellung 7 Tage oder mehr vergangen sind.
  - Wenn eine große Anzahl von Änderungen am Lambda-Funktionscode erkannt wird, ist die CloudFormation Bereitstellung die schnellste Methode, um Ihre Anwendung zu aktualisieren.

Im Folgenden wird ein Beispiel gezeigt:

```
$ sam sync --skip-deploy-sync
```

Synchronisieren Sie eine Ressource aus einem verschachtelten Stack

Um eine Ressource aus einem verschachtelten Stapel zu synchronisieren

1. Stellen Sie den Root-Stack bereit mit`--stack-name`.
2. Identifizieren Sie die Ressource im verschachtelten Stapel im folgenden Format:*`nestedStackId/resourceId`*.
3. Stellen Sie die Ressource im verschachtelten Stapel bereit mit. `--resource-id`

   Im Folgenden wird ein Beispiel gezeigt:

   ```
   $ sam sync --code --stack-name sam-app --resource-id myNestedStack/
   HelloWorldFunction
   ```

Weitere Informationen zum Erstellen verschachtelter Anwendungen finden Sie unter. Verwenden Sie Code und Ressourcen mithilfe verschachtelter Anwendungen in AWS SAM

Geben Sie einen bestimmten CloudFormation Stack an, der aktualisiert werden soll

Um einen bestimmten CloudFormation Stack für die Aktualisierung anzugeben, geben Sie die `--stack-name` Option an. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam sync --stack-name dev-sam-app
```

Beschleunigen Sie die Build-Zeiten, indem Sie Ihr Projekt im Quellordner erstellen

Für unterstützte Laufzeiten und Build-Methoden können Sie die `--build-in-source` Option verwenden, um Ihr Projekt direkt im Quellordner zu erstellen. Standardmäßig werden die AWS SAM CLI Builds in einem temporären Verzeichnis gespeichert, was das Kopieren von Quellcode und Projektdateien beinhaltet. Damit befinden `--build-in-source` sich die AWS SAM CLI Builds direkt in Ihrem Quellordner, was den Build-Prozess beschleunigt, da keine Dateien mehr in ein temporäres Verzeichnis kopiert werden müssen.

Eine Liste der unterstützten Laufzeiten und Build-Methoden finden Sie unter `--build-in-source`.

Geben Sie Dateien und Ordner an, die keine Synchronisierung initiieren

Verwenden Sie die `--watch-exclude` Option, um eine Datei oder einen Ordner anzugeben, die bei einer Aktualisierung keine Synchronisierung initiieren. Weitere Informationen zu dieser Option finden Sie unter `--watch-exclude`.

Im Folgenden finden Sie ein Beispiel, das die mit unserer `HelloWorldFunction` Funktion verknüpfte `package-lock.json` Datei ausschließt:

```
$ sam sync --watch --watch-exclude HelloWorldFunction=package-lock.json
```

Wenn dieser Befehl ausgeführt wird, AWS SAM CLI wird der Synchronisierungsvorgang initiiert. Diese umfasst die folgenden Funktionen:

- Führen Sie `sam build` das Programm aus, um Ihre Funktionen zu erstellen und Ihre Anwendung für die Bereitstellung vorzubereiten.
- Führen Sie `sam deploy` den Befehl aus, um Ihre Anwendung bereitzustellen.
- Achten Sie auf Änderungen an Ihrer Anwendung.

Wenn wir die `package-lock.json` Datei ändern, AWS SAM CLI wird keine Synchronisierung initiiert. Wenn eine andere Datei aktualisiert wird, AWS SAM CLI wird eine Synchronisierung initiiert, die die `package-lock.json` Datei einschließt.

Das Folgende ist ein Beispiel für die Angabe einer Lambda-Funktion eines untergeordneten Stacks:

```
$ sam sync --watch --watch-exclude ChildStackA/MyFunction=database.sqlite3
```

Fehlerbehebung

Informationen zur Fehlerbehebung bei AWS SAMCLI finden Sie unterAWS SAMCLIProblembehandlung.

Beispiele

Verwenden von Sam Sync zum Aktualisieren der Hello World-Anwendung

In diesem Beispiel beginnen wir mit der Initialisierung der Hello World-Beispielanwendung. Weitere Informationen zu dieser Anwendung finden Sie unterTutorial: Stellen Sie eine Hello World-Anwendung bereit mit AWS SAM.

Mit der Ausführung `sam sync` wird der Build- und Bereitstellungsprozess gestartet.

```
$ sam sync

The SAM CLI will use the AWS Lambda, Amazon API Gateway, and AWS StepFunctions APIs to
 upload your code without
performing a CloudFormation deployment. This will cause drift in your CloudFormation
 stack.
**The sync command should only be used against a development stack**.

Confirm that you are synchronizing a development stack.

Enter Y to proceed with the command, or enter N to cancel:
 [Y/n]:
Queued infra sync. Waiting for in progress code syncs to complete...
Starting infra sync.
Manifest file is changed (new hash: 3298f13049d19cffaa37ca931dd4d421) or dependency
 folder (.aws-sam/deps/0663e6fe-a888-4efb-b908-e2344261e9c7) is missing for
 (HelloWorldFunction), downloading dependencies and copying/building source
Building codeuri: /Users/.../Demo/sync/sam-app/hello_world runtime: python3.9 metadata:
 {} architecture: x86_64 functions: HelloWorldFunction
Running PythonPipBuilder:CleanUp
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource

Build Succeeded

Successfully packaged artifacts and wrote output template to file /var/
folders/45/5ct135bx3fn2551_ptl5g6_80000gr/T/tmpx_5t4u3f.
Execute the following command to deploy the packaged template
sam deploy --template-file /var/folders/45/5ct135bx3fn2551_ptl5g6_80000gr/T/tmpx_5t4u3f
 --stack-name <YOUR STACK NAME>


    Deploying with following values
    ===============================
    Stack name                   : sam-app
    Region                       : us-west-2
    Disable rollback             : False
    Deployment s3 bucket         : aws-sam-cli-managed-default-samclisam-s3-demo-
bucket-1a4x26zbcdkqr
    Capabilities                 : ["CAPABILITY_NAMED_IAM", "CAPABILITY_AUTO_EXPAND"]
    Parameter overrides          : {}
    Signing Profiles             : null
```

```
Initiating deployment
=====================


2023-03-17 11:17:19 - Waiting for stack create/update to complete

CloudFormation events from stack operations (refresh every 0.5 seconds)
---------------------------------------------------------------------------------------------------
ResourceStatus                      ResourceType
 LogicalResourceId                   ResourceStatusReason
---------------------------------------------------------------------------------------------------
CREATE_IN_PROGRESS                  AWS::CloudFormation::Stack          sam-app
                     Transformation succeeded
CREATE_IN_PROGRESS                  AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt   -
                                                                         ack
CREATE_IN_PROGRESS                  AWS::IAM::Role
 HelloWorldFunctionRole              -
CREATE_IN_PROGRESS                  AWS::IAM::Role
 HelloWorldFunctionRole              Resource creation Initiated
CREATE_IN_PROGRESS                  AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt   Resource creation Initiated
                                                                         ack
CREATE_COMPLETE                     AWS::IAM::Role
 HelloWorldFunctionRole              -
CREATE_COMPLETE                     AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt   -
                                                                         ack
CREATE_IN_PROGRESS                  AWS::Lambda::Function
 HelloWorldFunction                  -
CREATE_IN_PROGRESS                  AWS::Lambda::Function
 HelloWorldFunction                  Resource creation Initiated
CREATE_COMPLETE                     AWS::Lambda::Function
 HelloWorldFunction                  -
CREATE_IN_PROGRESS                  AWS::ApiGateway::RestApi
 ServerlessRestApi                   -
CREATE_IN_PROGRESS                  AWS::ApiGateway::RestApi
 ServerlessRestApi                   Resource creation Initiated
CREATE_COMPLETE                     AWS::ApiGateway::RestApi
 ServerlessRestApi                   -
CREATE_IN_PROGRESS                  AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment47fc2d   -
                                                                         5f9d
```

```
CREATE_IN_PROGRESS                  AWS::Lambda::Permission
 HelloWorldFunctionHelloWorldPermi   -
                                                                ssionProd
CREATE_IN_PROGRESS                  AWS::Lambda::Permission
 HelloWorldFunctionHelloWorldPermi   Resource creation Initiated
                                                                ssionProd
CREATE_IN_PROGRESS                  AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment47fc2d   Resource creation Initiated
                                                                5f9d
CREATE_COMPLETE                     AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment47fc2d   -
                                                                5f9d
CREATE_IN_PROGRESS                  AWS::ApiGateway::Stage
 ServerlessRestApiProdStage          -
CREATE_IN_PROGRESS                  AWS::ApiGateway::Stage
 ServerlessRestApiProdStage          Resource creation Initiated
CREATE_COMPLETE                     AWS::ApiGateway::Stage
 ServerlessRestApiProdStage          -
CREATE_COMPLETE                     AWS::Lambda::Permission
 HelloWorldFunctionHelloWorldPermi   -
                                                                ssionProd
CREATE_COMPLETE                     AWS::CloudFormation::Stack          sam-app
                      -
---------------------------------------------------------------------------------------------

CloudFormation outputs from deployed stack
---------------------------------------------------------------------------------------------
Outputs
---------------------------------------------------------------------------------------------
Key                 HelloWorldFunctionIamRole
Description         Implicit IAM Role created for Hello World function
Value               arn:aws:iam::012345678910:role/sam-app-HelloWorldFunctionRole-
BUFVMO2PJIYF

Key                 HelloWorldApi
Description         API Gateway endpoint URL for Prod stage for Hello World function
Value               https://pcrx5gdaof.execute-api.us-west-2.amazonaws.com/Prod/hello/

Key                 HelloWorldFunction
Description         Hello World Lambda Function ARN
Value               arn:aws:lambda:us-west-2:012345678910:function:sam-app-
HelloWorldFunction-2PlN6TPTQoco
---------------------------------------------------------------------------------------------
Stack creation succeeded. Sync infra completed.
```

```
Infra sync completed.
CodeTrigger not created as CodeUri or DefinitionUri is missing for ServerlessRestApi.
```

Sobald die Bereitstellung abgeschlossen ist, ändern wir den `HelloWorldFunction` Code. Der AWS SAMCLI erkennt diese Änderung und synchronisiert unsere Anwendung mit dem AWS Cloud. Da der AWS Service AWS Lambda unterstützt wird APIs, wird eine schnelle Synchronisierung durchgeführt.

```
Syncing Lambda Function HelloWorldFunction...
Manifest is not changed for (HelloWorldFunction), running incremental build
Building codeuri: /Users/.../Demo/sync/sam-app/hello_world runtime: python3.9 metadata:
 {} architecture: x86_64 functions: HelloWorldFunction
Running PythonPipBuilder:CopySource
Finished syncing Lambda Function HelloWorldFunction.
```

Als Nächstes ändern wir unseren API-Endpunkt in der AWS SAM Vorlage der Anwendung. Wir wechseln `/hello` zu`/helloworld`.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  HelloWorldFunction:
    ...
    Properties:
      ...
      Events:
        HelloWorld:
          Type: Api
          Properties:
            Path: /helloworld
            Method: get
```

Da die Amazon API Gateway Gateway-Ressource die AWS Service-API nicht unterstützt, führt sie AWS SAMCLI automatisch eine CloudFormation Bereitstellung durch. Im Folgenden finden Sie ein Beispiel für eine Ausgabe:

```
Queued infra sync. Waiting for in progress code syncs to complete...
Starting infra sync.
Manifest is not changed for (HelloWorldFunction), running incremental build
```

```
Building codeuri: /Users/.../Demo/sync/sam-app/hello_world runtime: python3.9 metadata:
 {} architecture: x86_64 functions: HelloWorldFunction
Running PythonPipBuilder:CopySource

Build Succeeded

Successfully packaged artifacts and wrote output template to file /var/
folders/45/5ct135bx3fn2551_ptl5g6_80000gr/T/tmpuabo0jb9.
Execute the following command to deploy the packaged template
sam deploy --template-file /var/folders/45/5ct135bx3fn2551_ptl5g6_80000gr/T/tmpuabo0jb9
 --stack-name <YOUR STACK NAME>


    Deploying with following values
    ===============================
    Stack name                   : sam-app
    Region                       : us-west-2
    Disable rollback             : False
    Deployment s3 bucket         : aws-sam-cli-managed-default-samclisam-s3-demo-
bucket-1a4x26zbcdkqr
    Capabilities                 : ["CAPABILITY_NAMED_IAM", "CAPABILITY_AUTO_EXPAND"]
    Parameter overrides          : {}
    Signing Profiles             : null

Initiating deployment
=====================


2023-03-17 14:41:18 - Waiting for stack create/update to complete

CloudFormation events from stack operations (refresh every 0.5 seconds)
---------------------------------------------------------------------------------------------
ResourceStatus                       ResourceType
 LogicalResourceId                    ResourceStatusReason
---------------------------------------------------------------------------------------------
UPDATE_IN_PROGRESS                   AWS::CloudFormation::Stack          sam-app
                      Transformation succeeded
UPDATE_IN_PROGRESS                   AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt    -
                                                                          ack
UPDATE_COMPLETE                      AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt    -
                                                                          ack
```

```
UPDATE_IN_PROGRESS                  AWS::ApiGateway::RestApi
 ServerlessRestApi                   -
UPDATE_COMPLETE                     AWS::ApiGateway::RestApi
 ServerlessRestApi                   -
CREATE_IN_PROGRESS                  AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment8cf30e   -                                                  d3cd
UPDATE_IN_PROGRESS                  AWS::Lambda::Permission
 HelloWorldFunctionHelloWorldPermi   Requested update requires the                      ssionProd
                     creation of a new physical

                     resource; hence creating one.
UPDATE_IN_PROGRESS                  AWS::Lambda::Permission
 HelloWorldFunctionHelloWorldPermi   Resource creation Initiated                        ssionProd
CREATE_IN_PROGRESS                  AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment8cf30e   Resource creation Initiated                       d3cd
CREATE_COMPLETE                     AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment8cf30e   -                                                  d3cd
UPDATE_IN_PROGRESS                  AWS::ApiGateway::Stage
 ServerlessRestApiProdStage          -
UPDATE_COMPLETE                     AWS::ApiGateway::Stage
 ServerlessRestApiProdStage          -
UPDATE_COMPLETE                     AWS::Lambda::Permission
 HelloWorldFunctionHelloWorldPermi   -                                                  ssionProd
UPDATE_COMPLETE_CLEANUP_IN_PROGRE   AWS::CloudFormation::Stack          sam-app
                     -
SS
DELETE_IN_PROGRESS                  AWS::Lambda::Permission
 HelloWorldFunctionHelloWorldPermi   -                                                  ssionProd
DELETE_IN_PROGRESS                  AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment47fc2d   -                                                  5f9d
DELETE_COMPLETE                     AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment47fc2d   -                                                  5f9d
UPDATE_COMPLETE                     AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt   -                                                  ack
```

```
DELETE_COMPLETE                     AWS::Lambda::Permission
 HelloWorldFunctionHelloWorldPermi   -
                                                                    ssionProd
UPDATE_COMPLETE                     AWS::CloudFormation::Stack      sam-app
                      -
-------------------------------------------------------------------------------------------------

CloudFormation outputs from deployed stack
-------------------------------------------------------------------------------------------------
Outputs
-------------------------------------------------------------------------------------------------
Key                 HelloWorldFunctionIamRole
Description         Implicit IAM Role created for Hello World function
Value               arn:aws:iam::012345678910:role/sam-app-HelloWorldFunctionRole-
BUFVMO2PJIYF

Key                 HelloWorldApi
Description         API Gateway endpoint URL for Prod stage for Hello World function
Value               https://pcrx5gdaof.execute-api.us-west-2.amazonaws.com/Prod/hello/

Key                 HelloWorldFunction
Description         Hello World Lambda Function ARN
Value               arn:aws:lambda:us-west-2:012345678910:function:sam-app-
HelloWorldFunction-2PlN6TPTQoco
-------------------------------------------------------------------------------------------------


Stack update succeeded. Sync infra completed.

Infra sync completed.
```

Weitere Informationen

Eine Beschreibung aller `sam sync` Optionen finden Sie unterSam Sync.

Überwachen Sie Ihre serverlose Anwendung mit AWS SAM

Nach der Bereitstellung können Sie Ihre serverlose Anwendung überwachen, um Einblicke in ihren Betrieb zu erhalten und Anomalien zu erkennen, was bei der Fehlerbehebung hilfreich sein kann. Dieser Abschnitt enthält Einzelheiten zur Überwachung Ihrer serverlosen Anwendung. Dies beinhaltet Informationen darüber, wie Sie Amazon so konfigurieren können CloudWatch , dass Sie benachrichtigt werden, wenn es Anomalien feststellt. Es enthält auch Informationen zur Arbeit mit Protokollen, einschließlich der Hervorhebung von Fehlern und Tipps zum Anzeigen, Filtern, Abrufen und Nachverfolgen von Protokollen.

Themen

- Verwenden von CloudWatch Application Insights zur Überwachung Ihrer AWS SAM serverlosen Anwendungen
- Mit Logins arbeiten AWS SAM

Verwenden von CloudWatch Application Insights zur Überwachung Ihrer AWS SAM serverlosen Anwendungen

Amazon CloudWatch Application Insights unterstützt Sie bei der Überwachung der AWS Ressourcen in Ihren Anwendungen, um potenzielle Probleme zu identifizieren. Es kann AWS Ressourcendaten auf Anzeichen von Problemen analysieren und automatisierte Dashboards zur Visualisierung dieser Probleme erstellen. Sie können CloudWatch Application Insights für die Verwendung mit Ihren AWS Serverless Application Model (AWS SAM) -Anwendungen konfigurieren. Weitere Informationen zu CloudWatch Application Insights finden Sie unter Amazon CloudWatch Application Insights im CloudWatch Amazon-Benutzerhandbuch.

Themen

- Konfiguration von CloudWatch Application Insights mit AWS SAM
- Nächste Schritte

Konfiguration von CloudWatch Application Insights mit AWS SAM

Konfigurieren Sie CloudWatch Application Insights für Ihre AWS SAM Anwendungen über die AWS SAM Befehlszeilenschnittstelle (AWS SAMCLI) oder über Ihre AWS SAM Vorlagen.

Konfigurieren Sie über AWS SAMCLI

Wenn Sie Ihre Anwendung mit initialisierensam init, aktivieren Sie CloudWatch Application Insights über den interaktiven Ablauf oder mithilfe der --application-insights Option.

Um CloudWatch Application Insights über den AWS SAMCLI interaktiven Flow zu aktivieren, geben Sie ein, **y** wenn Sie dazu aufgefordert werden.

```
Would you like to enable monitoring using CloudWatch Application Insights?
For more info, please view https://docs.aws.amazon.com/AmazonCloudWatch/latest/
monitoring/cloudwatch-application-insights.html [y/N]:
```

Gehen Sie wie folgt vor, um CloudWatch Application Insights mit der --application-insights Option zu aktivieren.

```
sam init --application-insights
```

Weitere Informationen zur Verwendung des sam init Befehls finden Sie untersam init.

Konfiguration mithilfe von AWS SAM Vorlagen

Aktivieren Sie CloudWatch Application Insights, indem Sie die `AWS::ApplicationInsights::Application` Ressourcen `AWS::ResourceGroups::Group` und in Ihren AWS SAM Vorlagen definieren.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  ApplicationResourceGroup:
    Type: AWS::ResourceGroups::Group
    Properties:
      Name:
        Fn::Join:
        - ''
        - - ApplicationInsights-SAM-
        - Ref: AWS::StackName
      ResourceQuery:
        Type: CLOUDFORMATION_STACK_1_0
  ApplicationInsightsMonitoring:
    Type: AWS::ApplicationInsights::Application
    Properties:
```

```
      ResourceGroupName:
        Fn::Join:
          - ''
          - - ApplicationInsights-SAM-
          - Ref: AWS::StackName
        AutoConfigurationEnabled: 'true'
    DependsOn: ApplicationResourceGroup
```

- `AWS::ResourceGroups::Group`— Erstellt eine Gruppe zur Organisation Ihrer AWS Ressourcen, um Aufgaben für eine große Anzahl von Ressourcen gleichzeitig zu verwalten und zu automatisieren. Hier erstellen Sie eine Ressourcengruppe zur Verwendung mit CloudWatch Application Insights. Weitere Informationen zu diesem Ressourcentyp finden Sie `AWS::ResourceGroups::Group`im AWS CloudFormation Benutzerhandbuch.
- `AWS::ApplicationInsights::Application`— Konfiguriert CloudWatch Application Insights für die Ressourcengruppe. Weitere Informationen zu diesem Ressourcentyp finden Sie `AWS::ApplicationInsights::Application`im AWS CloudFormation Benutzerhandbuch.

Beide Ressourcen werden CloudFormation bei der Anwendungsbereitstellung automatisch weitergeleitet. Sie können die CloudFormation Syntax in Ihrer AWS SAM Vorlage verwenden, um CloudWatch Application Insights weiter zu konfigurieren. Weitere Informationen finden Sie unter CloudFormation Vorlagen verwenden im CloudWatch Amazon-Benutzerhandbuch.

Wenn Sie den sam init --application-insights Befehl verwenden, werden diese beiden Ressourcen automatisch in Ihrer AWS SAM Vorlage generiert. Hier ist ein Beispiel für eine generierte Vorlage.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: >
  sam-app-test

  Sample SAM Template for sam-app-test

# More info about Globals: https://github.com/awslabs/serverless-application-model/
blob/master/docs/globals.rst
Globals:
  Function:
    Timeout: 3
    MemorySize: 128

Resources:
```

```
  HelloWorldFunction:
    Type: AWS::Serverless::Function # More info about Function Resource:
 https://github.com/awslabs/serverless-application-model/blob/master/
versions/2016-10-31.md#awsserverlessfunction
    Properties:
      CodeUri: hello_world/
      Handler: app.lambda_handler
      Runtime: python3.9
      Architectures:
      - x86_64
      Events:
        HelloWorld:
          Type: Api # More info about API Event Source: https://github.com/awslabs/
serverless-application-model/blob/master/versions/2016-10-31.md#api
          Properties:
            Path: /hello
            Method: get

  ApplicationResourceGroup:
    Type: AWS::ResourceGroups::Group
    Properties:
      Name:
        Fn::Join:
        - ''
        - - ApplicationInsights-SAM-
        - Ref: AWS::StackName
      ResourceQuery:
       Type: CLOUDFORMATION_STACK_1_0
  ApplicationInsightsMonitoring:
    Type: AWS::ApplicationInsights::Application
    Properties:
      ResourceGroupName:
        Fn::Join:
        - ''
        - - ApplicationInsights-SAM-
        - Ref: AWS::StackName
      AutoConfigurationEnabled: 'true'
    DependsOn: ApplicationResourceGroup

Outputs:
  # ServerlessRestApi is an implicit API created out of Events key under
 Serverless::Function
  # Find out more about other implicit resources you can reference within SAM
```

```
  # https://github.com/awslabs/serverless-application-model/blob/master/docs/internals/
generated_resources.rst#api
  HelloWorldApi:
    Description: API Gateway endpoint URL for Prod stage for Hello World function
    Value: !Sub "https://${ServerlessRestApi}.execute-api.${AWS::Region}.amazonaws.com/
Prod/hello/"
  HelloWorldFunction:
    Description: Hello World Lambda Function ARN
    Value: !GetAtt HelloWorldFunction.Arn
  HelloWorldFunctionIamRole:
    Description: Implicit IAM Role created for Hello World function
    Value: !GetAtt HelloWorldFunctionRole.Arn
```

Nächste Schritte

Verwenden Sie nach der Konfiguration von CloudWatch Application Insights, sam build um Ihre Anwendung sam deploy zu erstellen und bereitzustellen. Alle von CloudWatch Application Insights unterstützten Ressourcen werden für die Überwachung konfiguriert.

- Eine Liste der unterstützten Ressourcen finden Sie unter Unterstützte Logs und Metriken im CloudWatch Amazon-Benutzerhandbuch.
- Informationen zum Zugriff auf CloudWatch Application Insights finden Sie unter Access CloudWatch Application Insights im CloudWatch Amazon-Benutzerhandbuch.

Mit Logins arbeiten AWS SAM

Um die Problembehandlung zu vereinfachen, AWS SAMCLI hat der einen Befehl namens`sam logs`. Mit diesem Befehl können Sie von Ihrer Lambda-Funktion generierte Protokolle von der Befehlszeile abrufen.

> **Note**
>
> Der `sam logs` Befehl funktioniert für alle AWS Lambda Funktionen, nicht nur für die Funktionen, mit denen Sie sie bereitstellen. AWS SAM

Protokolle stapelweise CloudFormation abrufen

Wenn Ihre Funktion Teil eines CloudFormation Stacks ist, können Sie Logs mithilfe der logischen ID der Funktion abrufen:

```
sam logs -n HelloWorldFunction --stack-name mystack
```

Logs nach Lambda-Funktionsnamen abrufen

Oder Sie können Protokolle abrufen, indem Sie den Namen der Funktion verwenden:

```
sam logs -n mystack-HelloWorldFunction-1FJ8PD
```

Protokollierung von Protokollen

Fügen Sie die `--tail` Option hinzu, auf neue Protokolle zu warten und sie zu sehen, sobald sie eintreffen. Dies ist bei der Bereitstellung oder bei der Behebung eines Produktionsproblems hilfreich.

```
sam logs -n HelloWorldFunction --stack-name mystack --tail
```

Protokolle für einen bestimmten Zeitraum anzeigen

Mithilfe der `-e` Optionen `-s` und können Sie Protokolle für einen bestimmten Zeitraum anzeigen:

```
sam logs -n HelloWorldFunction --stack-name mystack -s '10min ago' -e '2min ago'
```

Protokolle filtern

Verwenden Sie `--filter` diese Option, um schnell nach Protokollen zu suchen, die Begriffen, Ausdrücken oder Werten in Ihren Protokollereignissen entsprechen:

```
sam logs -n HelloWorldFunction --stack-name mystack --filter "error"
```

In der Ausgabe werden alle Vorkommen des Wortes „Fehler“ AWS SAMCLI unterstrichen, sodass Sie das Filterschlüsselwort in der Protokollausgabe leicht finden können.

Fehler beim Hervorheben

Wenn Ihre Lambda-Funktion abstürzt oder das Zeitlimit überschritten wird, wird die Timeout-Meldung rot AWS SAMCLI hervorgehoben. Auf diese Weise können Sie innerhalb eines riesigen Stroms von Protokollausgaben ganz einfach bestimmte Ausführungen finden, bei denen das Timeout auftritt.

JSON ist hübsch beim Drucken

Wenn in Ihren Protokollnachrichten JSON-Zeichenketten gedruckt werden, AWS SAMCLI wird die JSON-Datei automatisch hübsch gedruckt, damit Sie die JSON-Daten visuell analysieren und verstehen können.

AWS SAM Referenz

Dieser Abschnitt enthält AWS SAM Referenzmaterial. Dazu gehören AWS SAMCLI Referenzmaterial wie Referenzinformationen zu AWS SAMCLI Befehlen und zusätzliche AWS SAMCLI Informationen wie Konfiguration, Versionskontrolle und Informationen zur Fehlerbehebung. Darüber hinaus enthält dieser Abschnitt Referenzinformationen zur AWS SAM Spezifikation und zur AWS SAM Vorlage, z. B. Referenzinformationen zu Konnektoren, Image-Repositorys und Bereitstellungen.

AWS SAM Spezifikation und Vorlage AWS SAM

Die AWS SAM Spezifikation ist eine Open-Source-Spezifikation unter der Apache 2.0-Lizenz. Die aktuelle Version der AWS SAM Spezifikation ist verfügbar in derAWS SAM Vorlage. AWS SAM Die Spezifikation enthält eine vereinfachte Kurzsyntax, mit der Sie die Funktionen, Ereignisse APIs, Konfigurationen und Berechtigungen Ihrer serverlosen Anwendung definieren.

Sie interagieren mit der AWS SAM Spezifikation über das Projektverzeichnis der AWS SAM Anwendung. Dabei handelt es sich um die Ordner und Dateien, die bei der Ausführung des sam init Befehls erstellt werden. Dieses Verzeichnis enthält die AWS SAM Vorlage, eine wichtige Datei, die Ihre AWS Ressourcen definiert. Die AWS SAM Vorlage ist eine Erweiterung einer AWS CloudFormation Vorlage. Die vollständige Referenz für CloudFormation Vorlagen finden Sie unter Vorlagenreferenz im AWS CloudFormation Benutzerhandbuch.

AWS SAMCLIBefehlsreferenz

Die AWS Serverless Application Model Befehlszeilenschnittstelle (AWS SAMCLI) ist ein Befehlszeilentool, das Sie zusammen mit AWS SAM Vorlagen und unterstützten Integrationen von Drittanbietern verwenden können, um Ihre serverlosen Anwendungen zu erstellen und auszuführen.

Sie können die AWS SAMCLI Befehle verwenden, um Ihre serverlosen Anwendungen zu entwickeln, zu testen und bereitzustellen. AWS Cloud Im Folgenden finden Sie einige Beispiele für AWS SAMCLI Befehle:

- `sam init`— Wenn Sie zum ersten Mal AWS SAMCLI Benutzer sind, können Sie den `sam init` Befehl ohne Parameter ausführen, um eine Hello World-Anwendung zu erstellen. Der Befehl generiert eine vorkonfigurierte AWS SAM Vorlage und einen Beispielanwendungscode in der von Ihnen ausgewählten Sprache.

- `sam local invoke`und `sam local start-api` — Verwenden Sie diese Befehle, um Ihren Anwendungscode lokal zu testen, bevor Sie ihn auf dem AWS Cloud bereitstellen.
- `sam logs`— Verwenden Sie diesen Befehl, um Logs abzurufen, die Ihre Lambda-Funktion generiert. Dies kann Ihnen beim Testen und Debuggen Ihrer Anwendung helfen, nachdem Sie sie auf dem bereitgestellt haben. AWS Cloud
- `sam package`— Verwenden Sie diesen Befehl, um Ihren Anwendungscode und Ihre Abhängigkeiten in einem Bereitstellungspaket zu bündeln. Sie benötigen das Bereitstellungspaket, um Ihre Anwendung auf das hochzuladen AWS Cloud.
- `sam deploy`— Verwenden Sie diesen Befehl, um Ihre serverlose Anwendung auf dem AWS Cloud bereitzustellen. Es erstellt die AWS Ressourcen und legt Berechtigungen und andere Konfigurationen fest, die in der AWS SAM Vorlage definiert sind.

Anweisungen zur Installation von finden AWS SAMCLI Sie unterInstallieren Sie das AWS SAMCLI.

AWS SAM Richtlinienvorlagen

Mit AWS SAM können Sie aus einer Liste von Richtlinienvorlagen auswählen, um die Berechtigungen Ihrer AWS Lambda Funktion auf die Ressourcen zu beschränken, die Ihre Anwendung verwendet. Eine Liste der verfügbaren Richtlinienvorlagen finden Sie unterTabelle mit Richtlinienvorlagen. Allgemeine Informationen zu Richtlinienvorlagen und AWS SAM finden Sie unterAWS SAM Richtlinienvorlagen.

Topics

- AWS SAM Vorlage
- AWS SAMCLIBefehlsreferenz
- AWS SAMCLIKonfigurationsdatei
- AWS SAM Steckverbinderreferenz
- AWS SAM Richtlinienvorlagen
- Bildrepositorien für AWS SAM
- Telemetrie in der AWS SAMCLI
- Richten Sie den Ressourcenzugriff in Ihrer AWS SAM Vorlage ein und verwalten Sie ihn

AWS SAMCLIBefehlsreferenz

Dieser Abschnitt enthält Referenzinformationen zu AWS SAMCLI Befehlen. Dieser enthält Informationen zur Verwendung, eine umfassende Liste der verschiedenen Optionen, die für jeden Befehl verfügbar sind, und zusätzliche Informationen. Gegebenenfalls umfassen zusätzliche Informationen Details wie Argumente, Umgebungsvariablen und Ereignisse. Einzelheiten finden Sie in den einzelnen Befehlen. Anweisungen zur Installation von finden AWS SAMCLI Sie unterInstallieren Sie das AWS SAMCLI.

Themen

- gleicher Build
- sam delete
- sam deploy
- sam init
- sam list
- sam local callback
- sam local execution
- sam local generate-event
- sam local invoke
- Sam lokale Start-API
- Sam Local Start-Lambda
- sam logs
- sam package
- sam pipeline bootstrap
- sam pipeline init
- sam publish
- sam remote callback
- sam remote execution
- sam remote invoke
- sam remote test-event
- Sam Sync
- sam traces

- sam validate

gleicher Build

Diese Seite enthält Referenzinformationen für den AWS Serverless Application Model Befehl Command Line Interface (AWS SAMCLI)`sam build`.

- Eine Einführung in das finden AWS SAMCLI Sie unter Was ist der AWS SAM CLI?
- Eine Dokumentation zur Verwendung des AWS SAMCLI `sam build` Befehls finden Sie unterEinführung in das Bauen mit AWS SAM.

Der `sam build` Befehl bereitet eine Anwendung auf nachfolgende Schritte im Entwickler-Workflow vor, wie z. B. das lokale Testen oder die Bereitstellung auf dem AWS Cloud.

Usage

```
$ sam build <arguments> <options>
```

Argumente

Resource ID (Ressourcen-ID)

Optional. Weist an, eine einzelne Ressource AWS SAM zu erstellen, die in einer AWS SAM Vorlage deklariert ist. Die Build-Artefakte für die angegebene Ressource sind die einzigen, die für nachfolgende Befehle im Workflow verfügbar sind, d. h. `sam package` und`sam deploy`.

Optionen

`--base-dir, -s` *`DIRECTORY`*

Löst relative Pfade zum Quellcode der Funktion oder Ebene in Bezug auf dieses Verzeichnis auf. Verwenden Sie diese Option, wenn Sie ändern möchten, wie relative Pfade zu Quellcode-Ordnern aufgelöst werden. Standardmäßig werden relative Pfade in Bezug auf den Speicherort der AWS SAM Vorlage aufgelöst.

Zusätzlich zu den Ressourcen in der Stammanwendung oder dem Stack, den Sie erstellen, bezieht sich diese Option auch auf verschachtelte Anwendungen oder Stacks.

Diese Option gilt für die folgenden Ressourcentypen und Eigenschaften:

- Ressourcentyp: `AWS::Serverless::Function` Eigentum: `CodeUri`
- Ressourcentyp: `AWS::Serverless::Function` Ressourcenattribut: `Metadata` Eintrag: `DockerContext`
- Ressourcentyp: `AWS::Serverless::LayerVersion` Immobilie: `ContentUri`
- Ressourcentyp: `AWS::Lambda::Function` Immobilie: `Code`
- Ressourcentyp: `AWS::Lambda::LayerVersion` Immobilie: `Content`

`--beta-features | --no-beta-features`

Betafunktionen zulassen oder verweigern.

`--build-dir, -b` *`DIRECTORY`*

Der Pfad zu einem Verzeichnis, in dem die erstellten Artefakte gespeichert sind. Dieses Verzeichnis und sein gesamter Inhalt werden mit dieser Option entfernt.

`--build-image` *`TEXT`*

Die URI des Container-Images, das Sie für den Build abrufen möchten. Ruft das Container-Image standardmäßig aus Amazon ECR AWS SAM Public ab. Verwenden Sie diese Option, um das Bild von einem anderen Ort abzurufen.

Sie können diese Option mehrfach angeben. Jede Instanz dieser Option kann entweder eine Zeichenfolge oder ein Schlüssel-Wert-Paar annehmen. Wenn Sie eine Zeichenfolge angeben, ist dies die URI des Container-Images, das für alle Ressourcen in Ihrer Anwendung verwendet werden soll. Beispiel, `sam build --use-container --build-image amazon/aws-sam-cli-build-image-python3.8`. Wenn Sie ein Schlüssel-Wert-Paar angeben, ist der Schlüssel der Ressourcenname, und der Wert ist der URI des Container-Images, das für diese Ressource verwendet werden soll. Zum Beispiel `sam build --use-container --build-image Function1=amazon/aws-sam-cli-build-image-python3.8`. Mit Schlüssel-Wert-Paaren können Sie verschiedene Container-Images für verschiedene Ressourcen angeben.

Diese Option gilt nur, wenn die `--use-container` Option angegeben ist, andernfalls wird ein Fehler auftreten.

`--build-in-source | --no-build-in-source`

Stellen `--build-in-source` Sie sicher, dass Ihr Projekt direkt im Quellordner erstellt wird.

Die `--build-in-source` Option unterstützt die folgenden Laufzeiten und Build-Methoden:

- Runtimes — Jede Node.js Laufzeit, die von der `sam init --runtime` Option unterstützt wird.
- Methoden erstellen —`Makefile`,`esbuild`.

Die `--build-in-source` Option ist nicht mit den folgenden Optionen kompatibel:

- `--hook-name`
- `--use-container`

Standardwert: `--no-build-in-source`

`--cached | --no-cached`

Aktiviert oder deaktiviert zwischengespeicherte Builds. Verwenden Sie diese Option, um Build-Artefakte wiederzuverwenden, die sich gegenüber früheren Builds nicht geändert haben. AWS SAM bewertet, ob Sie Dateien in Ihrem Projektverzeichnis geändert haben. Standardmäßig werden Builds nicht zwischengespeichert. Wenn die `--no-cached` Option aufgerufen wird, überschreibt sie die `cached = true` Einstellung in samconfig.toml.

> **Note**
>
> AWS SAM bewertet nicht, ob Sie Module von Drittanbietern geändert haben, von denen Ihr Projekt abhängt, für die Sie keine bestimmte Version bereitgestellt haben. Wenn Ihre Python-Funktion beispielsweise eine `requirements.txt` Datei mit dem Eintrag `requests=1.x` enthält und die neueste Version des Anforderungsmoduls von `1.1` zu geändert wird`1.2`, wird AWS SAM die neueste Version erst abgerufen, wenn Sie einen Build ohne Cache ausführen.

`--cache-dir`

Das Verzeichnis, in dem die Cache-Artefakte gespeichert werden, wenn `--cached` angegeben. Das Standard-Cache-Verzeichnis ist`.aws-sam/cache`.

`--config-env` *TEXT*

Der Umgebungsname, der die Standardparameterwerte in der zu verwendenden Konfigurationsdatei angibt. Der Standardwert ist „default“. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--config-file` *PATH*

Der Pfad und der Dateiname der Konfigurationsdatei, die die zu verwendenden Standardparameterwerte enthält. Der Standardwert ist "`samconfig.toml`" im Stammverzeichnis des Projektverzeichnisses. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--container-env-var, -e` *TEXT*

Umgebungsvariablen, die an den Build-Container übergeben werden sollen. Sie können diese Option mehrfach angeben. Jede Instanz dieser Option verwendet ein Schlüssel-Wert-Paar, wobei der Schlüssel die Ressourcen- und Umgebungsvariable ist und der Wert der Wert der Umgebungsvariablen ist. Beispiel: `--container-env-var Function1.GITHUB_TOKEN=TOKEN1 --container-env-var Function2.GITHUB_TOKEN=TOKEN2`.

Diese Option gilt nur, wenn die `--use-container` Option angegeben ist, andernfalls wird ein Fehler auftreten.

`--container-env-var-file, -ef` *PATH*

Der Pfad und der Dateiname einer JSON-Datei, die Werte für die Umgebungsvariablen des Containers enthält. Weitere Hinweise zu Container-Umgebungsvariablendateien finden Sie unterDatei mit Container-Umgebungsvariablen.

Diese Option gilt nur, wenn die `--use-container` Option angegeben ist, andernfalls wird ein Fehler auftreten.

`--debug`

Aktiviert die Debug-Protokollierung, um die AWS SAMCLI generierten Debug-Meldungen auszudrucken und Zeitstempel anzuzeigen.

`--docker-network` *TEXT*

Gibt den Namen oder die ID eines vorhandenen Docker Netzwerks an, zu dem Docker Lambda-Container eine Verbindung herstellen sollen, zusammen mit dem Standard-Bridge-Netzwerk. Wenn nicht angegeben, stellen die Lambda-Container nur eine Verbindung zum Docker Standard-Bridge-Netzwerk her.

`--exclude, -x`

Der Name der Ressource (n), die aus dem `sam build` ausgeschlossen werden soll. Wenn Ihre Vorlage beispielsweise `Function1Function2`, und enthält `Function3` und Sie ausführen`sam build --exclude Function2`, `Function3` wird nur `Function1` und erstellt.

`--help`

Zeigt diese Meldung an und beendet das Programm.

`--hook-name` *`TEXT`*

Der Name des Hooks, der zur Erweiterung der AWS SAMCLI Funktionalität verwendet wird.

Zulässige Werte:`terraform`.

`--manifest , -m` *`PATH`*

Der Pfad zu einer benutzerdefinierten Abhängigkeitsmanifestdatei (z. B. package.json), die anstelle der Standarddatei verwendet werden soll.

`--mount-symlinks`

Stellt sicher, dass AWS SAMCLI immer symbolische Links bereitgestellt werden, die in den zu erstellenden oder aufzurufenden Dateien vorhanden sind. Dies gilt nur für Symlinks im Verzeichnis der obersten Ebene (d. h. für Symlinks, die sich direkt im Stammverzeichnis der Funktion befinden). Standardmäßig werden Symlinks nicht gemountet, außer denen, die für die Verwendung `build-in-source` in NodeJS benötigt werden. `node_modules`

`--no-use-container`

Eine Option, mit der Sie das IDE-Toolkit verwenden können, um das Standardverhalten festzulegen. Sie können es auch verwendensam `build --no-use-container`, um anstelle eines Docker-Containers einen Build auf Ihrem lokalen Computer auszuführen.

`--output` *`[ text | json ]`*

Das Format der Befehlsausgabe:

- `text`— Druckt normale, für Menschen lesbare Ausgaben. Dies ist der Standardwert.
- `json`— Druckt eine strukturierte, maschinenlesbare Ausgabe. Verwenden Sie dieses Format, wenn Sie die Ausgabe programmgesteuert verwenden möchten, z. B. in CI/CD Pipelines, IDE-Erweiterungen und Tools. AI-assisted

Wenn Sie angeben`--output json`, AWS SAMCLI druckt das ein einzelnes JSON-Objekt, nachdem der Befehl abgeschlossen ist. Bei Erfolg enthält dieses Objekt Felder wie den Build-

Status, das Build-Verzeichnis, die Vorlagendatei und eine Liste der Ressourcen, die erstellt wurden. Im Fehlerfall enthält es ein `error` Objekt, das beschreibt, was schief gelaufen ist.

`--parallel`

Ermöglicht parallele Builds. Verwenden Sie diese Option, um die Funktionen und Ebenen Ihrer AWS SAM Vorlage parallel zu erstellen. Standardmäßig werden die Funktionen und Ebenen nacheinander erstellt.

`--parameter-overrides`

(Optional) Eine Zeichenfolge, die CloudFormation Parameterüberschreibungen enthält, die als Schlüssel-Wert-Paare kodiert sind. Verwendet dasselbe Format wie (). AWS Command Line Interface AWS CLI Zum Beispiel: '`ParameterKey=KeyPairName`, `ParameterValue` = `MyKey` `ParameterKey` =`InstanceType`, `ParameterValue` =`t1.micro`'. Diese Option ist nicht kompatibel mit`--hook-name`.

`--profile` *`TEXT`*

Das spezifische Profil aus Ihrer Anmeldeinformationsdatei, das AWS Anmeldeinformationen abruft.

`--region` *`TEXT`*

Das AWS-Region , für das die Bereitstellung erfolgen soll. Beispiel: us-east-1.

`--save-params`

Speichern Sie die Parameter, die Sie in der Befehlszeile angeben, in der AWS SAM Konfigurationsdatei.

`--skip-prepare-infra`

Überspringt die Vorbereitungsphase, wenn keine Änderungen an der Infrastruktur vorgenommen wurden. Mit der `--hook-name` Option verwenden.

`--skip-pull-image`

Gibt an, ob der Befehl das Herunterladen des neuesten Docker-Images für die Lambda-Laufzeit überspringen soll.

`--template-file, --template, -t` *`PATH`*

Der Pfad und der Dateiname der AWS SAM Vorlagendatei. `[default: template.[yaml|yml]]` Diese Option ist nicht kompatibel mit`--hook-name`.

`--terraform-project-root-path`

Der relative oder absolute Pfad zum Verzeichnis auf oberster Ebene, das Ihre Terraform Konfigurationsdateien oder den Funktionsquellcode enthält. Wenn sich diese Dateien außerhalb des Verzeichnisses befinden, das Ihr Terraform Root-Modul enthält, verwenden Sie diese Option, um den absoluten oder relativen Pfad anzugeben. Für diese Option muss die Einstellung auf gesetzt `--hook-name` sein`terraform`.

`--use-buildkit | --no-use-buildkit`

Aktiviere oder deaktiviere die Verwendung des BuildKit Plugins für Container-Image-Builds. Dabei wird das Finch oder verwendet DockerCLIs. Standardmäßig verwendet Builds den Legacy-Builder aus dem SDK. Wenn die `--no-use-buildkit` Option aufgerufen wird, überschreibt sie die `use-buildkit = true` Einstellung in samconfig.toml.

`--use-container`, `-u`

Wenn Ihre Funktionen von Paketen abhängen, die nativ kompilierte Abhängigkeiten haben, verwenden Sie diese Option, um Ihre Funktion in einem Docker-Container zu erstellen. Lambda-like

Beispiel

Ein detailliertes Beispiel und eine ausführliche Anleitung zur Verwendung des `sam build` Unterbefehls finden Sie unter. Einführung in das Bauen mit AWS SAM

sam delete

Diese Seite enthält Referenzinformationen für den AWS Serverless Application Model Befehl Command Line Interface (AWS SAMCLI)`sam delete`.

Eine Einführung in den finden AWS SAMCLI Sie unter Was ist der AWS SAM CLI?

Der `sam delete` Befehl löscht eine AWS SAM Anwendung, indem er den CloudFormation Stack, die Artefakte, die verpackt und in Amazon S3 und Amazon ECR bereitgestellt wurden, und die AWS SAM Vorlagendatei löscht.

Dieser Befehl prüft auch, ob ein Amazon ECR-Companion-Stack bereitgestellt ist, und fordert den Benutzer in diesem Fall auf, diesen Stack und die Amazon ECR-Repositorys zu löschen. Wenn `--no-prompts` angegeben, werden Companion-Stacks und Amazon ECR-Repositorys standardmäßig gelöscht.

Usage

```
$ sam delete <options>
```

Optionen

`--config-env` *TEXT*

Der Umgebungsname, der die Standardparameterwerte in der zu verwendenden Konfigurationsdatei angibt. Der Standardwert ist `default`. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--config-file` *PATH*

Der Pfad und der Dateiname der Konfigurationsdatei, die die zu verwendenden Standardparameterwerte enthält. Der Standardwert befindet sich `samconfig.toml` im Stammverzeichnis des Projektverzeichnisses. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--debug`

Aktiviert die Debug-Protokollierung, um die AWS SAMCLI generierte Debug-Nachricht zu drucken und Zeitstempel anzuzeigen.

`--help`

Zeigt diese Meldung an und beendet das Programm.

`--no-prompts`

Geben Sie diese Option an, um im nicht interaktiven Modus zu AWS SAM arbeiten. Der Stack-Name muss entweder zusammen mit der `--stack-name` Option oder in der `toml` Konfigurationsdatei angegeben werden.

`--profile` *TEXT*

Das spezifische Profil aus Ihrer Anmeldeinformationsdatei, das die AWS Anmeldeinformationen abruft.

`--region` *TEXT*

Die AWS Region, in der die Bereitstellung erfolgen soll. Beispiel: us-east-1.

`--s3-bucket`

Der Pfad des Amazon S3 S3-Buckets, den Sie löschen möchten.

`--s3-prefix`

Das Präfix des Amazon S3 S3-Buckets, den Sie löschen möchten.

`--save-params`

Speichern Sie die Parameter, die Sie in der Befehlszeile angeben, in der AWS SAM Konfigurationsdatei.

`--stack-name` *`TEXT`*

Der Name des CloudFormation Stacks, den Sie löschen möchten.

Beispiele

Der folgende Befehl löscht den Stack`MY-STACK`.

```
$ sam delete --stack-name MY-STACK
```

Der folgende Befehl löscht den Stack `MY-STACK` und den S3-Bucket: `sam-s3-demo-bucket`

```
$ sam delete \
    --stack-name MyStack \
    --s3-bucket MySAMBucket
```

sam deploy

Diese Seite enthält Referenzinformationen für den AWS Serverless Application Model Befehl Command Line Interface (AWS SAMCLI)`sam deploy`.

- Eine Einführung in das finden AWS SAMCLI Sie unter Was ist der AWS SAM CLI?
- Eine Dokumentation zur Verwendung des AWS SAMCLI `sam deploy` Befehls finden Sie unterEinführung in die Bereitstellung mit AWS SAM.

Der `sam deploy` Befehl stellt dem Benutzer eine AWS Cloud Anwendung AWS CloudFormation zur Verfügung.

Usage

```
$ <environment variables> sam deploy <options>
```

Umgebungsvariablen

`SAM_CLI_POLL_DELAY`

Stellen Sie in Ihrer Shell die `SAM_CLI_POLL_DELAY` Umgebungsvariable auf einen Wert von Sekunden ein, um zu konfigurieren, wie oft die AWS SAM-CLI den CloudFormation Stack-Status überprüft. Dies ist nützlich, wenn Sie Drosselungen von sehen. CloudFormation Diese Umgebungsvariable wird für die Abfrage von `describe_stack` API-Aufrufen verwendet, die während der Ausführung getätigt werden. `sam deploy`

Das Folgende ist ein Beispiel für diese Variable:

```
$ SAM_CLI_POLL_DELAY=5 sam deploy
```

Optionen

`--capabilities` *`LIST`*

Eine Liste von Funktionen, die Sie angeben müssen, um das Erstellen bestimmter Stacks CloudFormation zu ermöglichen. Einige Stack-Vorlagen können Ressourcen enthalten, die sich auf Ihre Berechtigungen auswirken AWS-Konto, z. B. durch das Erstellen neuer Benutzer AWS Identity and Access Management (IAM). Für diese Stacks müssen Sie deren Funktionen explizit bestätigen, indem Sie diese Option angeben. Die einzigen gültigen Werte sind `CAPABILITY_IAM` und `CAPABILITY_NAMED_IAM`. Wenn Sie über IAM-Ressourcen verfügen, können Sie beide Funktionen angeben. Wenn Sie IAM-Ressourcen mit benutzerdefinierten Namen haben, müssen Sie Folgendes angeben. `CAPABILITY_NAMED_IAM` Wenn Sie diese Option nicht angeben, gibt der Vorgang einen `InsufficientCapabilities` Fehler zurück.

Wenn Sie eine Anwendung bereitstellen, die verschachtelte Anwendungen enthält, müssen Sie bestätigen, dass `CAPABILITY_AUTO_EXPAND` die Anwendung verschachtelte Anwendungen enthält. Weitere Informationen finden Sie unter Bereitstellen verschachtelter Anwendungen.

`--config-env` *`TEXT`*

Der Umgebungsname, der die Standardparameterwerte in der zu verwendenden Konfigurationsdatei angibt. Der Standardwert ist `default`. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--config-file` *PATH*

Der Pfad und der Dateiname der Konfigurationsdatei, die die zu verwendenden Standardparameterwerte enthält. Der Standardwert befindet sich `samconfig.toml` im Stammverzeichnis des Projektverzeichnisses. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--confirm-changeset | --no-confirm-changeset`

Sie werden aufgefordert, zu bestätigen, ob das AWS SAMCLI berechnete Changeset bereitgestellt wird.

`--debug`

Schalten Sie die Debug-Protokollierung ein, um die AWS SAMCLI generierte Debug-Meldung auszudrucken und Zeitstempel anzuzeigen.

`--disable-rollback | --no-disable-rollback`

Geben Sie an, ob Ihr CloudFormation Stack zurückgesetzt werden soll, wenn während einer Bereitstellung ein Fehler auftritt. Wenn während einer Bereitstellung ein Fehler auftritt, wird Ihr CloudFormation Stack standardmäßig in den letzten stabilen Zustand zurückgesetzt. Wenn Sie angeben `--disable-rollback` und während einer Bereitstellung ein Fehler auftritt, werden Ressourcen, die vor dem Auftreten des Fehlers erstellt oder aktualisiert wurden, nicht zurückgesetzt.

`--express | --no-express`

Gibt an, ob der CloudFormation Expressmodus verwendet werden soll. Wenn diese Option aktiviert ist, werden die Stapelvorgänge abgeschlossen, sobald die Ressourcenkonfiguration angewendet wurde. Die Ressourcen werden weiterhin im Hintergrund bereit gestellt. Die Bereitstellungszeit kann um das bis zu 4-fache reduziert werden.

Der AWS SAMCLI übergibt das Rollback-Verhalten an den CloudFormation `DeploymentConfig` Parameter, wenn Sie es ebenfalls angeben`--disable-rollback`.

Der Standardwert ist `--no-express`.

`--fail-on-empty-changeset | --no-fail-on-empty-changeset`

Geben Sie an, ob ein Exit-Code ungleich Null zurückgegeben werden soll, wenn keine Änderungen am Stack vorgenommen werden müssen. Das Standardverhalten besteht darin, einen Exitcode ungleich Null zurückzugeben.

`--force-upload`

Geben Sie diese Option an, um Artefakte hochzuladen, auch wenn sie mit vorhandenen Artefakten im Amazon S3-Bucket übereinstimmen. Passende Artefakte werden überschrieben.

`--guided, -g`

Geben Sie diese Option an, damit die AWS SAMCLI Benutzeranweisungen Sie durch die Bereitstellung führen.

`--help`

Diese Meldung anzeigen und das Programm beenden.

`--image-repositories` *TEXT*

Eine Zuordnung von Funktionen zu ihrer Amazon ECR-Repository-URI. Referenzfunktionen anhand ihrer logischen ID. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam deploy --image-repositories Function1=123456789012.dkr.ecr.us-
east-1.amazonaws.com/my-repo
```

Sie können diese Option mehrmals in einem einzigen Befehl angeben.

`--image-repository` *TEXT*

Der Name des Amazon ECR-Repositorys, in das dieser Befehl das Bild Ihrer Funktion hochlädt. Diese Option ist für Funktionen erforderlich, die mit dem `Image` Pakettyp deklariert wurden.

`--kms-key-id` *TEXT*

Die ID eines AWS Key Management Service (AWS KMS) -Schlüssels, der zum Verschlüsseln von Artefakten verwendet wird, die sich im Amazon S3-Bucket befinden. Wenn Sie diese Option nicht angeben, werden S3-managed Amazon-Verschlüsselungsschlüssel AWS SAM verwendet.

`--metadata`

Eine Karte mit Metadaten, die an alle Artefakte angehängt werden, auf die in Ihrer Vorlage verwiesen wird.

`--no-execute-changeset`

Gibt an, ob das Changeset angewendet werden soll. Geben Sie diese Option an, wenn Sie Ihre Stack-Änderungen sehen möchten, bevor Sie das Changeset anwenden. Dieser Befehl erstellt ein CloudFormation Changeset und wird dann beendet, ohne das Changeset anzuwenden. Um das Changeset anzuwenden, führen Sie denselben Befehl ohne diese Option aus.

`--no-progressbar`

Zeigen Sie keine Fortschrittsleiste an, wenn Sie Artefakte auf Amazon S3 hochladen.

`--notification-arns` *`LIST`*

Eine Liste der thematischen ARNs von Amazon Simple Notification Service (Amazon SNS), die dem Stack CloudFormation zugeordnet sind.

`--on-failure [ROLLBACK | DELETE | DO_NOTHING]`

Geben Sie die Aktion an, die ausgeführt werden soll, wenn ein Stack nicht erstellt werden kann.

Die folgenden Optionen sind verfügbar:

- `ROLLBACK`— Setzt den Stack auf einen früheren als funktionierend bekannten Zustand zurück.
- `DELETE`— Setzt den Stapel auf einen früheren als funktionierend bekannten Zustand zurück, falls ein solcher vorhanden ist. Andernfalls wird der Stapel gelöscht.
- `DO_NOTHING`— Der Stapel wird weder zurückgesetzt noch gelöscht. Der Effekt ist der gleiche wie der von. `--disable-rollback`

Das Standardverhalten ist `ROLLBACK`.

> **Note**
>
> Sie können entweder die `--disable-rollback` Option oder die `--on-failure` Option angeben, aber nicht beide.

`--output` *`[ text | json ]`*

Das Format der Befehlsausgabe:

- `text`— Druckt normale, für Menschen lesbare Ausgaben. Dies ist der Standardwert.
- `json`— Druckt eine strukturierte, maschinenlesbare Ausgabe. Verwenden Sie dieses Format, wenn Sie die Ausgabe programmgesteuert verwenden möchten, z. B. in CI/CD Pipelines, IDE-Erweiterungen und Tools. AI-assisted

Eine Bereitstellung läuft über einen bestimmten Zeitraum. Wenn Sie angeben`--output json`, AWS SAMCLI druckt das im Verlauf der Bereitstellung einen Stream von JSON-Objekten, eines pro Zeile. Jedes Objekt enthält ein `type` Feld, das die Art der darin enthaltenen Informationen identifiziert, z. B. den Changeset, Bereitstellungsereignisse, Stack-Ausgaben und

das Endergebnis. Im Fehlerfall enthält das endgültige Objekt ein `error` Objekt, das beschreibt, was schief gelaufen ist.

> **Note**
>
> Die `--output json` Option ist nicht mit Optionen kompatibel, die interaktive Eingabeaufforderungen erfordern, wie z. B. `--guided` und`--confirm-changeset`.

`--parameter-overrides` *LIST*

Eine Zeichenfolge, die CloudFormation Parameterüberschreibungen enthält, die als Schlüssel-Wert-Paare kodiert sind. Bei jeder Überschreibung wird das Format verwendet. `ParameterKey=name,ParameterValue=value` Mehrere Überschreibungen werden durch Leerzeichen getrennt. Nachfolgend finden Sie zwei Beispiele:

```
$ sam deploy --parameter-overrides ParameterKey=value1,ParameterValue=value2
```

```
$ sam deploy --parameter-overrides ParameterKey=value1,ParameterValue=value2
 ParameterKey=hello,ParameterValue=world ParameterKey=apple,ParameterValue=banana
```

`--profile` *TEXT*

Das spezifische Profil aus Ihrer Anmeldeinformationsdatei, das Anmeldeinformationen abruft AWS .

`--region` *TEXT*

Das AWS-Region , für das die Bereitstellung erfolgen soll. Beispiel: us-east-1.

`--resolve-image-repos`

Erstellen Sie automatisch Amazon ECR-Repositorys, die Sie für das Paketieren und Bereitstellen von Bereitstellungen ohne Anleitung verwenden können. Diese Option gilt nur für Funktionen und Ebenen mit Angabe. `PackageType: Image` Wenn Sie die `--guided` Option angeben, wird sie AWS SAMCLI ignoriert`--resolve-image-repos`.

> **Note**
>
> Wenn mit dieser Option AWS SAM automatisch Amazon ECR-Repositorys für Funktionen oder Ebenen erstellt werden und Sie diese Funktionen oder Ebenen später aus Ihrer

AWS SAM Vorlage löschen, werden die entsprechenden Amazon ECR-Repositorys automatisch gelöscht.

`--resolve-s3`

Erstellen Sie automatisch einen Amazon S3-Bucket, den Sie für die Paketierung und Bereitstellung für nicht geführte Bereitstellungen verwenden können. Wenn Sie die `--guided` Option angeben, wird sie von der AWS SAM CLI ignoriert. `--resolve-s3` Wenn Sie sowohl die `--s3-bucket` `--resolve-s3` Optionen als auch angeben, tritt ein Fehler auf.

`--role-arn` *TEXT*

Der Amazon-Ressourcenname (ARN) einer IAM-Rolle, die bei der Anwendung des Changesets übernommen CloudFormation wird.

`--s3-bucket` *TEXT*

Der Name des Amazon S3-Buckets, in den dieser Befehl Ihre Vorlage hochlädt. CloudFormation Wenn Ihre Vorlage größer als 51.200 Byte ist, ist entweder die `--s3-bucket` Option oder die `--resolve-s3` Option erforderlich. Wenn Sie sowohl die `--s3-bucket` `--resolve-s3` Optionen als auch angeben, tritt ein Fehler auf.

`--s3-prefix` *TEXT*

Das Präfix, das den Namen der Artefakte hinzugefügt wird, die in den Amazon S3-Bucket hochgeladen werden. Der Präfixname ist ein Pfadname (Ordnername) für den Amazon S3-Bucket.

`--save-params`

Speichern Sie die Parameter, die Sie in der Befehlszeile angeben, in der AWS SAM Konfigurationsdatei.

`--signing-profiles` *LIST*

Die Liste der Signaturprofile, mit denen Sie Ihre Bereitstellungspakete signieren können. Diese Option verwendet eine Liste von Schlüssel-Wert-Paaren, wobei der Schlüssel der Name der Funktion oder Ebene ist, die signiert werden soll, und der Wert das Signaturprofil ist, wobei ein optionaler Profilbesitzer durch getrennt ist. `:` Beispiel, `FunctionNameToSign=SigningProfileName1 LayerNameToSign=SigningProfileName2:SigningProfileOwner`.

`--stack-name` *TEXT*

(Erforderlich) Der Name des CloudFormation Stacks, auf dem Sie die Bereitstellung durchführen. Wenn Sie einen vorhandenen Stack angeben, aktualisiert der Befehl den Stack. Wenn Sie einen neuen Stapel angeben, wird er mit dem Befehl erstellt.

`--tags` *LIST*

Eine Liste von Tags, die dem Stapel zugeordnet werden sollen, der erstellt oder aktualisiert wird. CloudFormation gibt diese Tags auch an Ressourcen im Stack weiter, die sie unterstützen.

`--template-file, --template, -t` *PATH*

Der Pfad und der Dateiname, in dem sich Ihre AWS SAM Vorlage befindet.

> **Note**
>
> Wenn Sie diese Option angeben, werden AWS SAM nur die Vorlage und die lokalen Ressourcen bereitgestellt, auf die sie verweist.

`--use-json`

Gibt JSON für die CloudFormation Vorlage aus. Die Standardausgabe ist YAML.

Beispiel

Ein detailliertes Beispiel und eine ausführliche Anleitung zur Verwendung des `sam deploy` Unterbefehls finden Sie unter. Einführung in die Bereitstellung mit AWS SAM

sam init

Diese Seite enthält Referenzinformationen für den AWS Serverless Application Model Befehl Command Line Interface (AWS SAMCLI)`sam init`.

- Eine Einführung in das finden AWS SAMCLI Sie unter Was ist der AWS SAM CLI?
- Eine Dokumentation zur Verwendung des AWS SAMCLI `sam init` Befehls finden Sie unterErstellen Sie Ihre Bewerbung in AWS SAM.

Der `sam init` Befehl bietet Optionen zum Initialisieren einer neuen serverlosen Anwendung.

Usage

```
$ sam init <options>
```

Optionen

`--app-template` *TEXT*

Die ID der verwalteten Anwendungsvorlage, die Sie verwenden möchten. Wenn Sie sich nicht sicher sind, rufen Sie `sam init` ohne Optionen für einen interaktiven Workflow an.

Dieser Parameter ist erforderlich, wenn `--no-interactive` er angegeben und nicht bereitgestellt `--location` wird.

Dieser Parameter ist nur in AWS SAMCLI Version 0.30.0 und höher verfügbar. Die Angabe dieses Parameters mit einer früheren Version führt zu einem Fehler.

`--application-insights | --no-application-insights`

Aktivieren Sie die Amazon CloudWatch Application Insights-Überwachung für Ihre Anwendung. Weitere Informationen hierzu finden Sie unter Verwenden von CloudWatch Application Insights zur Überwachung Ihrer AWS SAM serverlosen Anwendungen.

Die Standardoption ist `--no-application-insights`.

`--architecture, -a` *[x86_64 | arm64]*

Die Befehlssatzarchitektur für die Lambda-Funktionen Ihrer Anwendung. Geben Sie eines von `x86_64` oder `arm64` an.

`--base-image` *[amazon/dotnet8-base | amazon/dotnet6-base | amazon/java25-base | amazon/java21-base | amazon/java17-base | amazon/java17.al2023-base | amazon/java11-base | amazon/java11.al2023-base | amazon/java8.al2023-base | amazon/nodejs24.x-base | amazon/nodejs22.x-base | amazon/nodejs20.x-base | amazon/nodejs18.x-base | amazon/nodejs16.x-base | amazon/python3.14-base | amazon/python3.13-base | amazon/python3.12-base | amazon/python3.11-base | amazon/python3.10-base | amazon/python3.9-base | amazon/python3.8-base | amazon/ruby4.0-base | amazon/ruby3.4-base | amazon/ruby3.3-base | amazon/ruby3.2-base]*

Das Basis-Image Ihrer Anwendung. Diese Option gilt nur, wenn der Pakettyp lautet`Image`.

Dieser Parameter ist erforderlich, wenn er angegeben `--no-interactive` `--package-type` ist`Image`, als angegeben `--location` ist und nicht angegeben ist.

`--config-env` *`TEXT`*

Der Umgebungsname, der die Standardparameterwerte in der zu verwendenden Konfigurationsdatei angibt. Der Standardwert ist „default“. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--config-file` *`PATH`*

Der Pfad und der Dateiname der Konfigurationsdatei, die die zu verwendenden Standardparameterwerte enthält. Der Standardwert ist „samconfig.toml“ im Stammverzeichnis des Projektverzeichnisses. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--debug`

Aktiviert die Debug-Protokollierung, um die AWS SAMCLI generierten Debug-Meldungen auszudrucken und Zeitstempel anzuzeigen.

`--dependency-manager, -d` *`[ gradle | mod | maven | bundler | npm | cli-package | pip ]`*

Der Abhängigkeitsmanager Ihrer Lambda-Laufzeit.

`--extra-content`

Überschreiben Sie alle benutzerdefinierten Parameter in der `cookiecutter.json` Konfiguration der Vorlage, `{"customParam1": "customValue1", "customParam2":"customValue2"}` z. B.

`--help, -h`

Zeigt diese Meldung an und beendet das Programm.

`--location, -l` *`TEXT`*

Der Speicherort der Vorlage oder der Anwendung (Git, Mercurial HTTP/HTTPS, .zip-Datei, Pfad).

Dieser Parameter ist erforderlich, wenn `--no-interactive` und `--runtime--name`, und nicht angegeben `--app-template` sind.

Für Git-Repositorys müssen Sie den Speicherort des Stammverzeichnisses des Repositorys verwenden.

Für lokale Pfade muss die Vorlage entweder im ZIP-Datei- oder https://cookiecutter.readthedocs.io/en/latest/README.html Cookiecutter-Format vorliegen.

`--name, -n` *TEXT*

Der Name Ihres Projekts, das als Verzeichnis generiert werden soll.

Dieser Parameter ist erforderlich, wenn er angegeben `--no-interactive` `--location` ist und nicht bereitgestellt wird.

`--no-input`

Deaktiviert die Cookiecutter-Aufforderung und akzeptiert die in der Vorlagenkonfiguration definierten vcfdefault-Werte.

`--no-interactive`

Deaktiviert die interaktive Aufforderung zur Eingabe von Init-Parametern und schlägt fehl, wenn erforderliche Werte fehlen.

`--output` *[text | json]*

Das Format der Befehlsausgabe:

- `text`— Druckt normale, für Menschen lesbare Ausgaben. Dies ist der Standardwert.
- `json`— Druckt eine strukturierte, maschinenlesbare Ausgabe. Verwenden Sie dieses Format, wenn Sie die Ausgabe programmgesteuert verwenden möchten, z. B. in CI/CD Pipelines, IDE-Erweiterungen und Tools. AI-assisted

Wenn Sie angeben`--output json`, AWS SAMCLI druckt das ein einzelnes JSON-Objekt, nachdem der Befehl abgeschlossen ist. Bei Erfolg enthält dieses Objekt Felder wie das Projektverzeichnis und die Vorlagendatei. Sie können die Ausgabe mit nachfolgenden Befehlen verketten, z. `sam build` B. Bei einem Fehler enthält es ein `error` Objekt, das beschreibt, was schief gelaufen ist.

> **Note**
>
> Die `--output json` Option ist nicht mit dem interaktiven Arbeitsablauf kompatibel. Um sie zu verwenden, geben Sie einen vollständig nicht interaktiven Lauf an, indem Sie alle erforderlichen Optionen angeben.

`--output-dir, -o` *PATH*

Der Ort, an dem die initialisierte Anwendung ausgegeben wird.

`--package-type` *[Zip | Image]*

Der Pakettyp der Beispielanwendung. `Zip`erstellt ein ZIP-Dateiarchiv und `Image` erstellt ein Container-Image.

`--runtime, -r` *[dotnet8 | dotnet6 | java25 | java21 | java17 | java17.al2023 | java11 | java11.al2023 | java8.al2023 | nodejs24.x | nodejs22.x | nodejs20.x | nodejs18.x | nodejs16.x | python3.14 | python3.13 | python3.12 | python3.11 | python3.10 | python3.9 | python3.8 | ruby4.0 | ruby3.4 | ruby3.3 | ruby3.2]*

Die Lambda-Laufzeit Ihrer Anwendung. Diese Option gilt nur, wenn der Pakettyp lautet`Zip`.

Dieser Parameter ist erforderlich, wenn er angegeben `--no-interactive` `--package-type` ist`Zip`, als angegeben `--location` ist und nicht angegeben ist.

`--save-params`

Speichern Sie die Parameter, die Sie in der Befehlszeile angeben, in der AWS SAM Konfigurationsdatei.

`--tracing | --no-tracing`

Aktivieren Sie das AWS X-Ray Tracing für Ihre Lambda-Funktionen.

Beispiel

Ein detailliertes Beispiel und eine ausführliche Anleitung zur Verwendung des `sam init` Unterbefehls finden Sie unter. Erstellen Sie Ihre Bewerbung in AWS SAM

sam list

Diese Seite enthält Referenzinformationen für den AWS Serverless Application Model Befehl Command Line Interface (AWS SAMCLI)`sam list`.

Eine Einführung in den finden AWS SAMCLI Sie unter Was ist der AWS SAM CLI?

Der `sam list` Befehl gibt wichtige Informationen über die Ressourcen in Ihrer serverlosen Anwendung und den Status Ihrer serverlosen Anwendung aus. Verwenden Sie es sam list vor und nach der Bereitstellung, um Sie bei der lokalen und Cloud-Entwicklung zu unterstützen.

Usage

```
$ sam list <options> <subcommand>
```

Optionen

`--help, -h`

Zeige diese Nachricht und beende den Vorgang.

Unterbefehle

`endpoints`

Zeigt eine Liste der Cloud- und lokalen Endpunkte aus Ihrem CloudFormation Stack an. Weitere Informationen finden Sie unter sam list endpoints.

`resources`

Zeigt die Ressourcen in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) an, die AWS CloudFormation bei der Bereitstellung erstellt wurden. Weitere Informationen finden Sie unter sam list resources.

`stack-outputs`

Zeigt die Ausgaben Ihres CloudFormation Stacks aus einer AWS SAM CloudFormation OR-Vorlage an. Weitere Informationen finden Sie unter sam list stack-outputs.

sam list endpoints

Diese Seite enthält Referenzinformationen für den AWS Serverless Application Model `sam list endpoints` Unterbefehl Command Line Interface (AWS SAMCLI).

Eine Einführung in den finden Sie AWS SAMCLI unter Was ist der AWS SAM CLI?

Der `sam list endpoints` Unterbefehl zeigt eine Liste der Cloud- und lokalen Endpunkte aus Ihrem CloudFormation Stack an. Sie können mit diesen Ressourcen über die Befehle sam local und sam sync interagieren.

AWS Lambda und Amazon API Gateway Gateway-Ressourcentypen werden mit diesem Befehl unterstützt.

> **Note**
>
> Benutzerdefinierte Domains werden unterstützt, wenn sie für Ihre Amazon API Gateway Gateway-Ressourcen konfiguriert sind. Dieser Befehl gibt die benutzerdefinierte Domain statt des Standardendpunkts aus.

Usage

```
$ sam list endpoints <options>
```

Optionen

`--config-env` *TEXT*

Der Umgebungsname, der die Standardparameterwerte in der zu verwendenden Konfigurationsdatei angibt.

Standardwert: `default`

Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--config-file` *TEXT*

Der Pfad und der Dateiname der Konfigurationsdatei, die die zu verwendenden Standardparameterwerte enthält.

Standardwert: `samconfig.toml` im aktuellen Arbeitsverzeichnis.

Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--debug`

Schalten Sie die Debug-Protokollierung ein, um die von der generierten Debug-Meldungen AWS SAMCLI mit Zeitstempeln zu drucken.

`--help, -h`

Diese Nachricht anzeigen und beenden.

`--output [json|table]`

Geben Sie das Format für die Ausgabe der Ergebnisse an.

Standardwert: `table`

`--profile` *TEXT*

Wählen Sie ein bestimmtes Profil aus Ihrer Anmeldeinformationsdatei aus, um die AWS Anmeldeinformationen abzurufen.

`--region` *TEXT*

Stellen Sie die AWS Region des Dienstes ein. Beispiel, `us-east-1`.

`--save-params`

Speichern Sie die Parameter, die Sie in der Befehlszeile angeben, in der AWS SAM Konfigurationsdatei.

`--stack-name` *TEXT*

Name des bereitgestellten CloudFormation Stacks. Der Stack-Name befindet sich in der `samconfig.toml` Datei Ihrer Anwendung oder der angegebenen Konfigurationsdatei.

Wenn diese Option nicht angegeben ist, werden die in Ihrer Vorlage definierten lokalen Ressourcen angezeigt.

`--template-file, --template, -t` *PATH*

AWS SAM Vorlagendatei.

Standardwert: `template.[yaml|yml|json]`

Beispiele

Zeigt im JSON-Format eine Ausgabe der bereitgestellten Ressourcenendpunkte aus Ihrem CloudFormation Stack mit dem Namen `test-stack` an.

```
$ sam list endpoints --stack-name test-stack --output json

[
  {
    "LogicalResourceId": "HelloWorldFunction",
    "PhysicalResourceId": "sam-app-test-list-HelloWorldFunction-H85Y7yIV7ZLq",
    "CloudEndpoint": "https://zt55oi7kbljxjmcoahsj3cknwu0rposq.lambda-url.us-
east-1.on.aws/",
```

```
    "Methods": "-"
  },
  {
    "LogicalResourceId": "ServerlessRestApi",
    "PhysicalResourceId": "uj80uoe2o2",
    "CloudEndpoint": [
      "https://uj80uoe2o2.execute-api.us-east-1.amazonaws.com/Prod",
      "https://uj80uoe2o2.execute-api.us-east-1.amazonaws.com/Stage"
    ],
    "Methods": [
      "/hello['get']"
    ]
  }
]
```

sam list resources

Diese Seite enthält Referenzinformationen für den AWS Serverless Application Model `sam list resources` Unterbefehl Command Line Interface (AWS SAMCLI).

Eine Einführung in den finden Sie AWS SAMCLI unter Was ist der AWS SAM CLI?

Der `sam list resources` Unterbefehl zeigt die Ressourcen in Ihrer Vorlage AWS Serverless Application Model (AWS SAM) an, die AWS CloudFormation durch die AWS SAM Transformation bei der Bereitstellung erstellt wurden.

Verwenden Sie es vor der Bereitstellung sam list resources zusammen mit einer AWS SAM Vorlage, um zu sehen, welche Ressourcen erstellt werden. Geben Sie einen CloudFormation Stack-Namen ein, um eine konsolidierte Liste mit bereitgestellten Ressourcen anzuzeigen.

> **Note**
>
> Um eine Liste von Ressourcen aus Ihrer AWS SAM Vorlage zu generieren, wird eine lokale Transformation Ihrer Vorlage durchgeführt. Ressourcen, die unter bestimmten Bedingungen bereitgestellt werden, z. B. in einer bestimmten Region, sind in dieser Liste enthalten.

Usage

```
$ sam list resources <options>
```

Optionen

`--config-env` *TEXT*

Der Umgebungsname, der die Standardparameterwerte in der zu verwendenden Konfigurationsdatei angibt.

Standardwert: `default`

Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--config-file` *TEXT*

Der Pfad und der Dateiname der Konfigurationsdatei, die die zu verwendenden Standardparameterwerte enthält.

Standardwert: `samconfig.toml` im aktuellen Arbeitsverzeichnis.

Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--debug`

Schalten Sie die Debug-Protokollierung ein, um die von der generierten Debug-Meldungen AWS SAMCLI mit Zeitstempeln zu drucken.

`--help, -h`

Diese Nachricht anzeigen und beenden.

`--output [json|table]`

Geben Sie das Format für die Ausgabe der Ergebnisse an.

Standardwert: `table`

`--profile` *TEXT*

Wählen Sie ein bestimmtes Profil aus Ihrer Anmeldeinformationsdatei aus, um die AWS Anmeldeinformationen abzurufen.

`--region` *TEXT*

Stellen Sie die AWS Region des Dienstes ein. Beispiel, `us-east-1`.

`--save-params`

Speichern Sie die Parameter, die Sie in der Befehlszeile angeben, in der AWS SAM Konfigurationsdatei.

`--stack-name` *TEXT*

Name des bereitgestellten CloudFormation Stacks. Der Stack-Name befindet sich in der `samconfig.toml` Datei Ihrer Anwendung oder der angegebenen Konfigurationsdatei.

Falls angegeben, wird die logische Ressource IDs aus Ihrer Vorlage dem entsprechenden physischen IDs Eingang zugeordnet. CloudFormation Weitere Informationen zu physischen IDs Ressourcen finden Sie im AWS CloudFormation Benutzerhandbuch unter Ressourcenfelder.

Wenn diese Option nicht angegeben ist, werden die in Ihrer Vorlage definierten lokalen Ressourcen angezeigt.

`--template-file, --template, -t` *PATH*

AWS SAM Vorlagendatei.

Standardwert: `template.[yaml|yml|json]`

Beispiele

Zeigt im Tabellenformat eine Ausgabe der lokalen Ressourcen aus Ihrer AWS SAM Vorlage und der bereitgestellten Ressourcen aus Ihrem CloudFormation Stack mit dem Namen an`test-stack`. Führen Sie das Programm aus demselben Verzeichnis wie Ihre lokale Vorlage aus.

```
$ sam list resources --stack-name test-stack --output table

-------------------------------------------------------------------------------------------------
Logical ID                                                      Physical ID
-------------------------------------------------------------------------------------------------
HelloWorldFunction                                              sam-app-test-list-
HelloWorldFunction-H85Y7yIV7ZLq
HelloWorldFunctionHelloWorldPermissionProd                      sam-app-test-list-

 HelloWorldFunctionHelloWorldPermissionProd-1QH7CPOCBL2IK
HelloWorldFunctionRole                                          sam-app-test-list-
HelloWorldFunctionRole-SRJDMJ6F7F41
ServerlessRestApi                                               uj80uoe2o2
ServerlessRestApiDeployment47fc2d5f9d                           pncw5f
```

```
ServerlessRestApiProdStage                                  Prod
ServerlessRestApiDeployment5716dc08b                        -
-------------------------------------------------------------------------------------------------
```

sam list stack-outputs

Diese Seite enthält Referenzinformationen für den AWS Serverless Application Model `sam list stack-outputs` Unterbefehl Command Line Interface (AWS SAMCLI).

Eine Einführung in den finden Sie AWS SAMCLI unter Was ist der AWS SAM CLI?

Der `sam list stack-outputs` Unterbefehl zeigt die Ausgaben Ihres AWS CloudFormation Stacks aus einer AWS Serverless Application Model (AWS SAM) oder einer CloudFormation Vorlage an. Weitere Informationen dazu `Outputs` finden Sie unter Ausgaben im AWS CloudFormation Benutzerhandbuch.

Usage

```
$ sam list stack-outputs <options>
```

Optionen

`--config-env` *TEXT*

Der Umgebungsname, der die Standardparameterwerte in der zu verwendenden Konfigurationsdatei angibt.

Standardwert: `default`

Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--config-file` *TEXT*

Der Pfad und der Dateiname der Konfigurationsdatei, die die zu verwendenden Standardparameterwerte enthält.

Standardwert: `samconfig.toml` im aktuellen Arbeitsverzeichnis.

Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--debug`

Schalten Sie die Debug-Protokollierung ein, um die von der generierten Debug-Meldungen AWS SAMCLI mit Zeitstempeln zu drucken.

`--help, -h`

Diese Nachricht anzeigen und beenden.

`--output [json|table]`

Geben Sie das Format für die Ausgabe der Ergebnisse an.

Standardwert: `table`

`--profile` *TEXT*

Wählen Sie ein bestimmtes Profil aus Ihrer Anmeldeinformationsdatei aus, um die AWS Anmeldeinformationen abzurufen.

`--region` *TEXT*

Stellen Sie die AWS Region des Dienstes ein. Beispiel, `us-east-1`.

`--save-params`

Speichern Sie die Parameter, die Sie in der Befehlszeile angeben, in der AWS SAM Konfigurationsdatei.

`--stack-name` *TEXT*

Name des bereitgestellten CloudFormation Stacks. Der Stack-Name befindet sich in der `samconfig.toml` Datei Ihrer Anwendung oder der angegebenen Konfigurationsdatei.

Diese Option ist erforderlich.

Beispiele

Zeigt die Ausgaben der Ressourcen in Ihrem CloudFormation Stack mit dem Namen im Tabellenformat an`test-stack`.

```
$ sam list stack-outputs --stack-name test-stack --output table

-------------------------------------------------------------------------------------------------
OutputKey                                 OutputValue
 Description
-------------------------------------------------------------------------------------------------
```

```
HelloWorldFunctionIamRole                arn:aws:iam::account-number:role/sam-
 Implicit IAM Role created for Hello
                                         app-test-list-HelloWorldFunctionRole-    World
 function
                                         SRJDMJ6F7F41
HelloWorldApi                            https://uj80uoe2o2.execute-api.us-       API
 Gateway endpoint URL for Prod
                                         east-1.amazonaws.com/Prod/hello/         stage
 for Hello World function
HelloWorldFunction                       arn:aws:lambda:us-                       Hello
 World Lambda Function ARN
                                         east-1:account-number:function:sam-app-
                                         test-list-
                                         HelloWorldFunction-H85Y7yIV7ZLq
-----------------------------------------------------------------------------------------
```

sam local callback

Diese Seite enthält Referenzinformationen für den AWS Serverless Application Model Befehl Command Line Interface (AWS SAMCLI)`sam local callback`.

Eine Einführung in das finden AWS SAMCLI Sie unter Was ist der AWS SAM CLI?

Mit `sam local callback` diesem Befehl können Sie während der Entwicklung und beim Testen Rückrufe an lokale, dauerhafte Funktionsausführungen senden.

Usage

```
$ sam local callback <subcommand> <options>
```

Optionen

`--help, -h`

Zeige diese Nachricht und beende das Programm.

Unterbefehle

`succeed`

Sendet einen erfolgreichen Rückruf an eine dauerhafte Funktionsausführung. Weitere Informationen finden Sie unter sam local callback succeed.

`fail`

Sendet einen Fehler-Callback an eine dauerhafte Funktionsausführung. Weitere Informationen finden Sie unter sam local callback fail.

`heartbeat`

Sendet einen Heartbeat-Callback an eine dauerhafte Funktionsausführung. Weitere Informationen finden Sie unter sam local callback heartbeat.

sam local callback succeed

Sendet einen erfolgreichen Rückruf an eine dauerhafte Funktionsausführung.

Usage

```
$ sam local callback succeed CALLBACK_ID [OPTIONS]
```

Erforderliche Argumente

`CALLBACK_ID`

Die eindeutige Kennung des Rückrufs, an den eine Erfolgsantwort gesendet werden soll.

Optionen

`-r, --result TEXT`

Payload des Erfolgsergebnisses als Zeichenfolge.

`--region TEXT`

Legen Sie die AWS Region des Dienstes fest (z. B.`us-east-1`).

`--profile TEXT`

Wählen Sie ein bestimmtes Profil aus Ihrer Anmeldeinformationsdatei aus, um die AWS Anmeldeinformationen abzurufen.

`--config-env TEXT`

Umgebungsname, der die Standardparameterwerte in der Konfigurationsdatei angibt. Standard: `default`

`--config-file TEXT`

Konfigurationsdatei mit Standardparameterwerten. Standard: `samconfig.toml`

`--save-params`

Speichern Sie die über die Befehlszeile bereitgestellten Parameter in der Konfigurationsdatei.

`--beta-features / --no-beta-features`

Aktivieren/Deaktivieren von Beta-Funktionen.

`--debug`

Aktivieren Sie die Debug-Protokollierung, um von der AWS SAM CLI generierte Debug-Nachrichten zu drucken und Zeitstempel anzuzeigen.

`-h, --help`

Zeigt diese Meldung an und beendet das Programm.

Beispiele

Erfolgreicher Rückruf ohne Ergebnis senden:

```
$ sam local callback succeed my-callback-id
```

Erfolgs-Rückruf mit Ergebnis senden:

```
$ sam local callback succeed my-callback-id --result 'Task completed successfully'
```

Erfolgs-Rückruf mit Kurzoption senden:

```
$ sam local callback succeed my-callback-id -r 'Success result'
```

sam local callback fail

Sendet einen Fehler-Callback an eine dauerhafte Funktionsausführung.

Usage

```
$ sam local callback fail CALLBACK_ID [OPTIONS]
```

Erforderliche Argumente

`CALLBACK_ID`

Die eindeutige Kennung des Callbacks, an den eine Fehlerantwort gesendet werden soll.

Optionen

`--error-data TEXT`

Zusätzliche Fehlerdaten.

`--stack-trace TEXT`

Trace-Einträge stapeln. Diese Option kann mehrfach angegeben werden.

`--error-type TEXT`

Fehlertyp.

`--error-message TEXT`

Ausführliche Fehlermeldung.

`--region TEXT`

Stellen Sie die AWS Region des Dienstes ein (z. B.`us-east-1`).

`--profile TEXT`

Wählen Sie ein bestimmtes Profil aus Ihrer Anmeldeinformationsdatei aus, um die AWS Anmeldeinformationen abzurufen.

`--config-env TEXT`

Umgebungsname, der die Standardparameterwerte in der Konfigurationsdatei angibt. Standard: `default`

`--config-file TEXT`

Konfigurationsdatei mit Standardparameterwerten. Standard: `samconfig.toml`

`--save-params`

Speichern Sie die über die Befehlszeile bereitgestellten Parameter in der Konfigurationsdatei.

`--beta-features / --no-beta-features`

Aktivieren/Deaktivieren von Beta-Funktionen.

`--debug`

Aktivieren Sie die Debug-Protokollierung, um von der AWS SAM CLI generierte Debug-Nachrichten zu drucken und Zeitstempel anzuzeigen.

`-h, --help`

Zeigt diese Meldung an und beendet das Programm.

Beispiele

Fehlgeschlagener Rückruf ohne Parameter senden:

```
$ sam local callback fail my-callback-id
```

Fehlerrückruf mit Fehlermeldung senden:

```
$ sam local callback fail my-callback-id --error-message 'Task failed'
```

Fehler-Callback mit allen Parametern senden:

```
$ sam local callback fail my-callback-id --error-message 'Task failed' --error-type
 'ValidationError' --stack-trace 'at line 42' --error-data '{"code": 500}'
```

sam local callback heartbeat

Sendet einen Heartbeat-Callback an eine dauerhafte Funktionsausführung.

Usage

```
$ sam local callback heartbeat CALLBACK_ID [OPTIONS]
```

Erforderliche Argumente

`CALLBACK_ID`

Die eindeutige Kennung des Rückrufs, an den ein Heartbeat gesendet werden soll.

Optionen

`--region TEXT`

Stellen Sie die AWS Region des Dienstes ein (z. B.`us-east-1`).

`--profile TEXT`

Wählen Sie ein bestimmtes Profil aus Ihrer Anmeldeinformationsdatei aus, um die AWS Anmeldeinformationen abzurufen.

`--config-env TEXT`

Umgebungsname, der die Standardparameterwerte in der Konfigurationsdatei angibt. Standard: `default`

`--config-file TEXT`

Konfigurationsdatei mit Standardparameterwerten. Standard: `samconfig.toml`

`--save-params`

Speichern Sie die über die Befehlszeile bereitgestellten Parameter in der Konfigurationsdatei.

`--beta-features / --no-beta-features`

Aktivieren/Deaktivieren von Beta-Funktionen.

`--debug`

Aktivieren Sie die Debug-Protokollierung, um von der AWS SAM CLI generierte Debug-Nachrichten zu drucken und Zeitstempel anzuzeigen.

`-h, --help`

Zeigt diese Meldung an und beendet das Programm.

Beispiele

Heartbeat-Rückruf senden:

```
$ sam local callback heartbeat my-callback-id
```

sam local execution

Diese Seite enthält Referenzinformationen für den AWS Serverless Application Model Befehl Command Line Interface (AWS SAMCLI)`sam local execution`.

Eine Einführung in das finden AWS SAMCLI Sie unter Was ist der AWS SAM CLI?

Mit `sam local execution` diesem Befehl können Sie lokale dauerhafte Funktionsausführungen während der Entwicklung und des Testens verwalten und überprüfen.

> **Note**
>
> Für diese Befehle ist möglicherweise kein Zugriff auf AWS Anmeldeinformationen erforderlich.

Usage

```
$ sam local execution <subcommand> <options>
```

Optionen

`--help, -h`

Zeige diese Nachricht und beende das Programm.

Unterbefehle

`get`

Informieren Sie sich über die Ausführung einer dauerhaften Funktion. Weitere Informationen finden Sie unter sam local execution get.

`history`

Ruft den Ausführungsverlauf einer dauerhaften Funktionsausführung ab. Weitere Informationen finden Sie unter sam local execution history.

`stop`

Stoppt die Ausführung einer dauerhaften Funktion. Weitere Informationen finden Sie unter sam local execution stop.

sam local execution get

Rufen Sie die Details einer bestimmten dauerhaften Funktionsausführung ab.

> **Note**
>
> Für diesen Befehl ist möglicherweise kein Zugriff auf AWS Anmeldeinformationen erforderlich.

Usage

```
$ sam local execution get DURABLE_EXECUTION_ARN [OPTIONS]
```

Erforderliche Argumente

`DURABLE_EXECUTION_ARN`

Der Amazon-Ressourcenname (ARN) der dauerhaften Funktionsausführung, für die Details abgerufen werden sollen.

Optionen

`--format [summary|json]`

Ausgabeformat; Standard: `summary`

`--region TEXT`

Legen Sie die AWS Region des Dienstes fest (z. B.`us-east-1`).

`--profile TEXT`

Wählen Sie ein bestimmtes Profil aus Ihrer Anmeldeinformationsdatei aus, um die AWS Anmeldeinformationen abzurufen.

`--config-env TEXT`

Umgebungsname, der die Standardparameterwerte in der Konfigurationsdatei angibt. Standard: `default`

`--config-file TEXT`

Konfigurationsdatei mit Standardparameterwerten. Standard: `samconfig.toml`

`--save-params`

Speichern Sie die über die Befehlszeile bereitgestellten Parameter in der Konfigurationsdatei.

`--beta-features / --no-beta-features`

Aktivieren/Deaktivieren von Beta-Funktionen.

`--debug`

Aktivieren Sie die Debug-Protokollierung, um von der AWS SAM CLI generierte Debug-Nachrichten zu drucken und Zeitstempel anzuzeigen.

`-h, --help`

Zeigt diese Meldung an und wird beendet.

Beispiele

Rufen Sie die Ausführungsdetails im Standardzusammenfassungsformat ab:

```
$ sam local execution get arn:aws:lambda:us-east-1:123456789012:function:MyFunction:
$LATEST/durable-execution/my-execution-name/my-execution-id
```

Holen Sie sich Ausführungsdetails im JSON-Format:

```
$ sam local execution get arn:aws:lambda:us-east-1:123456789012:function:MyFunction:
$LATEST/durable-execution/my-execution-name/my-execution-id --format json
```

sam local execution history

Ruft den Ausführungsverlauf einer dauerhaften Funktionsausführung ab.

Usage

```
$ sam local execution history EXECUTION_ID [OPTIONS]
```

Erforderliche Argumente

`EXECUTION_ID`

Die eindeutige Kennung der dauerhaften Funktionsausführung, für die der Verlauf abgerufen werden soll.

Optionen

`--format [table|json]`

Ausgabeformat; Standard: `table`

`--region TEXT`

Legen Sie die AWS Region des Dienstes fest (z. B.`us-east-1`).

`--profile TEXT`

Wählen Sie ein bestimmtes Profil aus Ihrer Anmeldeinformationsdatei aus, um die AWS Anmeldeinformationen abzurufen.

`--config-env TEXT`

Umgebungsname, der die Standardparameterwerte in der Konfigurationsdatei angibt. Standard: `default`

`--config-file TEXT`

Konfigurationsdatei mit Standardparameterwerten. Standard: `samconfig.toml`

`--save-params`

Speichern Sie die über die Befehlszeile bereitgestellten Parameter in der Konfigurationsdatei.

`--beta-features / --no-beta-features`

Aktivieren/Deaktivieren von Beta-Funktionen.

`--debug`

Aktivieren Sie die Debug-Protokollierung, um von der AWS SAM CLI generierte Debug-Nachrichten zu drucken und Zeitstempel anzuzeigen.

`-h, --help`

Zeigt diese Meldung an und beendet das Programm.

Beispiele

Den Ausführungsverlauf im Tabellenformat abrufen:

```
$ sam local execution history my-execution-id
```

Ausführungshistorie im JSON-Format abrufen:

```
$ sam local execution history my-execution-id --format json
```

sam local execution stop

Stoppt die Ausführung einer laufenden dauerhaften Funktion.

> **Note**
>
> Für diesen Befehl ist möglicherweise kein Zugriff auf AWS Anmeldeinformationen erforderlich.

Usage

```
$ sam local execution stop DURABLE_EXECUTION_ARN [OPTIONS]
```

Erforderliche Argumente

`DURABLE_EXECUTION_ARN`

Der Amazon-Ressourcenname (ARN) der dauerhaften Funktionsausführung, die gestoppt werden soll.

Optionen

`--error-message TEXT`

Fehlermeldung, die mit der gestoppten Ausführung verknüpft werden soll.

`--error-type TEXT`

Fehlertyp, der der gestoppten Ausführung zugeordnet werden soll.

`--error-data TEXT`

Fehlerdaten, die der gestoppten Ausführung zugeordnet werden sollen.

`--stack-trace TEXT`

Trace-Einträge stapeln. Diese Option kann mehrfach angegeben werden.

`--region TEXT`

Legen Sie die AWS Region des Dienstes fest (z. B.`us-east-1`).

`--profile TEXT`

Wählen Sie ein bestimmtes Profil aus Ihrer Anmeldeinformationsdatei aus, um die AWS Anmeldeinformationen abzurufen.

`--config-env TEXT`

Umgebungsname, der die Standardparameterwerte in der Konfigurationsdatei angibt. Standard: `default`

`--config-file TEXT`

Konfigurationsdatei mit Standardparameterwerten. Standard: `samconfig.toml`

`--save-params`

Speichern Sie die über die Befehlszeile bereitgestellten Parameter in der Konfigurationsdatei.

`--beta-features / --no-beta-features`

Aktivieren/Deaktivieren von Beta-Funktionen.

`--debug`

Aktivieren Sie die Debug-Protokollierung, um von der AWS SAM CLI generierte Debug-Nachrichten zu drucken und Zeitstempel anzuzeigen.

`-h, --help`

Zeigt diese Meldung an und beendet das Programm.

Beispiele

Ausführung ohne Fehlerdetails beenden:

```
$ sam local execution stop arn:aws:lambda:us-east-1:123456789012:function:MyFunction:
$LATEST/durable-execution/my-execution-name/my-execution-id
```

Beenden Sie die Ausführung mit einer Fehlermeldung und geben Sie Folgendes ein:

```
$ sam local execution stop arn:aws:lambda:us-east-1:123456789012:function:MyFunction:
$LATEST/durable-execution/my-execution-name/my-execution-id --error-message "Execution
 cancelled" --error-type "UserCancellation"
```

Beenden Sie die Ausführung mit vollständigen Fehlerdetails und Stack-Trace:

```
$ sam local execution stop arn:aws:lambda:us-east-1:123456789012:function:MyFunction:
$LATEST/durable-execution/my-execution-name/my-execution-id --error-message "Task
 failed" --error-type "TaskFailure" --error-data '{"reason":"timeout"}' --stack-trace
 "at function1()" --stack-trace "at function2()"
```

sam local generate-event

Diese Seite enthält Referenzinformationen für den AWS Serverless Application Model Unterbefehl Command Line Interface (AWS SAMCLI)`sam local generate-event`.

- Eine Einführung in das finden Sie AWS SAMCLI unter Was ist der AWS SAM CLI?
- Eine Dokumentation zur Verwendung des AWS SAMCLI `sam local generate-event` Befehls finden Sie unterEinführung in das Testen mit sam local generate-event.

Der `sam local generate-event` Unterbefehl generiert Beispiele für Nutzdaten für Ereignisse, die unterstützt werden. AWS-Services

Usage

```
$ sam local generate-event <options> <service> <event> <event-options>
```

Optionen

`--config-env TEXT`

Der Name der Umgebung, der die Standardparameterwerte in der zu verwendenden Konfigurationsdatei angibt. Der Standardwert ist „default“. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--config-file` *`PATH`*

Der Pfad und der Dateiname der Konfigurationsdatei, die die zu verwendenden Standardparameterwerte enthält. Der Standardwert befindet sich `samconfig.toml` im

Stammverzeichnis des Projektverzeichnisses. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--help`

Zeigt diese Nachricht an und wird beendet.

Service

Führen Sie den folgenden Befehl aus, um eine Liste der unterstützten Dienste anzuzeigen:

```
$ sam local generate-event
```

Veranstaltung

Führen Sie den folgenden Befehl aus, um eine Liste der unterstützten Ereignisse anzuzeigen, die für jeden Dienst generiert werden können:

```
$ sam local generate-event <service>
```

Optionen für Ereignisse

Führen Sie den folgenden Befehl aus, um eine Liste der unterstützten Ereignisoptionen anzuzeigen, die Sie ändern können:

```
$ sam local generate-event <service> <event> --help
```

Beispiele

Beispiele zur Verwendung des `sam local generate-event` Unterbefehls finden Sie unter. Generieren Sie Beispielereignisse

sam local invoke

Diese Seite enthält Referenzinformationen für den AWS Serverless Application Model `sam local invoke` Unterbefehl Command Line Interface (AWS SAMCLI).

- Eine Einführung in den finden Sie AWS SAMCLI unter Was ist der AWS SAM CLI?
- Eine Dokumentation zur Verwendung des AWS SAMCLI `sam local invoke` Unterbefehls finden Sie unterEinführung in das Testen mit sam local invoke.

Der `sam local invoke` Unterbefehl initiiert einen einmaligen lokalen Aufruf einer Funktion. AWS Lambda

> **Note**
>
> Es wird nicht empfohlen, die lokalen Aufruffunktionen von SAM CLI in nicht vertrauenswürdigem Code zu verwenden. Um eine vollständige Isolierung von Ihrer lokalen Umgebung zu erreichen, führen Sie den Code direkt im Lambda-Service aus.

> **Note**
>
> `sam local invoke`Unterstützt für dauerhafte Funktionen die statusbehaftete Ausführung mit automatischem Checkpoint und automatischer Wiedergabe. Der Container läuft während der Ausführung dauerhafter Funktionen weiter, um die Beibehaltung und Wiederaufnahme des Zustands zu gewährleisten.

Usage

```
$ sam local invoke <arguments> <options>
```

> **Note**
>
> Wenn in Ihrer AWS SAM Vorlage mehr als eine Funktion definiert ist, geben Sie die logische ID der Funktion an, die Sie aufrufen möchten.

Argumente

Resource ID (Ressourcen-ID)

Die ID der aufzurufenden Lambda-Funktion.

Dieses Argument ist optional. Wenn Ihre Anwendung eine einzelne Lambda-Funktion enthält, wird sie von der AWS SAM CLI aufgerufen. Wenn Ihre Anwendung mehrere Funktionen enthält, geben Sie die ID der aufzurufenden Funktion an.

Gültige Werte: Die logische ID oder der Ressourcen-ARN der Ressource.

Optionen

`--add-host` *`LIST`*

Übergibt eine Zuordnung von Hostname zu IP-Adresse an die Hostdatei des Docker-Containers. Dieser Parameter kann mehrfach übergeben werden.

Example

Beispiel: `--add-host` *`example.com:127.0.0.1`*

`--beta-features | --no-beta-features`

Betafunktionen zulassen oder ablehnen.

`--config-env` *`TEXT`*

Der Umgebungsname, der die Standardparameterwerte in der zu verwendenden Konfigurationsdatei angibt. Der Standardwert ist „default“. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--config-file` *`PATH`*

Der Pfad und der Dateiname der Konfigurationsdatei, die die zu verwendenden Standardparameterwerte enthält. Der Standardwert ist "`samconfig.toml`" im Stammverzeichnis des Projektverzeichnisses. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--container-env-vars`

(Optional) Übergeben Sie beim lokalen Debuggen Umgebungsvariablen an den Image-Container der Lambda-Funktion.

`--container-host` *`TEXT`*

Host eines lokal emulierten Lambda-Containers. Der Standardwert ist `localhost`. Wenn Sie AWS SAMCLI in einem Docker-Container auf macOS laufen möchten, können Sie angeben`host.docker.internal`. Wenn Sie den Container auf einem anderen Host ausführen möchten als AWS SAMCLI, können Sie die IP-Adresse des Remote-Hosts angeben.

`--container-host-interface` *TEXT*

Die IP-Adresse der Host-Netzwerkschnittstelle, an die Container-Ports gebunden werden sollen. Der Standardwert ist `127.0.0.1`. Wird verwendet`0.0.0.0`, um an alle Schnittstellen zu binden.

`--container-dns` *TEXT*

Gibt einen benutzerdefinierten DNS-Server für den Docker-Container an. Dieser Parameter kann mehrfach übergeben werden, um mehrere DNS-Server anzugeben.

Example

Beispiel: `--container-dns` *8.8.8.8*

`--debug`

Aktiviert die Debug-Protokollierung, um die von ihm AWS SAMCLI generierten Debug-Meldungen zu drucken und Zeitstempel anzuzeigen.

`--debug-args` *TEXT*

Zusätzliche Argumente, die an den Debugger übergeben werden.

`--debug-port, -d` *TEXT*

Wenn angegeben, wird der Lambda-Funktionscontainer im Debug-Modus gestartet und dieser Port auf dem lokalen Host verfügbar gemacht.

`--debugger-path` *TEXT*

Der Hostpfad zu einem Debugger, der in den Lambda-Container gemountet ist.

`--docker-network` *TEXT*

Der Name oder die ID eines vorhandenen Docker-Netzwerks, mit dem Lambda-Docker-Container eine Verbindung herstellen sollen, zusammen mit dem Standard-Bridge-Netzwerk. Wenn dies nicht angegeben ist, stellen die Lambda-Container nur eine Verbindung zum Standard-Bridge-Docker-Netzwerk her.

`--docker-volume-basedir, -v` *TEXT*

Der Speicherort des Basisverzeichnisses, in dem die AWS SAM Datei existiert. Wenn Docker auf einem Remote-Computer ausgeführt wird, müssen Sie den Pfad, in dem sich die AWS SAM Datei befindet, auf dem Docker-Computer bereitstellen und diesen Wert so ändern, dass er mit dem Remote-Computer übereinstimmt.

`--durable-execution-name` *TEXT*

Name für die dauerhafte Ausführung (nur für dauerhafte Funktionen)

`--env-vars, -n` *PATH*

Die `.env` JSON-Datei oder die Datei, die Werte für die Umgebungsvariablen der Lambda-Funktion enthält. Das Dateiformat wird automatisch erkannt. Weitere Hinweise zu Umgebungsvariablendateien finden Sie unterUmgebungsvariablendatei.

`--event, -e` *PATH*

Die JSON-Datei, die Ereignisdaten enthält, die an die Lambda-Funktion übergeben werden, wenn sie aufgerufen wird. Wenn Sie diese Option nicht angeben, wird kein Ereignis angenommen. Um JSON von einzugeben`stdin`, müssen Sie den Wert '-' übergeben. Einzelheiten zu den Formaten von Ereignisnachrichten verschiedener AWS Dienste finden Sie unter Arbeiten mit anderen Diensten im AWS Lambda Entwicklerhandbuch.

`--force-image-build`

Gibt an, ob das Image, das zum Aufrufen von Lambda-Funktionen mit Ebenen verwendet wird, neu erstellt werden AWS SAMCLI soll.

`--help`

Zeigt diese Meldung an und wird beendet.

`--hook-name TEXT`

Der Name des Hooks, der zur Erweiterung der AWS SAMCLI Funktionalität verwendet wird.

Zulässige Werte:`terraform`.

`--invoke-image` *TEXT*

Die URI des Container-Images, das Sie für den lokalen Funktionsaufruf verwenden möchten.

Für Lambda-Funktionen mit `PackageType: Zip` überschreibt dies das Runtime-Basis-Image. Ruft standardmäßig das Container-Image aus Amazon ECR Public ab (die unter aufgeführt sind). AWS SAM Bildrepositorien für AWS SAM Verwenden Sie diese Option, um das Bild von einem anderen Speicherort abzurufen.

Für Lambda-Funktionen mit `PackageType: Image` überschreibt dies die `ImageUri` aus der erstellten Vorlage. Verwenden Sie dies, um lokal ein anderes Anwendungs-Image aufzurufen, ohne Ihre Vorlage zu ändern.

Sie können diese Option mehrfach angeben. Jede Instanz dieser Option kann entweder eine Zeichenfolge oder ein Schlüssel-Wert-Paar annehmen. Wenn Sie eine Zeichenfolge angeben, ist dies die URI des Container-Images, das für alle Funktionen in Ihrer Anwendung verwendet werden soll. Beispiel, `sam local invoke --invoke-image public.ecr.aws/sam/emu-python3.13`. Wenn Sie ein Schlüssel-Wert-Paar angeben, ist der Schlüssel der Ressourcenname und der Wert ist der URI des Container-Images, das für diese Ressource verwendet werden soll. Beispiel, `sam local invoke --invoke-image public.ecr.aws/sam/emu-python3.13 --invoke-image Function1=amazon/aws-sam-cli-emulation-image-python3.13`. Mit Schlüssel-Wert-Paaren können Sie verschiedene Container-Images für verschiedene Ressourcen angeben.

`--layer-cache-basedir` *DIRECTORY*

Gibt den Speicherort des Basisverzeichnisses an, in das die von Ihrer Vorlage verwendeten Ebenen heruntergeladen werden.

`--log-file, -l` *TEXT*

Die Protokolldatei, an die Laufzeitprotokolle gesendet werden sollen.

`--mount-symlinks`

Stellt sicher, dass die Symlinks, die in den zu erstellenden oder aufzurufenden Dateien vorhanden sind, AWS SAMCLI immer eingehängt werden. Dies gilt nur für symbolische Links im Verzeichnis der obersten Ebene (d. h. für symbolische Links, die sich direkt im Stammverzeichnis der Funktion befinden). Standardmäßig werden Symlinks nicht gemountet, außer denen, die für die Verwendung `node_modules` in `build-in-source` NodeJS benötigt werden.

`--no-event`

Ruft die Funktion mit einem leeren Ereignis auf.

`--no-memory-limit`

Beseitigt die Speicherbeschränkung im Container beim lokalen Aufrufen, auch wenn Speicher in der AWS SAM Vorlage konfiguriert ist.

`--parameter-overrides`

Eine Zeichenfolge, die CloudFormation Parameterüberschreibungen enthält, die als Schlüssel-Wert-Paare codiert sind. Verwenden Sie dasselbe Format wie (). AWS Command Line Interface AWS CLI Das AWS SAMCLI Format besteht aus expliziten Schlüsselwörtern und Werten, wobei jede Überschreibung durch ein Leerzeichen getrennt ist. Nachfolgend finden Sie zwei Beispiele:

- `--parameter-overrides ParameterKey=hello,ParameterValue=world`
- `--parameter-overrides ParameterKey=hello,ParameterValue=world ParameterKey=example1,ParameterValue=example2 ParameterKey=apple,ParameterValue=banana`

`--profile` *TEXT*

Das spezifische Profil aus Ihrer Anmeldeinformationsdatei, das die AWS Anmeldeinformationen abruft.

`--region` *TEXT*

Die AWS Region, in der die Bereitstellung erfolgen soll. Beispiel: us-east-1.

`--runtime` *TEXT*

Verwendet die angegebene Laufzeit, um eine Lambda-Funktion lokal aufzurufen. Dies überschreibt die in der Datei definierte Laufzeit. `template.yml` Dies ermöglicht auch das Testen von Lambda-Funktionen mit unterschiedlichen Laufzeiten, ohne die ursprüngliche Funktionskonfiguration zu ändern.

`--save-params`

Speichern Sie die Parameter, die Sie in der Befehlszeile angeben, in der AWS SAM Konfigurationsdatei.

`--shutdown`

Emuliert nach Abschluss des Aufrufs ein Shutdown-Ereignis, um zu testen, wie Erweiterungen das Shutdown-Verhalten behandeln.

`--skip-prepare-infra`

Überspringt die Vorbereitungsphase, wenn keine Änderungen an der Infrastruktur vorgenommen wurden. Zusammen mit der `--hook-name` Option verwenden.

`--skip-pull-image`

Standardmäßig AWS SAMCLI überprüft das die neueste Remote-Laufzeitumgebung von Lambda und aktualisiert Ihr lokales Image automatisch, damit es synchron bleibt.

Geben Sie diese Option an, um das Herunterladen des neuesten Docker Images für Ihre Lambda-Laufzeitumgebung zu überspringen.

`--template, -t` *PATH*

Die AWS SAM Vorlagendatei.

Diese Option ist nicht kompatibel mit`--hook-name`.

> **Note**
>
> Wenn Sie diese Option angeben, werden nur die Vorlage und die lokalen Ressourcen AWS SAM geladen, auf die sie verweist.

`--tenant-id` *TEXT*

Die Mandanten-ID für Lambda-Funktionen mit mehreren Mandanten. Wird verwendet, um die Rechenisolierung zwischen verschiedenen Mandanten sicherzustellen. Erforderlich beim Aufrufen von Funktionen, die mit dem Mandantenisolationsmodus konfiguriert sind.

`--terraform-plan-file`

Der relative oder absolute Pfad zu Ihrer lokalen Terraform Plandatei, wenn Sie with verwenden. AWS SAMCLI Terraform Cloud Für diese Option muss der `--hook-name` Wert auf gesetzt sein`terraform`.

Beispiele

Im folgenden Beispiel wird ein generiertes Ereignis für lokale Tests verwendet, indem ein `s3.json` Ereignis verwendet wird, um eine Lambda-Funktion lokal aufzurufen.

```
$ sam local invoke --event events/s3.json S3JsonLoggerFunction
```

Das folgende Beispiel testet die Funktion `HelloWorldFunction` mit der Python 3.11-Laufzeit

```
$ sam local invoke --runtime python3.11 HelloWorldFunction
```

Im folgenden Beispiel wird die Funktion `HelloWorldFunction` mit einem dauerhaften Ausführungsnamen getestet

```
$ sam local invoke HelloWorldFunction --durable-execution-name my-execution
```

Sam lokale Start-API

Diese Seite enthält Referenzinformationen für den AWS Serverless Application Model Unterbefehl Command Line Interface (AWS SAMCLI)`sam local start-api`.

- Eine Einführung in das finden Sie AWS SAMCLI unter Was ist der AWS SAM CLI?
- Eine Dokumentation zur Verwendung des AWS SAMCLI `sam local start-api` Unterbefehls finden Sie unterEinführung in das Testen mit dieselbe lokale Start-API.

Der `sam local start-api` Unterbefehl führt Ihre AWS Lambda Funktionen lokal aus, um sie über einen lokalen HTTP-Serverhost zu testen.

Usage

```
$ sam local start-api <options>
```

Optionen

`--add-host` *`LIST`*

Übergibt eine Zuordnung von Hostname zu IP-Adresse an die Hostdatei des Docker-Containers. Dieser Parameter kann mehrfach übergeben werden.

Example

Beispiel: `--add-host` *`example.com:127.0.0.1`*

`--beta-features | --no-beta-features`

Betafunktionen zulassen oder ablehnen.

`--config-env` *`TEXT`*

Der Umgebungsname, der die Standardparameterwerte in der zu verwendenden Konfigurationsdatei angibt. Der Standardwert ist „default“. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--config-file` *`PATH`*

Der Pfad und der Dateiname der Konfigurationsdatei, die die zu verwendenden Standardparameterwerte enthält. Der Standardwert ist „samconfig.toml“ im Stammverzeichnis

des Projektverzeichnisses. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--container-env-vars`

Optional. Übergeben Sie beim lokalen Debuggen Umgebungsvariablen an den Image-Container.

`--container-host` *TEXT*

Host eines lokal emulierten Lambda-Containers. Der Standardwert ist `localhost`. Wenn Sie AWS SAMCLI in einem Docker-Container auf macOS laufen möchten, können Sie angeben`host.docker.internal`. Wenn Sie den Container auf einem anderen Host ausführen möchten als AWS SAMCLI, können Sie die IP-Adresse des Remote-Hosts angeben.

`--container-host-interface` *TEXT*

Die IP-Adresse der Host-Netzwerkschnittstelle, an die Container-Ports gebunden werden sollen. Der Standardwert ist `127.0.0.1`. Wird verwendet`0.0.0.0`, um an alle Schnittstellen zu binden.

`--container-dns` *TEXT*

Gibt einen benutzerdefinierten DNS-Server für den Docker-Container an. Dieser Parameter kann mehrfach übergeben werden, um mehrere DNS-Server anzugeben.

Example

Beispiel: `--container-dns` *8.8.8.8*

`--debug`

Aktiviert die Debug-Protokollierung, um die von den generierten Debug-Nachrichten zu drucken AWS SAMCLI und Zeitstempel anzuzeigen.

`--debug-args` *TEXT*

Zusätzliche Argumente, die an den Debugger übergeben werden müssen.

`--debug-function`

Optional. Gibt die Lambda-Funktion an, auf die Debug-Optionen angewendet werden sollen, wenn sie angegeben `--warm-containers` ist. Dieser Parameter gilt für `--debug-port`-- `debugger-path`, und. `--debug-args`

`--debug-port, -d` *TEXT*

Wenn angegeben, wird der Lambda-Funktionscontainer im Debug-Modus gestartet und dieser Port auf dem lokalen Host verfügbar gemacht.

`--debugger-path` *TEXT*

Der Hostpfad zu einem Debugger, der in den Lambda-Container gemountet wird.

`--docker-network` *TEXT*

Der Name oder die ID eines vorhandenen Docker-Netzwerks, mit dem sich die Lambda-Docker-Container verbinden sollen, zusammen mit dem Standard-Bridge-Netzwerk. Wenn dies nicht angegeben ist, stellen die Lambda-Container nur eine Verbindung zum Standard-Bridge-Docker-Netzwerk her.

`--docker-volume-basedir, -v` *TEXT*

Der Speicherort des Basisverzeichnisses, in dem die AWS SAM Datei existiert. Wenn Docker auf einem Remote-Computer ausgeführt wird, müssen Sie den Pfad, in dem sich die AWS SAM Datei befindet, auf dem Docker-Computer bereitstellen und diesen Wert so ändern, dass er mit dem Remote-Computer übereinstimmt.

`--env-vars, -n` *PATH*

Die `.env` JSON-Datei oder die Datei, die Werte für die Umgebungsvariablen der Lambda-Funktion enthält. Das Dateiformat wird automatisch erkannt.

`--force-image-build`

Gibt an, ob das Bild, das zum Aufrufen von Funktionen mit Ebenen verwendet wurde, neu erstellt werden AWS SAM CLI soll.

`--help`

Zeigt diese Meldung an und wird beendet.

`--hook-name` *TEXT*

Der Name des Hooks, der zur Erweiterung der AWS SAMCLI Funktionalität verwendet wird.

Zulässige Werte:`terraform`.

`--host` *TEXT*

Der lokale Hostname oder die IP-Adresse, an die gebunden werden soll (Standard: '127.0.0.1').

`--invoke-image` *TEXT*

Die URI des Container-Images, das Sie für Ihre Lambda-Funktionen verwenden möchten.

Für Lambda-Funktionen mit `PackageType: Zip` überschreibt dies das Runtime-Basis-Image. Ruft das Container-Image standardmäßig aus Amazon ECR AWS SAM Public ab. Verwenden Sie diese Option, um das Bild von einem anderen Speicherort abzurufen.

Für Lambda-Funktionen mit `PackageType: Image` überschreibt dies die `ImageUri` aus der erstellten Vorlage. Verwenden Sie dies, um lokal ein anderes Anwendungs-Image aufzurufen, ohne Ihre Vorlage zu ändern.

Sie können diese Option mehrfach angeben. Jede Instanz dieser Option kann entweder eine Zeichenfolge oder ein Schlüssel-Wert-Paar annehmen. Wenn Sie eine Zeichenfolge angeben, ist dies die URI des Container-Images, das für alle Funktionen in Ihrer Anwendung verwendet werden soll. Beispiel, `sam local start-api --invoke-image public.ecr.aws/sam/emu-python3.13`. Wenn Sie ein Schlüssel-Wert-Paar angeben, ist der Schlüssel der Ressourcenname und der Wert ist der URI des Container-Images, das für diese Ressource verwendet werden soll. Beispiel, `sam local start-api --invoke-image public.ecr.aws/sam/emu-python3.13 --invoke-image Function1=amazon/aws-sam-cli-emulation-image-python3.13`. Mit Schlüssel-Wert-Paaren können Sie verschiedene Container-Images für verschiedene Ressourcen angeben.

`--layer-cache-basedir` *DIRECTORY*

Gibt den Speicherort an, in den die von Ihrer Vorlage verwendeten Ebenen heruntergeladen werden.

`--log-file, -l` *TEXT*

Die Protokolldatei, an die Laufzeitprotokolle gesendet werden sollen.

`--no-memory-limit`

Beseitigt die Speicherbeschränkung im Container beim lokalen Aufruf, auch wenn Speicher in der AWS SAM Vorlage konfiguriert ist.

`--no-watch`

Deaktiviert die Dateiüberwachung bei Verwendung mit. `--warm-containers` Wenn angegeben, überwacht der Ihren Quellcode oder Ihre Vorlage AWS SAMCLI nicht auf Änderungen, und warme Container werden bei Codeänderungen nicht automatisch neu gestartet. Um lokale Änderungen zu übernehmen, beenden Sie den Befehl und führen Sie ihn erneut aus.

Verwenden Sie diese Option, wenn die Dateiüberwachung einen hohen CPU- oder I/O Overhead verursacht, z. B. in großen Monorepos, Umgebungen mit Antiviren-Software-Scans oder Projekten, in denen mehrere Lambda-Funktionen sich eine teilen. `CodeUri`

Diese Option wird ignoriert, wenn sie nicht angegeben `--warm-containers` ist.

`--parameter-overrides`

Eine Zeichenfolge, die CloudFormation Parameterüberschreibungen enthält, die als Schlüssel-Wert-Paare codiert sind. Verwenden Sie dasselbe Format wie (). AWS Command Line Interface AWS CLI Das AWS SAMCLI Format besteht aus expliziten Schlüsselwörtern und Werten, wobei jede Überschreibung durch ein Leerzeichen getrennt ist. Nachfolgend finden Sie zwei Beispiele:

- `--parameter-overrides ParameterKey=hello,ParameterValue=world`
- `--parameter-overrides ParameterKey=hello,ParameterValue=world ParameterKey=example1,ParameterValue=example2 ParameterKey=apple,ParameterValue=banana`

`--port, -p` *`INTEGER`*

Die lokale Portnummer, auf der gewartet werden soll (Standard: '3000').

`--profile` *`TEXT`*

Das spezifische Profil aus Ihrer Anmeldeinformationsdatei, das die AWS Anmeldeinformationen abruft.

`--region` *`TEXT`*

Die AWS Region, in der die Bereitstellung erfolgen soll. Beispiel: us-east-1.

`--save-params`

Speichern Sie die Parameter, die Sie in der Befehlszeile angeben, in der AWS SAM Konfigurationsdatei.

`--shutdown`

Emuliert nach Abschluss des Aufrufs ein Shutdown-Ereignis, um zu testen, wie Erweiterungen das Shutdown-Verhalten behandeln.

`--skip-prepare-infra`

Überspringt die Vorbereitungsphase, wenn keine Änderungen an der Infrastruktur vorgenommen wurden. Zusammen mit der `--hook-name` Option verwenden.

`--skip-pull-image`

Gibt an, ob die CLI das Abrufen des neuesten Docker-Images für die Lambda-Laufzeit überspringen soll.

`--ssl-cert-file` *PATH*

Pfad zur SSL-Zertifikatsdatei (Standard: Keine). Wenn Sie diese Option verwenden, muss die `--ssl-key-file` Option auch verwendet werden.

`--ssl-key-file` *PATH*

Pfad zur SSL-Schlüsseldatei (Standard: Keine). Wenn Sie diese Option verwenden, muss die `--ssl-cert-file` Option auch verwendet werden.

`--static-dir, -s` *TEXT*

Alle statischen Asset-Dateien (z. B. CSS/JavaScript/HTML), die sich in diesem Verzeichnis befinden, werden unter angezeigt/.

`--template, -t` *PATH*

Die AWS SAM Vorlagendatei.

> **Note**
>
> Wenn Sie diese Option angeben, werden nur die Vorlage und die lokalen Ressourcen AWS SAM geladen, auf die sie verweist.

`--terraform-plan-file`

Der relative oder absolute Pfad zu Ihrer lokalen Terraform Plandatei, wenn Sie AWS SAMCLI with verwendenTerraform Cloud. Für diese Option muss der `--hook-name` Wert auf gesetzt seinterraform.

`--warm-containers` *[EAGER | LAZY]*

Optional. Gibt an, wie Container für jede Funktion AWS SAMCLI verwaltet werden.

Zwei Optionen sind verfügbar:

`EAGER`: Container für alle Funktionen werden beim Start geladen und bleiben zwischen Aufrufen bestehen.

`LAZY`: Container werden nur geladen, wenn jede Funktion zum ersten Mal aufgerufen wird. Diese Container bleiben für weitere Aufrufe bestehen.

Beispiel

Im folgenden Beispiel wird ein lokaler Server gestartet, sodass Sie Ihre Anwendung über die API testen können. Damit dieser Befehl funktioniert, muss die Anwendung installiert sein und Docker muss ausgeführt werden.

```
$ sam local start-api --port 3000
```

Sam Local Start-Lambda

Diese Seite enthält Referenzinformationen für den AWS Serverless Application Model Unterbefehl Command Line Interface ()AWS SAMCLI. `sam local start-lambda`

- Eine Einführung in das finden Sie AWS SAMCLI unter Was ist der AWS SAM CLI?
- Eine Dokumentation zur Verwendung des AWS SAMCLI `sam local start-lambda` Unterbefehls finden Sie unterEinführung in das Testen mit sam local start-lambda.

Der `sam local start-lambda` Unterbefehl startet einen lokalen Endpunkt, der emuliert werden soll. AWS Lambda

Usage

```
$ sam local start-lambda <options>
```

Optionen

`--add-host` *`LIST`*

Übergibt eine Zuordnung von Hostname zu IP-Adresse an die Hostdatei des Docker-Containers. Dieser Parameter kann mehrfach übergeben werden.

Example

Beispiel: `--add-host` *`example.com:127.0.0.1`*

`--beta-features | --no-beta-features`

Betafunktionen zulassen oder ablehnen.

`--config-env` *TEXT*

Der Umgebungsname, der die Standardparameterwerte in der zu verwendenden Konfigurationsdatei angibt. Der Standardwert ist „default“. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--config-file` *PATH*

Der Pfad und der Dateiname der Konfigurationsdatei, die die zu verwendenden Standardparameterwerte enthält. Der Standardwert ist „samconfig.toml“ im Stammverzeichnis des Projektverzeichnisses. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--container-env-vars`

Optional. Übergeben Sie beim lokalen Debuggen Umgebungsvariablen an den Image-Container.

`--container-host` *TEXT*

Host eines lokal emulierten Lambda-Containers. Der Standardwert ist `localhost`. Wenn Sie AWS SAMCLI in einem Docker-Container auf macOS laufen möchten, können Sie angeben`host.docker.internal`. Wenn Sie den Container auf einem anderen Host ausführen möchten als AWS SAMCLI, können Sie die IP-Adresse des Remote-Hosts angeben.

`--container-host-interface` *TEXT*

Die IP-Adresse der Host-Netzwerkschnittstelle, an die Container-Ports gebunden werden sollen. Der Standardwert ist `127.0.0.1`. Wird verwendet`0.0.0.0`, um an alle Schnittstellen zu binden.

`--container-dns` *TEXT*

Gibt einen benutzerdefinierten DNS-Server für den Docker-Container an. Dieser Parameter kann mehrfach übergeben werden, um mehrere DNS-Server anzugeben.

Example

Beispiel: `--container-dns` *8.8.8.8*

`--debug`

Aktiviert die Debug-Protokollierung, um die von den generierten Debug-Nachrichten zu drucken AWS SAMCLI und Zeitstempel anzuzeigen.

`--debug-args` *TEXT*

Zusätzliche Argumente, die an den Debugger übergeben werden müssen.

`--debug-function`

Optional. Gibt die Lambda-Funktion an, auf die Debug-Optionen angewendet werden sollen, wenn sie angegeben `--warm-containers` ist. Dieser Parameter gilt für `--debug-port--debugger-path`, und. `--debug-args`

`--debug-port, -d` *TEXT*

Wenn angegeben, wird der Lambda-Funktionscontainer im Debug-Modus gestartet und dieser Port auf dem lokalen Host verfügbar gemacht.

`--debugger-path` *TEXT*

Der Hostpfad zu einem Debugger, der in den Lambda-Container gemountet werden soll.

`--docker-network` *TEXT*

Der Name oder die ID eines vorhandenen Docker-Netzwerks, mit dem Lambda-Docker-Container eine Verbindung herstellen sollen, zusammen mit dem Standard-Bridge-Netzwerk. Wenn dies angegeben ist, stellen die Lambda-Container nur eine Verbindung zum Standard-Bridge-Docker-Netzwerk her.

`--docker-volume-basedir, -v` *TEXT*

Der Speicherort des Basisverzeichnisses, in dem die AWS SAM Datei existiert. Wenn Docker auf einem Remote-Computer ausgeführt wird, müssen Sie den Pfad, in dem sich die AWS SAM Datei befindet, auf dem Docker-Computer bereitstellen und diesen Wert so ändern, dass er mit dem Remote-Computer übereinstimmt.

`--env-vars, -n` *PATH*

Die `.env` JSON-Datei oder die Datei, die Werte für die Umgebungsvariablen der Lambda-Funktion enthält. Das Dateiformat wird automatisch erkannt.

`--force-image-build`

Geben Sie an, ob das Bild, das zum Aufrufen von Funktionen mit Ebenen verwendet wurde, neu erstellt werden CLI soll.

`--help`

Zeigt diese Meldung an und wird beendet.

`--hook-name` *TEXT*

Der Name des Hooks, der zur Erweiterung der AWS SAMCLI Funktionalität verwendet wird.

Zulässige Werte:`terraform`.

`--host` *TEXT*

Der lokale Hostname oder die IP-Adresse, an die gebunden werden soll (Standard: '127.0.0.1').

`--invoke-image` *TEXT*

Die URI des Container-Images, das Sie für den lokalen Funktionsaufruf verwenden möchten.

Für Lambda-Funktionen mit `PackageType: Zip` überschreibt dies das Runtime-Basis-Image. Ruft das Container-Image standardmäßig aus Amazon ECR AWS SAM Public ab. Verwenden Sie diese Option, um das Bild von einem anderen Speicherort abzurufen.

Für Lambda-Funktionen mit `PackageType: Image` überschreibt dies die `ImageUri` aus der erstellten Vorlage. Verwenden Sie dies, um lokal ein anderes Anwendungs-Image aufzurufen, ohne Ihre Vorlage zu ändern.

Sie können diese Option mehrfach angeben. Jede Instanz dieser Option kann entweder eine Zeichenfolge oder ein Schlüssel-Wert-Paar annehmen. Wenn Sie eine Zeichenfolge angeben, ist dies die URI des Container-Images, das für alle Funktionen in Ihrer Anwendung verwendet werden soll. Beispiel, `sam local start-lambda --invoke-image public.ecr.aws/sam/emu-python3.13`. Wenn Sie ein Schlüssel-Wert-Paar angeben, ist der Schlüssel der Ressourcenname und der Wert ist der URI des Container-Images, das für diese Ressource verwendet werden soll. Beispiel, `sam local start-lambda --invoke-image public.ecr.aws/sam/emu-python3.13 --invoke-image Function1=amazon/aws-sam-cli-emulation-image-python3.13`. Mit Schlüssel-Wert-Paaren können Sie verschiedene Container-Images für verschiedene Ressourcen angeben.

`--layer-cache-basedir` *DIRECTORY*

Gibt den Speicherort an, in den die von Ihrer Vorlage verwendeten Ebenen heruntergeladen werden.

`--log-file, -l` *TEXT*

Die Protokolldatei, an die Laufzeitprotokolle gesendet werden sollen.

`--no-memory-limit`

Beseitigt die Speicherbeschränkung im Container beim lokalen Aufruf, auch wenn Speicher in der AWS SAM Vorlage konfiguriert ist.

`--no-watch`

Deaktiviert die Dateiüberwachung, wenn es mit verwendet wird. `--warm-containers` Wenn angegeben, überwacht der Ihren Quellcode oder Ihre Vorlage AWS SAMCLI nicht auf Änderungen, und warme Container werden bei Codeänderungen nicht automatisch neu gestartet. Um lokale Änderungen zu übernehmen, beenden Sie den Befehl und führen Sie ihn erneut aus.

Verwenden Sie diese Option, wenn die Dateiüberwachung einen hohen CPU- oder I/O Overhead verursacht, z. B. in großen Monorepos, Umgebungen mit Antiviren-Software-Scans oder Projekten, in denen mehrere Lambda-Funktionen sich eine teilen. `CodeUri`

Diese Option wird ignoriert, wenn sie nicht angegeben `--warm-containers` ist.

`--parameter-overrides`

Eine Zeichenfolge, die CloudFormation Parameterüberschreibungen enthält, die als Schlüssel-Wert-Paare codiert sind. Verwenden Sie dasselbe Format wie (). AWS Command Line Interface AWS CLI Das AWS SAMCLI Format besteht aus expliziten Schlüsselwörtern und Werten, wobei jede Überschreibung durch ein Leerzeichen getrennt ist. Nachfolgend finden Sie zwei Beispiele:

- `--parameter-overrides ParameterKey=hello,ParameterValue=world`
- `--parameter-overrides ParameterKey=hello,ParameterValue=world ParameterKey=example1,ParameterValue=example2 ParameterKey=apple,ParameterValue=banana`

`--port, -p` *`INTEGER`*

Die lokale Portnummer, auf der gewartet werden soll (Standard: '3001').

`--profile` *`TEXT`*

Das spezifische Profil aus Ihrer Anmeldeinformationsdatei, das die Anmeldeinformationen abruft AWS .

`--region` *`TEXT`*

Die AWS Region, in der die Bereitstellung erfolgen soll. Beispiel: us-east-1.

`--save-params`

Speichern Sie die Parameter, die Sie in der Befehlszeile angeben, in der AWS SAM Konfigurationsdatei.

`--shutdown`

Emuliert nach Abschluss des Aufrufs ein Shutdown-Ereignis, um zu testen, wie Erweiterungen das Shutdown-Verhalten behandeln.

`--skip-prepare-infra`

Überspringt die Vorbereitungsphase, wenn keine Änderungen an der Infrastruktur vorgenommen wurden. Zusammen mit der `--hook-name` Option verwenden.

`--skip-pull-image`

Gibt an, ob das Abrufen des neuesten Docker-Images für die Lambda-Laufzeit übersprungen werden CLI soll.

`--template, -t` *`PATH`*

Die AWS SAM Vorlagendatei.

> **Note**
>
> Wenn Sie diese Option angeben, werden nur die Vorlage und die lokalen Ressourcen AWS SAM geladen, auf die sie verweist. Diese Option ist nicht kompatibel mit`--hook-name`.

`--terraform-plan-file`

Der relative oder absolute Pfad zu Ihrer lokalen Terraform Plandatei, wenn Sie AWS SAMCLI with verwendenTerraform Cloud. Für diese Option muss der `--hook-name` Wert auf gesetzt seinterraform.

`--warm-containers` *`[EAGER | LAZY]`*

Optional. Gibt an, wie Container für jede Funktion AWS SAMCLI verwaltet werden.

Zwei Optionen sind verfügbar:

- `EAGER`: Container für alle Funktionen werden beim Start geladen und bleiben zwischen Aufrufen bestehen.
- `LAZY`: Container werden nur geladen, wenn jede Funktion zum ersten Mal aufgerufen wird. Diese Container bleiben für weitere Aufrufe bestehen.

Beispiel

Ein ausführliches Beispiel und eine ausführliche Anleitung zur Verwendung des `sam local start-lambda` Unterbefehls finden Sie unter. Einführung in das Testen mit sam local start-lambda

sam logs

Diese Seite enthält Referenzinformationen für den AWS Serverless Application Model Befehl Command Line Interface (AWS SAMCLI)`sam logs`.

Eine Einführung in den finden AWS SAMCLI Sie unter Was ist der AWS SAM CLI?

Der `sam logs` Befehl ruft Protokolle ab, die von Ihren AWS Lambda Funktionen generiert wurden.

Usage

```
$ sam logs <options>
```

Optionen

`--config-env` *TEXT*

Der Umgebungsname, der die Standardparameterwerte in der zu verwendenden Konfigurationsdatei angibt. Der Standardwert ist „default“. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--config-file` *PATH*

Der Pfad und der Dateiname der Konfigurationsdatei, die die zu verwendenden Standardparameterwerte enthält. Der Standardwert ist „samconfig.toml“ im Stammverzeichnis des Projektverzeichnisses. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--cw-log-group` *LIST*

Schließt Protokolle aus den von Ihnen angegebenen CloudWatch Logs-Protokollgruppen ein. Wenn Sie diese Option zusammen mit angebenname, AWS SAM werden zusätzlich zu den Protokollen der genannten Ressourcen auch Protokolle aus den angegebenen Protokollgruppen eingeschlossen.

`--debug`

Aktiviert die Debug-Protokollierung, um die von den generierten Debug-Nachrichten zu drucken AWS SAMCLI und Zeitstempel anzuzeigen.

`---end-time, e` *TEXT*

Ruft Protokolle bis zu diesem Zeitpunkt ab. Bei der Uhrzeit kann es sich um relative Werte wie „vor 5 Minuten“, „Morgen“ oder um einen formatierten Zeitstempel wie „2018-01-01 10:10:10“ handeln.

`--filter` *TEXT*

Ermöglicht die Angabe eines Ausdrucks, um schnell Logs zu finden, die Begriffen, Ausdrücken oder Werten in Ihren Protokollereignissen entsprechen. Dies kann ein einfaches Schlüsselwort (z. B. „Fehler“) oder ein Muster sein, das von Amazon CloudWatch Logs unterstützt wird. Die Syntax finden Sie in der Amazon CloudWatch Logs-Dokumentation.

`--help`

Zeigt diese Meldung an und wird beendet.

`--include-traces`

Schließt Röntgenspuren in die Protokollausgabe ein.

`--name, -n` *TEXT*

Der Name der Ressource, für die Protokolle abgerufen werden sollen. Wenn diese Ressource Teil eines CloudFormation Stacks ist, kann dies die logische ID der Funktionsressource in der AWS SAM Vorlage CloudFormation/sein. Durch erneutes Wiederholen des Parameters können mehrere Namen angegeben werden. Wenn sich die Ressource in einem verschachtelten Stapel befindet, kann dem Namen der Name des geschachtelten Stacks vorangestellt werden, um Protokolle von dieser Ressource abzurufen (/). NestedStackLogicalId ResourceLogicalId Wenn der Ressourcenname nicht angegeben ist, wird der angegebene Stapel gescannt und es werden Protokollinformationen für alle unterstützten Ressourcen abgerufen. Wenn Sie diese Option nicht angeben, werden Protokolle für alle Ressourcen im AWS SAM Stapel abgerufen, die Sie angeben. Die folgenden Ressourcentypen werden unterstützt:

- `AWS::Serverless::Function`
- `AWS::Lambda::Function`
- `AWS::Serverless::Api`
- `AWS::ApiGateway::RestApi`

- `AWS::Serverless::HttpApi`
- `AWS::ApiGatewayV2::Api`
- `AWS::Serverless::StateMachine`
- `AWS::StepFunctions::StateMachine`

`--output` *TEXT*

Gibt das Ausgabeformat für Protokolle an. Um formatierte Protokolle zu drucken, geben Sie an`text`. Um die Protokolle als JSON zu drucken, geben Sie an`json`.

`--profile` *TEXT*

Das spezifische Profil aus Ihrer Anmeldeinformationsdatei, das die AWS Anmeldeinformationen abruft.

`--region` *TEXT*

Die AWS Region, in der die Bereitstellung erfolgen soll. Beispiel: us-east-1.

`--save-params`

Speichern Sie die Parameter, die Sie in der Befehlszeile angeben, in der AWS SAM Konfigurationsdatei.

`--stack-name` *TEXT*

Der Name des CloudFormation Stacks, zu dem die Ressource gehört.

`--start-time, -s` *TEXT*

Ruft Logs ab diesem Zeitpunkt ab. Bei der Uhrzeit kann es sich um relative Werte wie „vor 5 Minuten“, „Gestern“ oder um einen formatierten Zeitstempel wie „2018-01-01 10:10:10“ handeln. Der Standardwert ist 'vor 10 Minuten'.

`--tail, -t`

Verfolgt die Protokollausgabe. Dadurch wird das Endzeitargument ignoriert und es werden weiterhin Protokolle abgerufen, sobald sie verfügbar sind.

Beispiele

Wenn Ihre Funktionen Teil eines CloudFormation Stacks sind, können Sie Protokolle abrufen, indem Sie bei der Angabe des Stack-Namens die logische ID der Funktion verwenden.

```
$ sam logs -n HelloWorldFunction --stack-name myStack
```

Mithilfe der Optionen -s (--start-time) und -e (--end-time) können Sie Logs für einen bestimmten Zeitraum anzeigen.

```
$ sam logs -n HelloWorldFunction --stack-name myStack -s '10min ago' -e '2min ago'
```

Sie können auch die `--tail` Option hinzufügen, auf neue Protokolle zu warten und sie zu sehen, sobald sie eintreffen.

```
$ sam logs -n HelloWorldFunction --stack-name myStack --tail
```

Verwenden Sie `--filter` diese Option, um schnell nach Protokollen zu suchen, die Begriffen, Ausdrücken oder Werten in Ihren Protokollereignissen entsprechen.

```
$ sam logs -n HelloWorldFunction --stack-name myStack --filter "error"
```

Die Protokolle für eine Ressource in einem untergeordneten Stapel anzeigen.

```
$ sam logs --stack-name myStack -n childStack/HelloWorldFunction
```

Protokolldateien für alle unterstützten Ressourcen in Ihrer Anwendung.

```
$ sam logs --stack-name sam-app --tail
```

Rufen Sie Protokolle für eine bestimmte Lambda-Funktion und API Gateway Gateway-API in Ihrer Anwendung ab.

```
$ sam logs --stack-name sam-app --name HelloWorldFunction --name HelloWorldRestApi
```

Rufen Sie Protokolle für alle unterstützten Ressourcen in Ihrer Anwendung und zusätzlich aus den angegebenen Protokollgruppen ab.

```
$ sam logs --cw-log-group /aws/lambda/myfunction-123 --cw-log-group /aws/lambda/
myfunction-456
```

sam package

Das AWS Serverless Application Model Command Line Interface (AWS SAM CLI) packt eine AWS SAM Anwendung.

Dieser Befehl erstellt eine `.zip` Datei mit Ihrem Code und Ihren Abhängigkeiten und lädt die Datei auf Amazon Simple Storage Service (Amazon S3) hoch. AWS SAM aktiviert die Verschlüsselung für alle in Amazon S3 gespeicherten Dateien. Anschließend wird eine Kopie Ihrer AWS SAM Vorlage zurückgegeben und Verweise auf lokale Artefakte durch den Amazon S3 S3-Speicherort ersetzt, an den der Befehl die Artefakte hochgeladen hat.

Wenn Sie diesen Befehl verwenden, wird standardmäßig AWS SAMCLI davon ausgegangen, dass Ihr aktuelles Arbeitsverzeichnis das Stammverzeichnis Ihres Projekts ist. Der AWS SAMCLI erste versucht, eine mit dem gleicher Build Befehl erstellte Vorlagendatei zu finden, die sich im `.aws-sam` Unterordner befindet und benannt `template.yaml` ist. Als Nächstes AWS SAMCLI versucht der, eine Vorlagendatei mit dem Namen `template.yaml` oder `template.yml` im aktuellen Arbeitsverzeichnis zu finden. Wenn Sie die `--template` Option angeben, wird AWS SAMCLI das Standardverhalten außer Kraft gesetzt und es werden nur diese AWS SAM Vorlage und die lokalen Ressourcen, auf die sie verweist, gepackt.

> **Note**
>
> sam deployführt jetzt implizit die Funktionalität von aus. `sam package` Sie können den sam deploy Befehl direkt verwenden, um Ihre Anwendung zu verpacken und bereitzustellen.

Usage

```
$ sam package <arguments> <options>
```

Argumente

Resource ID (Ressourcen-ID)

Die ID der Lambda-Funktion, die verpackt werden soll.

Dieses Argument ist optional. Wenn Ihre Anwendung eine einzelne Lambda-Funktion enthält, packt die AWS SAM CLI sie. Wenn Ihre Anwendung mehrere Funktionen enthält, geben Sie die ID der Funktion an, um eine einzelne Funktion zu verpacken.

Gültige Werte: Die logische ID oder der Ressourcen-ARN der Ressource.

Optionen

`--config-env` *TEXT*

Der Umgebungsname, der die Standardparameterwerte in der zu verwendenden Konfigurationsdatei angibt. Der Standardwert ist „default“. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--config-file` *PATH*

Der Pfad und der Dateiname der Konfigurationsdatei, die die zu verwendenden Standardparameterwerte enthält. Der Standardwert ist „samconfig.toml“ im Stammverzeichnis des Projektverzeichnisses. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--debug`

Aktiviert die Debug-Protokollierung, um die von den generierten Debug-Nachrichten zu drucken und Zeitstempel anzuzeigen. AWS SAMCLI

`--force-upload`

Überschreiben Sie vorhandene Dateien im Amazon S3 S3-Bucket. Geben Sie dieses Flag an, um Artefakte hochzuladen, auch wenn sie mit vorhandenen Artefakten im Amazon S3 S3-Bucket übereinstimmen.

`--help`

Zeigt diese Nachricht an und wird beendet.

`--image-repository` *TEXT*

Die URI des Amazon Elastic Container Registry (Amazon ECR) -Repositorys, in das dieser Befehl das Image Ihrer Funktion hochlädt. Erforderlich für Funktionen, die mit dem `Image` Pakettyp deklariert wurden.

`--kms-key-id` *TEXT*

Die ID eines AWS Key Management Service (AWS KMS) -Schlüssels, der zum Verschlüsseln von Artefakten verwendet wird, die sich im Amazon S3 S3-Bucket befinden. Wenn diese Option nicht angegeben ist, werden von Amazon AWS SAM S3 verwaltete Verschlüsselungsschlüssel verwendet.

`--metadata`

(Optional) Eine Metadatenübersicht, die an alle Artefakte angehängt werden kann, auf die in Ihrer Vorlage verwiesen wird.

`--no-progressbar`

Zeigen Sie keinen Fortschrittsbalken an, wenn Sie Artefakte auf Amazon S3 hochladen.

`--output-template-file` *PATH*

Der Pfad zu der Datei, in die der Befehl die verpackte Vorlage schreibt. Wenn Sie keinen Pfad angeben, schreibt der Befehl die Vorlage in die Standardausgabe.

`--profile` *TEXT*

Das spezifische Profil aus Ihrer Anmeldeinformationsdatei, das die AWS Anmeldeinformationen abruft.

`--region` *TEXT*

Die AWS Region, in der die Bereitstellung erfolgen soll. Beispiel: us-east-1.

`--resolve-s3`

Erstellen Sie automatisch einen Amazon S3 S3-Bucket, der für die Verpackung verwendet werden soll. Wenn Sie `--s3-bucket` sowohl die als auch die `--resolve-s3` Optionen angeben, tritt ein Fehler auf.

`--s3-bucket` *TEXT*

Der Name des Amazon S3 S3-Buckets, in den dieser Befehl Ihr Artefakt hochlädt. Wenn Ihr Artefakt größer als 51.200 Byte ist, ist entweder die Option `--s3-bucket` oder die Option erforderlich. `--resolve-s3` Wenn Sie sowohl die `--s3-bucket` `--resolve-s3` Optionen als auch angeben, tritt ein Fehler auf.

`--s3-prefix` *TEXT*

Dem Namen der Artefakte, die in den Amazon S3 S3-Bucket hochgeladen werden, wurde ein Präfix hinzugefügt. Der Präfixname ist ein Pfadname (Ordnername) für den Amazon S3 S3-Bucket. Dies gilt nur für Funktionen, die mit dem `Zip` Pakettyp deklariert wurden.

`--save-params`

Speichern Sie die Parameter, die Sie in der Befehlszeile angeben, in der AWS SAM Konfigurationsdatei.

`--signing-profiles` *LIST*

(Optional) Die Liste der Signaturprofile, mit denen Sie Ihre Bereitstellungspakete signieren können. Dieser Parameter verwendet eine Liste von Schlüssel-Wert-Paaren, wobei der Schlüssel der Name der Funktion oder Ebene ist, die signiert werden soll, und der Wert das Signaturprofil ist, wobei der optionale Profilbesitzer durch getrennt ist. `:` Beispiel, `FunctionNameToSign=SigningProfileName1 LayerNameToSign=SigningProfileName2:SigningProfileOwner`.

`--template-file, --template, -t` *PATH*

Der Pfad und der Dateiname, in dem sich Ihre AWS SAM Vorlage befindet.

> **Note**
>
> Wenn Sie diese Option angeben, werden nur die Vorlage und die lokalen Ressourcen, auf die sie verweist, AWS SAM verpackt.

`--use-json`

Geben Sie JSON für die CloudFormation Vorlage aus. YAML wird standardmäßig verwendet.

Beispiel

Im folgenden Beispiel werden Artefakte für eine Lambda-Funktion und CodeDeploy -Anwendungen erstellt und verpackt. Artefakte werden in einen Amazon S3 S3-Bucket hochgeladen. Die Ausgabe des Befehls ist eine neue Datei mit dem Namen `package.yml`.

```
$ sam package \
  --template-file template.yml \
  --output-template-file package.yml \
  --s3-bucket amzn-s3-demo-bucket
```

sam pipeline bootstrap

Diese Seite enthält Referenzinformationen für den AWS Serverless Application Model `sam local pipeline bootstrap` Unterbefehl Command Line Interface (AWS SAMCLI).

Eine Einführung in das finden Sie AWS SAMCLI unter Was ist der AWS SAM CLI?

Der `sam pipeline bootstrap` Unterbefehl generiert die erforderlichen AWS Infrastrukturressourcen, um eine Verbindung zu Ihrem CI/CD-System herzustellen. Dieser Schritt muss für jede Bereitstellungsphase in Ihrer Pipeline ausgeführt werden, bevor der Befehl ausgeführt wird. sam pipeline init

Dieser Unterbefehl richtet die folgenden AWS Infrastrukturressourcen ein:

- Option zur Konfiguration von Pipeline-Berechtigungen über:
  - Ein Pipeline-IAM-Benutzer mit einer Zugriffsschlüssel-ID und geheimen Schlüsselzugriffsanmeldeinformationen, die mit dem CI/CD System geteilt werden sollen.

    > **Note**
    >
    > Wir empfehlen, die Zugangsschlüssel regelmäßig zu wechseln. Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter Regelmäßiges Rotieren von Zugriffsschlüsseln für Anwendungsfälle, die langfristige Anmeldeinformationen erfordern.

  - Unterstützte CI/CD Plattformen über OIDC. Eine Einführung in die Verwendung von OIDC mit AWS SAM Pipeline finden Sie unter. So verwenden Sie die OIDC-Authentifizierung mit Pipelines AWS SAM
- Eine CloudFormation Ausführungs-IAM-Rolle, die CloudFormation zur Bereitstellung der Anwendung übernommen wurde. AWS SAM
- Ein Amazon S3 S3-Bucket zur Aufbewahrung der AWS SAM Artefakte.
- Optional ein Amazon ECR-Image-Repository für Lambda-Bereitstellungspakete für Container-Images (falls Sie über eine Ressource vom Typ `Image` Paket verfügen).

Usage

```
$ sam pipeline bootstrap <options>
```

Optionen

`--bitbucket-repo-uuid` *TEXT*

Die UUID des Bitbucket-Repositorys. Diese Option ist spezifisch für die Verwendung von Bitbucket OIDC für Berechtigungen.

> **Note**
>
> Dieser Wert ist unter https://bitbucket.org/*workspace*//-connect zu finden *repository* admin/addon/admin/pipelines/openid

`--bucket` *TEXT*

Der ARN des Amazon S3 S3-Buckets, der die AWS SAM Artefakte enthält.

`--cicd-provider` *TEXT*

Die CI/CD Plattform für die AWS SAM Pipeline.

`--cloudformation-execution-role` *TEXT*

Der ARN der IAM-Rolle, der CloudFormation bei der Bereitstellung des Anwendungsstapels übernommen werden soll. Geben Sie dies nur an, wenn Sie Ihre eigene Rolle verwenden möchten. Andernfalls erstellt der Befehl eine neue Rolle.

`--config-env` *TEXT*

Der Umgebungsname, der die Standardparameterwerte in der zu verwendenden Konfigurationsdatei angibt. Der Standardwert ist **`default`**. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--config-file` *PATH*

Der Pfad und der Dateiname der Konfigurationsdatei, die die zu verwendenden Standardparameterwerte enthält. Der Standardwert befindet sich `samconfig.toml` im Stammverzeichnis des Projektverzeichnisses. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--confirm-changeset | --no-confirm-changeset`

Fordern Sie auf, die Bereitstellung Ihrer Ressourcen zu bestätigen.

`--create-image-repository | --no-create-image-repository`

Geben Sie an, ob ein Amazon ECR-Image-Repository erstellt werden soll, falls keines bereitgestellt wird. Das Amazon ECR-Repository enthält die Container-Images von Lambda-Funktionen oder Ebenen mit dem Pakettyp. `Image` Der Standardwert ist `--no-create-image-repository`.

`--debug`

Aktiviert die Debug-Protokollierung und druckt die von ihm AWS SAMCLI generierten Debug-Nachrichten sowie die Anzeige von Zeitstempeln.

`--deployment-branch` *TEXT*

Name der Filiale, von der aus Bereitstellungen erfolgen. Diese Option ist spezifisch für die Verwendung von GitHub Actions OIDC für Berechtigungen.

`--github-org` *TEXT*

Die GitHub Organisation, zu der das Repository gehört. Wenn keine Organisation existiert, geben Sie den Benutzernamen des Repository-Besitzers ein. Diese Option ist spezifisch für die Verwendung von GitHub Actions OIDC für Berechtigungen.

`--github-repo` *TEXT*

Name des GitHub Repositorys, von dem aus Bereitstellungen erfolgen. Diese Option ist spezifisch für die Verwendung von GitHub Actions OIDC für Berechtigungen.

`--gitlab-group` *TEXT*

Die GitLab Gruppe, zu der das Repository gehört. Diese Option ist spezifisch für die Verwendung von GitLab OIDC für Berechtigungen.

`--gitlab-project` *TEXT*

Der GitLab Projektname. Diese Option ist spezifisch für die Verwendung von GitLab OIDC für Berechtigungen.

`--help, -h`

Zeigt diese Nachricht an und wird beendet.

`--image-repository` *TEXT*

Der ARN eines Amazon ECR-Image-Repositorys, das die Container-Images von Lambda-Funktionen oder Ebenen mit dem Pakettyp enthält. `Image` Falls angegeben, werden die `--create-image-repository` Optionen ignoriert. Falls nicht angegeben und `--create-image-repository` angegeben, erstellt der Befehl eine.

`--interactive | --no-interactive`

Deaktivieren Sie die interaktive Aufforderung für Bootstrap-Parameter und schlagen Sie fehl, wenn erforderliche Parameter fehlen. Der Standardwert ist `--interactive`. Für diesen Befehl `--stage` ist der einzige erforderliche Parameter.

> **Note**
>
> Wenn zusammen mit angegeben `--no-interactive` wird`--use-oidc-provider`, müssen alle erforderlichen Parameter für Ihren OIDC-Anbieter enthalten sein.

`--oidc-client-id` *TEXT*

Die Client-ID, die für die Verwendung mit Ihrem OIDC-Anbieter konfiguriert ist.

`--oidc-provider` *[github-actions | gitlab | bitbucket-pipelines]*

Name des CI/CD Anbieters, der für OIDC-Berechtigungen verwendet wird. GitLab GitHub, und Bitbucket werden unterstützt.

`--oidc-provider-url` *TEXT*

Die URL für den OIDC-Anbieter. Der Wert muss mit beginnen. **`https://`**

`--permissions-provider` *[oidc | iam]*

Wählen Sie einen Berechtigungsanbieter aus, der die Rolle der Pipeline-Ausführung übernehmen soll. Der Standardwert ist **`iam`**.

`--pipeline-execution-role` *TEXT*

Der ARN der IAM-Rolle, den der Pipeline-Benutzer annehmen muss, um in dieser Phase zu arbeiten. Geben Sie dies nur an, wenn Sie Ihre eigene Rolle verwenden möchten. Wenn nicht angegeben, erstellt dieser Befehl eine neue Rolle.

`--pipeline-user` *TEXT*

Der Amazon-Ressourcenname (ARN) des IAM-Benutzers, dessen Zugriffsschlüssel-ID und geheimer Zugriffsschlüssel mit dem CI/CD System geteilt werden. Er wird verwendet, um diesem IAM-Benutzer die Erlaubnis zu erteilen, auf das entsprechende AWS Konto zuzugreifen. Falls nicht angegeben, erstellt der Befehl einen IAM-Benutzer zusammen mit der Zugriffsschlüssel-ID und den Anmeldeinformationen für den geheimen Zugriffsschlüssel.

`--profile` *TEXT*

Das spezifische Profil aus Ihrer Anmeldeinformationsdatei, das die Anmeldeinformationen abruft AWS .

`--region` *TEXT*

Die AWS Region, in der die Bereitstellung erfolgen soll. Beispiel, `us-east-1`.

`--save-params`

Speichern Sie die Parameter, die Sie in der Befehlszeile angeben, in der AWS SAM Konfigurationsdatei.

`--stage` *TEXT*

Der Name der entsprechenden Bereitstellungsphase. Er wird als Suffix für die erstellten AWS Infrastrukturressourcen verwendet.

Fehlerbehebung

Fehler: Erforderlicher Parameter fehlt

Wenn zusammen mit angegeben `--no-interactive` wird `--use-oidc-provider` und keiner der erforderlichen Parameter angegeben wird, wird diese Fehlermeldung zusammen mit einer Beschreibung der fehlenden Parameter angezeigt.

Beispiel

Das folgende Beispiel erstellt die AWS Ressourcen, die für die Erstellung Ihres CI/CD Systems erforderlich sind, aktiviert die Debug-Protokollierung und druckt Debug-Meldungen, die generiert wurden von AWS SAMCLI: verwendet ein generiertes Ereignis für lokale Tests, indem ein Ereignis verwendet wird, um eine `s3.json` Lambda-Funktion lokal aufzurufen

```
$ sam pipeline bootstrap --debug
```

sam pipeline init

Diese Seite enthält Referenzinformationen für den AWS Serverless Application Model `sam pipeline init` Unterbefehl Command Line Interface (AWS SAMCLI).

Eine Einführung in den finden Sie AWS SAMCLI unter Was ist der AWS SAM CLI?

Der `sam pipeline init` Unterbefehl generiert eine Pipeline-Konfigurationsdatei, mit der Ihr CI/CD System serverlose Anwendungen bereitstellen kann. AWS SAM

Vor der Verwendung sam pipeline init müssen Sie die erforderlichen Ressourcen für jede Phase in Ihrer Pipeline per Bootstrap bereitstellen. Sie können dies tun, indem Sie sich durch den Prozess sam pipeline init --bootstrap zur Erstellung der Setup- und Konfigurationsdatei führen lassen oder auf Ressourcen verweisen, die Sie zuvor mit dem sam pipeline bootstrap Befehl erstellt haben.

Usage

```
$ sam pipeline init <options>
```

Optionen

`--bootstrap`

Aktivieren Sie den interaktiven Modus, der den Benutzer durch die Erstellung der erforderlichen AWS Infrastrukturressourcen führt.

`--config-env` *TEXT*

Der Umgebungsname, der die Standardparameterwerte in der zu verwendenden Konfigurationsdatei angibt. Der Standardwert ist `default`. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--config-file` *TEXT*

Der Pfad und der Dateiname der Konfigurationsdatei, die die zu verwendenden Standardparameterwerte enthält. Der Standardwert befindet sich `samconfig.toml` im Stammverzeichnis des Projekts. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--debug`

Aktiviert die Debug-Protokollierung, um die von ihm AWS SAMCLI generierten Debug-Meldungen zu drucken und Zeitstempel anzuzeigen.

`--help, -h`

Zeigt diese Meldung an und beendet das Programm.

`--save-params`

Speichern Sie die Parameter, die Sie in der Befehlszeile angeben, in der AWS SAM Konfigurationsdatei.

Beispiel

Das folgende Beispiel zeigt Ihnen, wie Sie die `--bootstrap` Option verwenden können, um einen interaktiven Modus zu durchlaufen, der Sie durch die Erstellung der erforderlichen AWS Infrastrukturressourcen führt:

```
$ sam pipeline init --bootstrap
```

sam publish

Diese Seite enthält Referenzinformationen für den AWS Serverless Application Model Befehl Command Line Interface (AWS SAMCLI)`sam publish`.

Eine Einführung in das finden AWS SAMCLI Sie unter Was ist der AWS SAM CLI?

Der `sam publish` Befehl veröffentlicht eine AWS SAM Anwendung auf dem AWS Serverless Application Repository. Dieser Befehl verwendet eine verpackte AWS SAM Vorlage und veröffentlicht die Anwendung in der angegebenen AWS Region.

Der `sam publish` Befehl erwartet, dass die AWS SAM Vorlage einen `Metadata` Abschnitt enthält, der Anwendungsmetadaten enthält, die für die Veröffentlichung erforderlich sind. In `Metadata` diesem Abschnitt müssen sich die `ReadmeUrl` Eigenschaften `LicenseUrl` und auf Amazon Simple Storage Service (Amazon S3) -Buckets beziehen, nicht auf lokale Dateien. Weitere Informationen zum `Metadata` Abschnitt der AWS SAM Vorlage finden Sie unterVeröffentlichen Sie Ihre Bewerbung mit dem AWS SAMCLI.

`sam publish`Erstellt die Anwendung standardmäßig als privat. Bevor andere AWS Konten Ihre Anwendung anzeigen und bereitstellen können, müssen Sie sie teilen. Informationen zur gemeinsamen Nutzung von Anwendungen finden Sie unter Beispiele für AWS Serverless Application Repository ressourcenbasierte Richtlinien im AWS Serverless Application Repository Entwicklerhandbuch.

> **Note**
>
> Unterstützt derzeit `sam publish` nicht das Veröffentlichen von verschachtelten Anwendungen, die lokal angegeben sind. Wenn Ihre Anwendung verschachtelte Anwendungen enthält, müssen Sie diese separat von der veröffentlichen, AWS Serverless Application Repository bevor Sie Ihre übergeordnete Anwendung veröffentlichen.

Usage

```
$ sam publish <options>
```

Optionen

`--config-env` *TEXT*

Der Umgebungsname, der die Standardparameterwerte in der zu verwendenden Konfigurationsdatei angibt. Der Standardwert ist „default“. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--config-file` *PATH*

Der Pfad und der Dateiname der Konfigurationsdatei, die die zu verwendenden Standardparameterwerte enthält. Der Standardwert ist "`samconfig.toml`" im Stammverzeichnis des Projektverzeichnisses. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--debug`

Aktiviert die Debug-Protokollierung, um die von ihm AWS SAMCLI generierten Debug-Meldungen zu drucken und Zeitstempel anzuzeigen.

`--help`

Zeigt diese Meldung an und beendet das Programm.

`--profile` *TEXT*

Das spezifische Profil aus Ihrer Anmeldeinformationsdatei, das die Anmeldeinformationen abruft AWS .

`--region` *TEXT*

Die AWS Region, in der die Bereitstellung erfolgen soll. Beispiel: us-east-1.

`--save-params`

Speichern Sie die Parameter, die Sie in der Befehlszeile angeben, in der AWS SAM Konfigurationsdatei.

`--semantic-version` *TEXT*

(Optional) Verwenden Sie diese Option, um eine semantische Version Ihrer Anwendung bereitzustellen, die die `SemanticVersion` Eigenschaft im `Metadata` Abschnitt der Vorlagendatei überschreibt. Weitere Informationen zur semantischen Versionierung finden Sie in der Semantic Versioning-Spezifikation.

`--template, -t PATH`

Der Pfad der Vorlagendatei. AWS SAM `[default: template.[yaml|yml]]`

Beispiele

Um eine Anwendung zu veröffentlichen:

```
$ sam publish --template packaged.yaml --region us-east-1
```

sam remote callback

Diese Seite enthält Referenzinformationen für den AWS Serverless Application Model Befehl Command Line Interface (AWS SAMCLI)`sam remote callback`.

Eine Einführung in das finden AWS SAMCLI Sie unter Was ist der AWS SAM CLI?

Mit dem `sam remote callback` Befehl können Sie Rückrufe an Remoteausführungen von dauerhaften Funktionen senden, die in AWS Lambda ausgeführt werden.

> **Note**
>
> Diese Befehle erfordern Zugriff auf Anmeldeinformationen. AWS

Usage

```
$ sam remote callback <subcommand> <options>
```

Optionen

`--help, -h`

Zeige diese Nachricht und beende das Programm.

Unterbefehle

`succeed`

Sendet einen erfolgreichen Rückruf an die Ausführung einer dauerhaften Remote-Funktion. Weitere Informationen finden Sie unter sam remote callback succeed.

`fail`

Sendet einen fehlgeschlagenen Rückruf an die Ausführung einer dauerhaften Remotefunktion. Weitere Informationen finden Sie unter sam remote callback fail.

`heartbeat`

Sendet einen Callback-Heartbeat an eine dauerhafte Remotefunktion. Weitere Informationen finden Sie unter sam remote callback heartbeat.

sam remote callback succeed

Sendet einen erfolgreichen Rückruf an die Ausführung einer dauerhaften Remote-Funktion.

> **Note**
>
> Dieser Befehl erfordert Zugriff auf AWS Anmeldeinformationen.

Usage

```
$ sam remote callback succeed CALLBACK_ID [OPTIONS]
```

Erforderliche Argumente

`CALLBACK_ID`

Die eindeutige Kennung des Rückrufs, an den eine Erfolgsantwort gesendet werden soll.

Optionen

`-r, --result TEXT`

Payload des Erfolgsergebnisses als Zeichenfolge.

`--region TEXT`

Legen Sie die AWS Region des Dienstes fest (z. B.`us-east-1`).

`--profile TEXT`

Wählen Sie ein bestimmtes Profil aus Ihrer Anmeldeinformationsdatei aus, um die AWS Anmeldeinformationen abzurufen.

`--config-env TEXT`

Umgebungsname, der die Standardparameterwerte in der Konfigurationsdatei angibt. Standard: `default`

`--config-file TEXT`

Konfigurationsdatei mit Standardparameterwerten. Standard: `samconfig.toml`

`--save-params`

Speichern Sie die über die Befehlszeile bereitgestellten Parameter in der Konfigurationsdatei.

`--beta-features / --no-beta-features`

Aktivieren/Deaktivieren von Beta-Funktionen.

`--debug`

Aktivieren Sie die Debug-Protokollierung, um von der AWS SAM CLI generierte Debug-Nachrichten zu drucken und Zeitstempel anzuzeigen.

`-h, --help`

Zeigt diese Meldung an und beendet das Programm.

Beispiele

Erfolgreicher Rückruf ohne Ergebnis senden:

```
$ sam remote callback succeed my-callback-id
```

Erfolgs-Rückruf mit Ergebnis senden:

```
$ sam remote callback succeed my-callback-id --result 'Task completed successfully'
```

sam remote callback fail

Sendet einen Callback-Fehler an die Ausführung einer dauerhaften Remotefunktion.

> **Note**
>
> Dieser Befehl erfordert Zugriff auf AWS Anmeldeinformationen.

Usage

```
$ sam remote callback fail CALLBACK_ID [OPTIONS]
```

Erforderliche Argumente

`CALLBACK_ID`

Die eindeutige Kennung des Callbacks, an den eine Fehlerantwort gesendet werden soll.

Optionen

`--error-data TEXT`

Zusätzliche Fehlerdaten.

`--stack-trace TEXT`

Trace-Einträge stapeln. Diese Option kann mehrfach angegeben werden.

`--error-type TEXT`

Fehlertyp.

`--error-message TEXT`

Ausführliche Fehlermeldung.

`--region TEXT`

Stellen Sie die AWS Region des Dienstes ein (z. B.`us-east-1`).

`--profile TEXT`

Wählen Sie ein bestimmtes Profil aus Ihrer Anmeldeinformationsdatei aus, um die AWS Anmeldeinformationen abzurufen.

`--config-env TEXT`

Umgebungsname, der die Standardparameterwerte in der Konfigurationsdatei angibt. Standard: `default`

`--config-file TEXT`

Konfigurationsdatei mit Standardparameterwerten. Standard: `samconfig.toml`

`--save-params`

Speichern Sie die über die Befehlszeile bereitgestellten Parameter in der Konfigurationsdatei.

`--beta-features / --no-beta-features`

Aktivieren/Deaktivieren von Beta-Funktionen.

`--debug`

Aktivieren Sie die Debug-Protokollierung, um von der AWS SAM CLI generierte Debug-Nachrichten zu drucken und Zeitstempel anzuzeigen.

`-h, --help`

Zeigt diese Meldung an und beendet das Programm.

Beispiele

Fehlgeschlagener Rückruf ohne Parameter senden:

```
$ sam remote callback fail my-callback-id
```

Fehlerrückruf mit Fehlermeldung senden:

```
$ sam remote callback fail my-callback-id --error-message 'Task failed'
```

Fehler-Callback mit allen Parametern senden:

```
$ sam remote callback fail my-callback-id --error-message 'Task failed' --error-type
 'ValidationError' --stack-trace 'at line 42' --error-data '{"code": 500}'
```

sam remote callback heartbeat

Sendet einen Callback-Heartbeat an eine dauerhafte Remotefunktion.

> **Note**
>
> Dieser Befehl erfordert Zugriff auf AWS Anmeldeinformationen.

Usage

```
$ sam remote callback heartbeat CALLBACK_ID [OPTIONS]
```

Erforderliche Argumente

`CALLBACK_ID`

Die eindeutige Kennung des Rückrufs, an den ein Heartbeat gesendet werden soll.

Optionen

`--region TEXT`

Stellen Sie die AWS Region des Dienstes ein (z. B.`us-east-1`).

`--profile TEXT`

Wählen Sie ein bestimmtes Profil aus Ihrer Anmeldeinformationsdatei aus, um die AWS Anmeldeinformationen abzurufen.

`--config-env TEXT`

Umgebungsname, der die Standardparameterwerte in der Konfigurationsdatei angibt. Standard: `default`

`--config-file TEXT`

Konfigurationsdatei mit Standardparameterwerten. Standard: `samconfig.toml`

`--save-params`

Speichern Sie die über die Befehlszeile bereitgestellten Parameter in der Konfigurationsdatei.

`--beta-features / --no-beta-features`

Aktivieren/Deaktivieren von Beta-Funktionen.

`--debug`

Aktivieren Sie die Debug-Protokollierung, um von der AWS SAM CLI generierte Debug-Nachrichten zu drucken und Zeitstempel anzuzeigen.

`-h, --help`

Zeigt diese Meldung an und beendet das Programm.

Beispiele

Heartbeat-Rückruf senden:

```
$ sam remote callback heartbeat my-callback-id
```

sam remote execution

Diese Seite enthält Referenzinformationen für den AWS Serverless Application Model Befehl Command Line Interface (AWS SAMCLI)`sam remote execution`.

Eine Einführung in den finden AWS SAMCLI Sie unter Was ist der AWS SAM CLI?

Mit `sam remote execution` diesem Befehl können Sie die in AWS Lambda ausgeführten dauerhaften Remote-Funktionen verwalten und überprüfen.

> **Note**
>
> Diese Befehle erfordern Zugriff auf AWS Anmeldeinformationen.

Usage

```
$ sam remote execution <subcommand> <options>
```

Optionen

`--help, -h`

Diese Nachricht anzeigen und beenden.

Unterbefehle

`get`

Erfahren Sie mehr über eine dauerhafte Ausführung. Weitere Informationen finden Sie unter sam remote execution get.

`history`

Ruft den Ausführungsverlauf einer dauerhaften Funktionsausführung ab. Weitere Informationen finden Sie unter sam remote execution history.

`stop`

Stoppt die Ausführung einer dauerhaften Funktion. Weitere Informationen finden Sie unter sam remote execution stop.

sam remote execution get

Rufen Sie die Details einer bestimmten dauerhaften Funktionsausführung von AWS Lambda ab.

Der Ausführungs-ARN sollte in dem Format vorliegen, das von Invoke zurückgegeben wurde oder in den Ausführungsprotokollen zu finden ist.

> **Note**
>
> Dieser Befehl erfordert Zugriff auf AWS Anmeldeinformationen.

Usage

```
$ sam remote execution get DURABLE_EXECUTION_ARN [OPTIONS]
```

Erforderliche Argumente

`DURABLE_EXECUTION_ARN`

Der Amazon-Ressourcenname (ARN) der dauerhaften Funktionsausführung, für die Details abgerufen werden sollen.

Optionen

`--format [summary|json]`

Ausgabeformat; Standard: `summary`

`--region TEXT`

Legen Sie die AWS Region des Dienstes fest (z. B.`us-east-1`).

`--profile TEXT`

Wählen Sie ein bestimmtes Profil aus Ihrer Anmeldeinformationsdatei aus, um die AWS Anmeldeinformationen abzurufen.

`--config-env TEXT`

Umgebungsname, der die Standardparameterwerte in der Konfigurationsdatei angibt. Standard: `default`

`--config-file TEXT`

Konfigurationsdatei mit Standardparameterwerten. Standard: `samconfig.toml`

`--save-params`

Speichern Sie die über die Befehlszeile bereitgestellten Parameter in der Konfigurationsdatei.

`--beta-features / --no-beta-features`

Aktivieren/Deaktivieren von Beta-Funktionen.

`--debug`

Aktivieren Sie die Debug-Protokollierung, um von der AWS SAM CLI generierte Debug-Nachrichten zu drucken und Zeitstempel anzuzeigen.

`-h, --help`

Zeigt diese Meldung an und beendet das Programm.

Beispiele

Ruft die Ausführungsdetails im Standard-Zusammenfassungsformat ab:

```
$ sam remote execution get 'arn:aws:lambda:us-east-1:123456789012:function:my-function:$LATEST/durable-execution/c63eec67-3415-4eb4-a495-116aa3a86278/1d454231-a3ad-3694-aa03-c917c175db55'
```

Holen Sie sich Ausführungsdetails im JSON-Format:

```
$ sam remote execution get 'arn:aws:lambda:us-east-1:123456789012:function:my-function:$LATEST/durable-execution/c63eec67-3415-4eb4-a495-116aa3a86278/1d454231-a3ad-3694-aa03-c917c175db55' --format json
```

> **Note**
>
> Sie müssen sicherstellen, dass Steuerzeichen im Ausführungs-ARN wie $ bei der Verwendung von Shell-Befehlen ordnungsgemäß maskiert werden.

sam remote execution history

Rufen Sie den Ausführungshistorie einer bestimmten dauerhaften Funktionsausführung von AWS Lambda ab.

Der Ausführungs-ARN sollte das Format haben, das von den Ausführungsprotokollen zurückgegeben StartDurableExecution oder in diesen gefunden wurde.

> **Note**
>
> Dieser Befehl erfordert Zugriff auf AWS Anmeldeinformationen.

Usage

```
$ sam remote execution history DURABLE_EXECUTION_ARN [OPTIONS]
```

Erforderliche Argumente

`DURABLE_EXECUTION_ARN`

Der Amazon-Ressourcenname (ARN) der dauerhaften Funktionsausführung, für die der Verlauf abgerufen werden soll.

Optionen

`--format [table|json]`

Ausgabeformat; Standard: `table`

`--region TEXT`

Legen Sie die AWS Region des Dienstes fest (z. B.`us-east-1`).

`--profile TEXT`

Wählen Sie ein bestimmtes Profil aus Ihrer Anmeldeinformationsdatei aus, um die AWS Anmeldeinformationen abzurufen.

`--config-env TEXT`

Umgebungsname, der die Standardparameterwerte in der Konfigurationsdatei angibt. Standard: `default`

`--config-file TEXT`

Konfigurationsdatei mit Standardparameterwerten. Standard: `samconfig.toml`

`--save-params`

Speichern Sie die über die Befehlszeile bereitgestellten Parameter in der Konfigurationsdatei.

`--beta-features / --no-beta-features`

Aktivieren/Deaktivieren von Beta-Funktionen.

`--debug`

Aktivieren Sie die Debug-Protokollierung, um von der AWS SAM CLI generierte Debug-Nachrichten zu drucken und Zeitstempel anzuzeigen.

`-h, --help`

Zeigt diese Meldung an und beendet das Programm.

Beispiele

Ruft den Ausführungsverlauf im Standardtabellenformat ab:

```
$ sam remote execution history arn:aws:lambda:us-
east-1:123456789012:function:MyFunction:$LATEST/durable-execution/my-execution-name/my-
execution-id
```

Den Ausführungsverlauf im JSON-Format abrufen:

```
$ sam remote execution history arn:aws:lambda:us-
east-1:123456789012:function:MyFunction:$LATEST/durable-execution/my-execution-name/my-
execution-id --format json
```

sam remote execution stop

Stoppen Sie die Ausführung einer laufenden dauerhaften Funktion in AWS Lambda.

Der Ausführungs-ARN sollte das Format haben, das von den Ausführungsprotokollen zurückgegeben StartDurableExecution oder in diesen gefunden wurde.

> **Note**
>
> Dieser Befehl erfordert Zugriff auf AWS Anmeldeinformationen.

Usage

```
$ sam remote execution stop DURABLE_EXECUTION_ARN [OPTIONS]
```

Erforderliche Argumente

`DURABLE_EXECUTION_ARN`

Der Amazon-Ressourcenname (ARN) der dauerhaften Funktionsausführung, die gestoppt werden soll.

Optionen

`--error-message TEXT`

Fehlermeldung, die mit der gestoppten Ausführung verknüpft werden soll.

`--error-type TEXT`

Fehlertyp, der der gestoppten Ausführung zugeordnet werden soll.

`--error-data TEXT`

Fehlerdaten, die der gestoppten Ausführung zugeordnet werden sollen.

`--stack-trace TEXT`

Trace-Einträge stapeln. Diese Option kann mehrfach angegeben werden.

`--region TEXT`

Legen Sie die AWS Region des Dienstes fest (z. B.`us-east-1`).

`--profile TEXT`

Wählen Sie ein bestimmtes Profil aus Ihrer Anmeldeinformationsdatei aus, um die AWS Anmeldeinformationen abzurufen.

`--config-env TEXT`

Umgebungsname, der die Standardparameterwerte in der Konfigurationsdatei angibt. Standard: `default`

`--config-file TEXT`

Konfigurationsdatei mit Standardparameterwerten. Standard: `samconfig.toml`

`--save-params`

Speichern Sie die über die Befehlszeile bereitgestellten Parameter in der Konfigurationsdatei.

`--beta-features / --no-beta-features`

Aktivieren/Deaktivieren von Beta-Funktionen.

`--debug`

Aktivieren Sie die Debug-Protokollierung, um von der AWS SAM CLI generierte Debug-Nachrichten zu drucken und Zeitstempel anzuzeigen.

`-h, --help`

Zeigt diese Meldung an und beendet das Programm.

Beispiele

Ausführung ohne Fehlerdetails beenden:

```
$ sam remote execution stop arn:aws:lambda:us-east-1:123456789012:function:MyFunction:
$LATEST/durable-execution/my-execution-name/my-execution-id
```

Beenden Sie die Ausführung mit einer Fehlermeldung und geben Sie Folgendes ein:

```
$ sam remote execution stop arn:aws:lambda:us-east-1:123456789012:function:MyFunction:
$LATEST/durable-execution/my-execution-name/my-execution-id --error-message "Execution
 cancelled" --error-type "UserCancellation"
```

Beenden Sie die Ausführung mit vollständigen Fehlerdetails und Stack-Trace:

```
$ sam remote execution stop arn:aws:lambda:us-east-1:123456789012:function:MyFunction:
$LATEST/durable-execution/my-execution-name/my-execution-id --error-message "Task
 failed" --error-type "TaskFailure" --error-data '{"reason":"timeout"}' --stack-trace
 "at function1()" --stack-trace "at function2()"
```

sam remote invoke

Diese Seite enthält Referenzinformationen für den AWS Serverless Application Model Befehl Command Line Interface (AWS SAMCLI)`sam remote invoke`.

- Eine Einführung in den finden AWS SAMCLI Sie unter Was ist der AWS SAM CLI?
- Eine Dokumentation zur Verwendung des AWS SAMCLI `sam remote invoke` Befehls finden Sie unterEinführung in das Testen in der Cloud mit sam remote invoke.

Der `sam remote invoke` Befehl ruft unterstützte Ressourcen in der AWS Cloud auf.

Usage

```
$ sam remote invoke <arguments> <options>
```

Argumente

Resource ID (Ressourcen-ID)

Die ID der unterstützten Ressource, die aufgerufen werden soll.

Dieses Argument akzeptiert die folgenden Werte:

- Amazon Resource Name (ARN) — Der ARN der Ressource.

> **Tip**
>
> Verwenden Sie`sam list stack-outputs --stack-name <stack-name>`, um den ARN Ihrer Ressourcen zu erhalten.

- Logische ID — Die logische ID der Ressource. Sie müssen auch den AWS CloudFormation Stack-Namen mithilfe der `--stack-name` Option angeben.
- Physikalische ID — Die physische ID der Ressource. Diese ID wird erstellt, wenn Sie eine Ressource mithilfe von bereitstellen CloudFormation.

> **Tip**
>
> Verwenden Sie`sam list resources --stack-name <stack-name>`, um die physische ID Ihrer Ressourcen abzurufen.

Wenn Sie einen ARN oder eine physische ID angeben:

Wenn Sie einen ARN oder eine physische ID angeben, geben Sie keinen Stacknamen an. Wenn der Stack-Name mithilfe der `--stack-name` Option bereitgestellt wird oder wenn der Stack-Name in Ihrer Konfigurationsdatei definiert ist, AWS SAM CLI wird Ihre Ressourcen-ID automatisch als logischen ID-Wert aus dem CloudFormation Stack verarbeitet.

Wenn Sie keine Ressourcen-ID angeben:

Wenn Sie keine Ressourcen-ID angeben, aber mit der `--stack-name` Option einen Stacknamen angeben, versucht die AWS SAM CLI, mithilfe der folgenden Logik automatisch eine Ressource in Ihrem CloudFormation Stack aufzurufen:

1. AWS SAM CLISie identifiziert die Ressourcentypen in der folgenden Reihenfolge und fährt mit dem nächsten Schritt fort, sobald der Ressourcentyp in Ihrem Stack gefunden wurde:
   a. Lambda
   b. Step Functions
   c. Amazon SQS
   d. Kinesis Data Streams
2. Wenn der Ressourcentyp eine einzelne Ressource in Ihrem Stack hat, AWS SAM CLI wird sie aufgerufen. Wenn mehrere Ressourcen des Ressourcentyps in Ihrem Stack vorhanden sind, AWS SAM CLI wird ein Fehler zurückgegeben.

Im Folgenden finden Sie Beispiele dafür, was AWS SAM CLI sie tun werden:

- Stack, der zwei Lambda-Funktionen und eine Amazon SQS SQS-Warteschlange enthält — Der sucht den AWS SAM CLI Lambda-Ressourcentyp und gibt Fehler zurück, da der Stack mehr als eine Lambda-Funktion enthält.
- Stack, der eine Lambda-Funktion und zwei Amazon Kinesis Data Streams Streams-Anwendungen enthält — Der sucht AWS SAM CLI die Lambda-Funktion und ruft sie auf, da der Stack eine einzige Lambda-Ressource enthält.
- Stack, der eine einzelne Amazon SQS SQS-Warteschlange und zwei Kinesis Data Streams Streams-Anwendungen enthält — Der sucht die AWS SAM CLI Amazon SQS SQS-Warteschlange und ruft sie auf, da der Stack eine einzige Amazon SQS SQS-Warteschlange enthält.

Optionen

`--beta-features | --no-beta-features`

Betafunktionen zulassen oder ablehnen.

`--config-env` *`TEXT`*

Geben Sie die zu verwendende Umgebung aus Ihrer AWS SAMCLI Konfigurationsdatei an.

Standardwert: `default`

`--config-file` *`FILENAME`*

Geben Sie den Pfad und den Dateinamen Ihrer Konfigurationsdatei an.

Weitere Informationen zu Konfigurationsdateien finden Sie unter Konfiguration der AWS SAMCLI.

Standard: `samconfig.toml` im Stammverzeichnis Ihres Projektverzeichnisses.

`--debug`

Aktivieren Sie die Debug-Protokollierung. Dadurch werden Debug-Nachrichten und Zeitstempel gedruckt, die von der generiert wurden. AWS SAMCLI

`--event, -e` *`TEXT`*

Das Ereignis, das an die Zielressource gesendet werden soll.

`--event-file` *`FILENAME`*

Der Pfad zu einer Datei, die das Ereignis enthält, das an die Zielressource gesendet werden soll.

`--help, -h`

Zeigt die Hilfemeldung an und beendet den Vorgang.

`--output [ text | json ]`

Geben Sie die Ergebnisse Ihres Aufrufs in einem bestimmten Ausgabeformat aus.

`json`— Die Anforderungsmetadaten und die Ressourcenantwort werden in der JSON-Struktur zurückgegeben. Die Antwort enthält die vollständige SDK-Ausgabe.

`text`— Die Metadaten der Anfrage werden in Textstruktur zurückgegeben. Die Ressourcenantwort wird im Ausgabeformat der aufgerufenen Ressource zurückgegeben.

`--parameter`

Zusätzliche Boto3Parameter, die Sie an die aufgerufene Ressource übergeben können.

Amazon Kinesis Data Streams

Die folgenden zusätzlichen Parameter können verwendet werden, um einen Datensatz in den Kinesis-Datenstrom aufzunehmen:

- `ExplicitHashKey='string'`
- `PartitionKey='string'`
- `SequenceNumberForOrdering='string'`
- `StreamARN='string'`

Eine Beschreibung der einzelnen Parameter finden Sie unter kinesis.client.PUT_RECORD.

AWS Lambda

Die folgenden zusätzlichen Parameter können verwendet werden, um eine Lambda-Ressource aufzurufen und eine gepufferte Antwort zu erhalten:

- `ClientContext='base64-encoded string'`
- `InvocationType='[ DryRun | Event | RequestResponse ]'`
- `LogType='[ None | Tail ]'`
- `Qualifier='string'`

Die folgenden zusätzlichen Parameter können verwendet werden, um eine Lambda-Ressource mit Antwortstreaming aufzurufen:

- `ClientContext='base64-encoded string'`
- `InvocationType='[ DryRun | RequestResponse ]'`
- `LogType='[ None | Tail ]'`
- `Qualifier='string'`

Eine Beschreibung der einzelnen Parameter finden Sie im Folgenden:

- Lambda mit gepufferter Antwort — Lambda.Client.Invoke
- Lambda mit Antwortstreaming — Lambda.Client.Invoke_with_Response_Stream

Amazon-Simple-Queue-Service (Amazon SQS)

Die folgenden zusätzlichen Parameter können verwendet werden, um eine Nachricht an eine Amazon SQS SQS-Warteschlange zu senden:

- `DelaySeconds=integer`
- `MessageAttributes='json string'`
- `MessageDeduplicationId='string'`
- `MessageGroupId='string'`
- `MessageSystemAttributes='json string'`

Eine Beschreibung der einzelnen Parameter finden Sie unter sqs.client.send_Message.

AWS Step Functions

Die folgenden zusätzlichen Parameter können verwendet werden, um die Ausführung einer Zustandsmaschine zu starten:

- `name='string'`
- `traceHeader='string'`

Eine Beschreibung der einzelnen Parameter finden Sie unter SFN.Client.Start_Execution.

`--profile TEXT`

Das spezifische Profil aus Ihrer Anmeldeinformationsdatei zum Abrufen der Anmeldeinformationen. AWS

`--region TEXT`

Das AWS-Region der Ressource. Beispiel, `us-east-1`.

`--stack-name` *TEXT*

Der Name des CloudFormation Stacks, zu dem die Ressource gehört.

`--tenant-id` *TEXT*

Die Mandanten-ID für Lambda-Funktionen mit mehreren Mandanten. Wird verwendet, um die Rechenisolierung zwischen verschiedenen Mandanten sicherzustellen. Erforderlich beim Aufrufen von Funktionen, die mit dem Mandantenisolationsmodus konfiguriert sind.

`--test-event-name` *NAME*

Der Name des gemeinsam nutzbaren Testereignisses, das an Ihre Lambda-Funktion übergeben werden soll.

> **Note**
>
> Diese Option unterstützt nur Lambda-Funktionen.

Beispiel

Das folgende Beispiel ruft unterstützte Ressourcen in der AWS Cloud auf und aktiviert die Debug-Protokollierung, die Debug-Nachrichten und Zeitstempel ausgibt, die generiert wurden von: AWS SAMCLI

```
$ sam remote invoke--debug
```

sam remote test-event

Diese Seite enthält Referenzinformationen für den AWS Serverless Application Model Befehl Command Line Interface (AWS SAMCLI)`sam remote test-event`.

- Eine Einführung in das finden AWS SAMCLI Sie unter Was ist der AWS SAM CLI?
- Eine Dokumentation zur Verwendung des AWS SAMCLI `sam remote test-event` Befehls finden Sie unterEinführung in Cloud-Tests mit sam remote test-event.

Der `sam remote test-event` Befehl interagiert mit gemeinsam nutzbaren Testereignissen in der EventBridge Amazon-Schemaregistry.

Usage

```
$ sam remote test-event <options> <subcommand>
```

Optionen

`--help, -h`

Zeigt die Hilfemeldung an und beendet das Programm.

Unterbefehle

`delete`

Löscht ein gemeinsam nutzbares Testereignis aus der EventBridge Schemaregistrierung. Weitere Referenzinformationen finden Sie untersam remote test-event delete.

`get`

Ruft ein gemeinsam nutzbares Testereignis aus der EventBridge Schemaregistrierung ab. Weitere Referenzinformationen finden Sie untersam remote test-event get.

`list`

Listet bestehende Testereignisse für eine AWS Lambda Funktion auf, die gemeinsam genutzt werden können. Weitere Referenzinformationen finden Sie untersam remote test-event list.

`put`

Speichern Sie ein Ereignis aus einer lokalen Datei in der EventBridge Schemaregistrierung. Weitere Referenzinformationen finden Sie untersam remote test-event put.

sam remote test-event delete

Diese Seite enthält Referenzinformationen für den AWS Serverless Application Model `sam remote test-event delete` Unterbefehl Command Line Interface (AWS SAMCLI).

- Eine Einführung in den finden Sie AWS SAMCLI unter Was ist der AWS SAM CLI?
- Eine Dokumentation zur Verwendung des AWS SAMCLI `sam remote test-event` Befehls finden Sie unterEinführung in Cloud-Tests mit sam remote test-event.

Der `sam remote test-event delete` Unterbefehl löscht ein gemeinsam nutzbares Testereignis aus der EventBridge Amazon-Schemaregistry.

Usage

```
$ sam remote test-event delete <arguments> <options>
```

Argumente

Resource ID (Ressourcen-ID)

Die ID der AWS Lambda Funktion, die dem gemeinsam nutzbaren Testereignis zugeordnet ist.

Wenn Sie eine logische ID angeben, müssen Sie mithilfe der `--stack-name` Option auch einen Wert für den AWS CloudFormation Stack angeben, der der Lambda-Funktion zugeordnet ist.

Gültige Werte: Die logische ID oder Ressource der RessourceARN.

Optionen

`--config-env` *TEXT*

Der Umgebungsname, der die Standardparameterwerte in der zu verwendenden Konfigurationsdatei angibt. Der Standardwert ist „default“. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--config-file` *PATH*

Der Pfad und der Dateiname der Konfigurationsdatei, die die zu verwendenden Standardparameterwerte enthält. Der Standardwert ist "`samconfig.toml`" im Stammverzeichnis des Projektverzeichnisses. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--help, -h`

Zeigt die Hilfemeldung an und beendet das Programm.

`--name` *TEXT*

Der Name des gemeinsam nutzbaren Testereignisses, das gelöscht werden soll.

`--stack-name` *TEXT*

Der Name des CloudFormation Stacks, der der Lambda-Funktion zugeordnet ist.

Diese Option ist erforderlich, wenn Sie die logische ID der Lambda-Funktion als Argument angeben.

sam remote test-event get

Diese Seite enthält Referenzinformationen für den AWS Serverless Application Model `sam remote test-event get` Unterbefehl Command Line Interface (AWS SAMCLI).

- Eine Einführung in den finden Sie AWS SAMCLI unter Was ist der AWS SAM CLI?
- Eine Dokumentation zur Verwendung des AWS SAMCLI `sam remote test-event` Befehls finden Sie unterEinführung in Cloud-Tests mit sam remote test-event.

Der `sam remote test-event get` Unterbefehl ruft ein gemeinsam nutzbares Testereignis aus der EventBridge Amazon-Schemaregistry ab.

Usage

```
$ sam remote test-event get <arguments> <options>
```

Argumente

Resource ID (Ressourcen-ID)

Die ID der AWS Lambda Funktion, die dem abzurufenden gemeinsam nutzbaren Testereignis zugeordnet ist.

Wenn Sie eine logische ID angeben, müssen Sie mithilfe der `--stack-name` Option auch einen Wert für den AWS CloudFormation Stack angeben, der der Lambda-Funktion zugeordnet ist.

Gültige Werte: Die logische ID oder Ressource der RessourceARN.

Optionen

`--config-env` *TEXT*

Der Umgebungsname, der die Standardparameterwerte in der zu verwendenden Konfigurationsdatei angibt. Der Standardwert ist „default“. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--config-file` *PATH*

Der Pfad und der Dateiname der Konfigurationsdatei, die die zu verwendenden Standardparameterwerte enthält. Der Standardwert ist "`samconfig.toml`" im Stammverzeichnis des Projektverzeichnisses. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--help, -h`

Zeigt die Hilfemeldung an und beendet das Programm.

`--name` *TEXT*

Der Name des gemeinsam nutzbaren Testereignisses, das abgerufen werden soll.

`--output-file` *FILENAME*

Der Dateipfad und der Name, unter dem das Ereignis auf Ihrem lokalen Computer gespeichert werden soll.

Wenn Sie diese Option nicht angeben, AWS SAM CLI wird der Inhalt des gemeinsam nutzbaren Testereignisses auf Ihrer Konsole ausgegeben.

`--stack-name` *TEXT*

Der Name des CloudFormation Stacks, der der Lambda-Funktion zugeordnet ist.

Diese Option ist erforderlich, wenn Sie die logische ID der Lambda-Funktion als Argument angeben.

sam remote test-event list

Diese Seite enthält Referenzinformationen für den AWS Serverless Application Model `sam remote test-event list` Unterbefehl Command Line Interface (AWS SAMCLI).

- Eine Einführung in den finden Sie AWS SAMCLI unter Was ist der AWS SAM CLI?
- Eine Dokumentation zur Verwendung des AWS SAMCLI `sam remote test-event` Befehls finden Sie unterEinführung in Cloud-Tests mit sam remote test-event.

Der `sam remote test-event list` Unterbefehl listet die vorhandenen gemeinsam nutzbaren Testereignisse für eine bestimmte AWS Lambda Funktion aus der EventBridge Amazon-Schemaregistrierung auf.

Usage

```
$ sam remote test-event list <arguments> <options>
```

Argumente

Resource ID (Ressourcen-ID)

Die ID der Lambda-Funktion, die den gemeinsam nutzbaren Testereignissen zugeordnet ist.

Wenn Sie eine logische ID angeben, müssen Sie mithilfe der `--stack-name` Option auch einen Wert für den AWS CloudFormation Stack angeben, der der Lambda-Funktion zugeordnet ist.

Gültige Werte: Die logische ID oder Ressource der RessourceARN.

Optionen

`--config-env` *TEXT*

Der Umgebungsname, der die Standardparameterwerte in der zu verwendenden Konfigurationsdatei angibt. Der Standardwert ist „default“. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--config-file` *PATH*

Der Pfad und der Dateiname der Konfigurationsdatei, die die zu verwendenden Standardparameterwerte enthält. Der Standardwert ist "`samconfig.toml`" im Stammverzeichnis des Projektverzeichnisses. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--help, -h`

Zeigt die Hilfemeldung an und beendet das Programm.

`--stack-name` *TEXT*

Der Name des CloudFormation Stacks, der der Lambda-Funktion zugeordnet ist.

Diese Option ist erforderlich, wenn Sie die logische ID der Lambda-Funktion als Argument angeben.

Beispiele

Beispiele zur Verwendung dieses Befehls finden Sie unterTestereignisse auflisten, die gemeinsam genutzt werden können.

sam remote test-event put

Diese Seite enthält Referenzinformationen für den AWS Serverless Application Model `sam remote test-event put` Unterbefehl Command Line Interface (AWS SAMCLI).

- Eine Einführung in den finden Sie AWS SAMCLI unter Was ist der AWS SAM CLI?
- Eine Dokumentation zur Verwendung des AWS SAMCLI `sam remote test-event` Befehls finden Sie unterEinführung in Cloud-Tests mit sam remote test-event.

Der `sam remote test-event put` Unterbefehl speichert ein gemeinsam nutzbares Testereignis von Ihrem lokalen Computer in der EventBridge Amazon-Schemaregistrierung.

Usage

```
$ sam remote test-event put <arguments> <options>
```

Argumente

Resource ID (Ressourcen-ID)

Die ID der AWS Lambda Funktion, die dem gemeinsam nutzbaren Testereignis zugeordnet ist.

Wenn Sie eine logische ID angeben, müssen Sie mithilfe der `--stack-name` Option auch einen Wert für den AWS CloudFormation Stack angeben, der der Lambda-Funktion zugeordnet ist.

Gültige Werte: Die logische ID oder Ressource der RessourceARN.

Optionen

`--config-env` *TEXT*

Der Umgebungsname, der die Standardparameterwerte in der zu verwendenden Konfigurationsdatei angibt. Der Standardwert ist „default“. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--config-file` *PATH*

Der Pfad und der Dateiname der Konfigurationsdatei, die die zu verwendenden Standardparameterwerte enthält. Der Standardwert ist "`samconfig.toml`" im Stammverzeichnis des Projektverzeichnisses. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--file` *FILENAME*

Der Dateipfad und der Name des Ereignisses auf Ihrem lokalen Computer.

Geben Sie `-` als Dateinamenwert an, aus dem gelesen werden soll`stdin`.

Diese Option ist erforderlich.

`--force`, `-f`

Überschreibt ein gemeinsam nutzbares Testereignis mit demselben Namen.

`--help, -h`

Zeigt die Hilfemeldung an und beendet das Programm.

`--name` *TEXT*

Der Name, unter dem das gemeinsam nutzbare Testereignis gespeichert werden soll.

Wenn ein gemeinsam nutzbares Testereignis mit demselben Namen in der EventBridge Schemaregistrierung vorhanden ist, AWS SAM CLI wird es nicht überschrieben. Um zu überschreiben, fügen Sie die Option hinzu. `--force`

`--output-file` *FILENAME*

Der Dateipfad und der Name, unter dem das Ereignis auf Ihrem lokalen Computer gespeichert werden soll.

Wenn Sie diese Option nicht angeben, AWS SAM CLI wird der Inhalt des gemeinsam nutzbaren Testereignisses auf Ihrer Konsole ausgegeben.

`--stack-name` *TEXT*

Der Name des CloudFormation Stacks, der der Lambda-Funktion zugeordnet ist.

Diese Option ist erforderlich, wenn Sie die logische ID der Lambda-Funktion als Argument angeben.

Beispiel

Ein Beispiel zur Verwendung dieses Befehls finden Sie unterTestereignisse speichern, die gemeinsam genutzt werden können.

Sam Sync

Diese Seite enthält Referenzinformationen für den AWS Serverless Application Model Befehl Command Line Interface (AWS SAMCLI)`sam sync`.

- Eine Einführung in den finden AWS SAMCLI Sie unter Was ist der AWS SAM CLI?
- Eine Dokumentation zur Verwendung von finden Sie unterAWS SAM CLI. AWS SAMCLI

Der `sam sync` Befehl synchronisiert lokale Anwendungsänderungen mit dem AWS Cloud.

Usage

```
$ sam sync <options>
```

Optionen

`--base-dir, -s` *`DIRECTORY`*

Löst relative Pfade zum Quellcode der Funktion oder Ebene in Bezug auf dieses Verzeichnis auf. Verwenden Sie diese Option, um zu ändern, wie relative Pfade zu Quellcodeordnern aufgelöst werden. Standardmäßig werden relative Pfade in Bezug auf den Speicherort der AWS SAM Vorlage aufgelöst.

Zusätzlich zu den Ressourcen in der Root-Anwendung oder dem Root-Stack, den Sie erstellen, gilt diese Option auch für verschachtelte Anwendungen oder Stacks. Darüber hinaus gilt diese Option für die folgenden Ressourcentypen und Eigenschaften:

- Ressourcentyp: `AWS::Serverless::Function` Eigenschaft: `CodeUri`
- Ressourcentyp: `AWS::Serverless::Function` Ressourcenattribut: `Metadata` Eintrag: `DockerContext`
- Ressourcentyp: `AWS::Serverless::LayerVersion` Immobilie: `ContentUri`
- Ressourcentyp: `AWS::Lambda::Function` Immobilie: `Code`

- Ressourcentyp: `AWS::Lambda::LayerVersion` Immobilie: `Content`

`--build-image` *TEXT*

Der URI für das Container-Image, das Sie beim Erstellen Ihrer Anwendung verwenden möchten. AWS SAM Verwendet standardmäßig den Container-Image-Repository-URI von Amazon Elastic Container Registry (Amazon ECR) Public. Geben Sie diese Option an, um ein anderes Bild zu verwenden.

Sie können diese Option in einem einzigen Befehl mehrfach verwenden. Jede Option akzeptiert eine Zeichenfolge oder ein Schlüssel-Wert-Paar.

- Zeichenfolge — Geben Sie den URI des Container-Images an, das alle Ressourcen in Ihrer Anwendung verwenden werden. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam sync --build-image amazon/aws-sam-cli-build-image-python3.8
```

- Key-value pair — Geben Sie den Ressourcennamen als Schlüssel und die Container-Image-URI an, die mit dieser Ressource als Wert verwendet werden soll. Verwenden Sie dieses Format, um für jede Ressource in Ihrer Anwendung einen anderen Container-Image-URI anzugeben. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam sync --build-image Function1=amazon/aws-sam-cli-build-image-python3.8
```

Diese Option gilt nur, wenn die `--use-container` Option angegeben ist, andernfalls tritt ein Fehler auf.

`--build-in-source | --no-build-in-source`

Ermöglicht `--build-in-source` es, Ihr Projekt direkt im Quellordner zu erstellen.

Die `--build-in-source` Option unterstützt die folgenden Laufzeiten und Build-Methoden:

- Laufzeiten — Jede Node.js Laufzeit, die von der `sam init --runtime` Option unterstützt wird.
- Methoden erstellen —`Makefile`,`esbuild`.

Die `--build-in-source` Option ist mit den folgenden Optionen nicht kompatibel:

- `--use-container`

Standardwert: `--no-build-in-source`

`--capabilities` *`LIST`*

Eine Liste von Funktionen, die Sie angeben, um die Erstellung bestimmter Stapel CloudFormation zu ermöglichen. Einige Stack-Vorlagen können Ressourcen enthalten, die sich auf Ihre AWS-Konto Berechtigungen auswirken können. Zum Beispiel durch das Erstellen neuer AWS Identity and Access Management (IAM-) Benutzer. Geben Sie diese Option an, um die Standardwerte zu überschreiben. Gültige Werte sind z. B. die Folgenden:

- CAPABILITY_IAM
- CAPABILITY_NAMED_IAM
- CAPABILITY_RESOURCE_POLICY
- CAPABILITY_AUTO_EXPAND

Standard: und `CAPABILITY_NAMED_IAM` `CAPABILITY_AUTO_EXPAND`

`--code`

AWS SAM Synchronisiert standardmäßig alle Ressourcen in Ihrer Anwendung. Geben Sie diese Option an, um nur Coderessourcen zu synchronisieren, die Folgendes beinhalten:

- `AWS::Serverless::Function`
- `AWS::Lambda::Function`
- `AWS::Serverless::LayerVersion`
- `AWS::Lambda::LayerVersion`
- `AWS::Serverless::Api`
- `AWS::ApiGateway::RestApi`
- `AWS::Serverless::HttpApi`
- `AWS::ApiGatewayV2::Api`
- `AWS::Serverless::StateMachine`
- `AWS::StepFunctions::StateMachine`

Zum Synchronisieren von Coderessourcen werden AWS Dienst-APIs direkt AWS SAM verwendet, anstatt sie über bereitzustellen AWS CloudFormation. Um Ihren CloudFormation Stack zu aktualisieren, führen Sie sam sync --watch oder aussam deploy.

`--config-env` *TEXT*

Der Umgebungsname, der die Standardparameterwerte in der zu verwendenden Konfigurationsdatei angibt. Der Standardwert ist „default“. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--config-file` *PATH*

Der Pfad und der Dateiname der Konfigurationsdatei, die die zu verwendenden Standardparameterwerte enthält. Der Standardwert ist "`samconfig.toml`" im Stammverzeichnis des Projektverzeichnisses. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--dependency-layer | --no-dependency-layer`

Gibt an, ob die Abhängigkeiten einzelner Funktionen in eine weitere Ebene aufgeteilt werden sollen, um den Synchronisierungsvorgang zu beschleunigen.

Standardwert: `--dependency-layer`

`--express | --no-express`

Gibt an, ob der CloudFormation Expressmodus für Infrastrukturbereitstellungen verwendet werden soll. Wenn diese Option aktiviert ist, werden die Stack-Operationen abgeschlossen, sobald die Ressourcenkonfiguration angewendet wurde. Die Ressourcen werden weiterhin im Hintergrund bereit.

Der Expressmodus gilt nur für den Infrastruktur-Synchronisierungspfad. Code-only Synchronisationen (Lambda-Funktionsupdates, Bereitstellungen) umgehen diese Option CloudFormation und sind von dieser Option nicht betroffen.

Der Standardwert ist `--no-express`.

> **Kommende Standardänderung**
>
> Die Standardeinstellung wird `--express` auf den 1. September 2026 geändert.

`--image-repository` *TEXT*

Der Name des Amazon Elastic Container Registry (Amazon ECR) -Repositorys, in das dieser Befehl das Image Ihrer Funktion hochlädt. Erforderlich für Funktionen, die mit dem `Image` Pakettyp deklariert wurden.

`--image-repositories` *TEXT*

Eine Zuordnung von Funktionen zu ihrer Amazon ECR-Repository-URI. Referenzfunktionen anhand ihrer logischen ID. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam sync --image-repositories Function1=123456789012.dkr.ecr.us-
east-1.amazonaws.com/my-repo
```

Sie können diese Option in einem einzigen Befehl mehrfach angeben.

`--kms-key-id` *TEXT*

Die ID eines AWS Key Management Service (AWS KMS) -Schlüssels, der zum Verschlüsseln von Artefakten verwendet wird, die sich im Amazon S3 S3-Bucket befinden. Wenn Sie diese Option nicht angeben, werden S3-managed Amazon-Verschlüsselungsschlüssel AWS SAM verwendet.

`--metadata`

Eine Metadatenübersicht, die an alle Artefakte angehängt werden kann, auf die Sie in Ihrer Vorlage verweisen.

`--notification-arns` *LIST*

Eine Liste der Themen-ARNs von Amazon Simple Notification Service (Amazon SNS), die dem CloudFormation Stack zugeordnet sind.

`--no-use-container`

Eine Option, mit der Sie das IDE-Toolkit verwenden können, um das Standardverhalten festzulegen.

`--parameter-overrides`

Eine Zeichenfolge, die CloudFormation Parameterüberschreibungen enthält, die als Schlüssel-Wert-Paare kodiert sind. Verwenden Sie dasselbe Format wie (). AWS Command Line Interface AWS CLI Das AWS SAMCLI Format besteht aus expliziten Schlüsselwörtern und Werten, wobei jede Überschreibung durch ein Leerzeichen getrennt ist. Nachfolgend finden Sie zwei Beispiele:

- `--parameter-overrides ParameterKey=hello,ParameterValue=world`
- `--parameter-overrides ParameterKey=hello,ParameterValue=world ParameterKey=example1,ParameterValue=example2 ParameterKey=apple,ParameterValue=banana`

`--resource` *TEXT*

Gibt den zu synchronisierenden Ressourcentyp an. Um mehrere Ressourcen zu synchronisieren, können Sie diese Option mehrmals angeben. Diese Option wird zusammen mit der `--code` Option unterstützt. Bei dem Wert muss es sich um eine der unten aufgeführten Ressourcen handeln`--code`. Beispiel, `--resource AWS::Serverless::Function --resource AWS::Serverless::LayerVersion`.

`--resource-id` *TEXT*

Gibt die Ressourcen-ID an, die synchronisiert werden soll. Um mehrere Ressourcen zu synchronisieren, können Sie diese Option mehrmals angeben. Diese Option wird zusammen mit der `--code` Option unterstützt. Beispiel, `--resource-id Function1 --resource-id Function2`.

`--role-arn` *TEXT*

Der Amazon-Ressourcenname (ARN) einer IAM-Rolle, der beim Anwenden des Changesets übernommen CloudFormation wird.

`--s3-bucket` *TEXT*

Der Name des Amazon Simple Storage Service (Amazon S3) -Buckets, in den dieser Befehl Ihre CloudFormation Vorlage hochlädt. Wenn Ihre Vorlage größer als 51.200 Byte ist, ist entweder die Option `--s3-bucket` oder die `--resolve-s3` Option erforderlich. Wenn Sie sowohl die `--s3-bucket` `--resolve-s3` Optionen als auch angeben, tritt ein Fehler auf.

`--s3-prefix` *TEXT*

Das Präfix, das den Namen der Artefakte hinzugefügt wurde, die Sie in den Amazon S3 S3-Bucket hochladen. Der Präfixname ist ein Pfadname (Ordnername) für den Amazon S3 S3-Bucket. Dies gilt nur für Funktionen, die mit dem `Zip` Pakettyp deklariert wurden.

`--save-params`

Speichert die Parameter, die Sie in der Befehlszeile angeben, in der AWS SAM Konfigurationsdatei.

`--skip-deploy-sync | --no-skip-deploy-sync`

Gibt `--skip-deploy-sync` an, dass die anfängliche Infrastruktursynchronisierung übersprungen werden soll, wenn sie nicht erforderlich ist. Der vergleicht AWS SAMCLI Ihre lokale AWS SAM Vorlage mit der bereitgestellten CloudFormation Vorlage und führt nur dann eine Bereitstellung durch, wenn eine Änderung erkannt wird.

Gibt `--no-skip-deploy-sync` an, dass bei jeder Ausführung eine CloudFormation Bereitstellung durchgeführt werden `sam sync` soll.

Weitere Informationen hierzu finden Sie unter Überspringen Sie die erste CloudFormation Bereitstellung.

Standardwert: `--skip-deploy-sync`

`--stack-name` *TEXT*

Der Name des CloudFormation Stacks für Ihre Anwendung.

Diese Option ist erforderlich.

`--tags` *LIST*

Eine Liste von Tags, die dem Stack zugeordnet werden sollen, der erstellt oder aktualisiert wird. CloudFormation leitet diese Tags auch an Ressourcen im Stack weiter, die sie unterstützen.

`--template-file, --template, -t` *PATH*

Der Pfad und der Dateiname, in dem sich Ihre AWS SAM Vorlage befindet.

> **Note**
>
> Wenn Sie diese Option angeben, werden AWS SAM nur die Vorlage und die lokalen Ressourcen bereitgestellt, auf die sie verweist.

`--use-container, -u`

Wenn Ihre Funktionen von Paketen abhängen, die nativ kompilierte Abhängigkeiten haben, verwenden Sie diese Option, um Ihre Funktion in einem AWS Lambdaähnlichen Docker Container zu erstellen.

> **Note**
>
> Derzeit ist diese Option nicht kompatibel mit`--dependency-layer`. Wenn Sie `--use-container` mit verwenden`--dependency-layer`, AWS SAMCLI informiert Sie das und fährt mit fort`--no-dependency-layer`.

`--watch`

Startet einen Prozess, der Ihre lokale Anwendung auf Änderungen überwacht und diese automatisch mit der AWS Cloud synchronisiert. Wenn Sie diese Option angeben, werden standardmäßig alle Ressourcen in Ihrer Anwendung AWS SAM synchronisiert, wenn Sie sie aktualisieren. AWS SAM Führt mit dieser Option eine erste CloudFormation Bereitstellung durch. AWS SAM Verwendet dann AWS Service-APIs, um die Coderessourcen zu aktualisieren. AWS SAM verwendet CloudFormation , um Infrastrukturressourcen zu aktualisieren, wenn Sie Ihre AWS SAM Vorlage aktualisieren.

`--watch-exclude` *TEXT*

Schließt eine Datei oder einen Ordner von der Überwachung auf Dateiänderungen aus. Um diese Option verwenden zu können, `--watch` muss sie ebenfalls angegeben werden.

Diese Option erhält ein Schlüssel-Wert-Paar:

- Schlüssel — Die logische ID einer Lambda-Funktion in Ihrer Anwendung.
- Wert — Der zugehörige Dateiname oder Ordner, der ausgeschlossen werden soll.

Wenn Sie Dateien oder Ordner aktualisieren, die mit dieser `--watch-exclude` Option angegeben wurden, AWS SAM CLI wird keine Synchronisierung initiiert. Wenn jedoch eine Aktualisierung anderer Dateien oder Ordner eine Synchronisierung einleitet, werden diese Dateien oder Ordner in diese Synchronisierung aufgenommen.

Sie können diese Option in einem einzigen Befehl mehrfach angeben.

Beispiele

Beispiele zur Verwendung dieses Befehls finden Sie unterOptionen für den Befehl sam sync.

sam traces

Diese Seite enthält Referenzinformationen für den AWS Serverless Application Model Befehl Command Line Interface (AWS SAMCLI)`sam traces`.

Eine Einführung in das finden AWS SAMCLI Sie unter Was ist der AWS SAM CLI?

Der `sam traces` Befehl ruft AWS X-Ray Spuren AWS-Konto in Ihrem ab. AWS-Region

Usage

```
$ sam traces <options>
```

Optionen

`--config-env` *TEXT*

Der Umgebungsname, der die Standardparameterwerte in der zu verwendenden Konfigurationsdatei angibt. Der Standardwert ist „default“. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--config-file` *PATH*

Der Pfad und der Dateiname der Konfigurationsdatei, die die zu verwendenden Standardparameterwerte enthält. Der Standardwert ist "`samconfig.toml`" im Stammverzeichnis des Projektverzeichnisses. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--end-time` *TEXT*

Ruft Traces bis zu diesem Zeitpunkt ab. Bei der Uhrzeit kann es sich um relative Werte wie „vor 5 Minuten“, „Morgen“ oder um einen formatierten Zeitstempel wie „2018-01-01 10:10:10“ handeln.

`--output` *TEXT*

Gibt das Ausgabeformat für Protokolle an. Um formatierte Protokolle zu drucken, geben Sie an`text`. Um die Protokolle als JSON zu drucken, geben Sie an`json`.

`--save-params`

Speichern Sie die Parameter, die Sie in der Befehlszeile angeben, in der AWS SAM Konfigurationsdatei.

`--start-time` *TEXT*

Ruft ab diesem Zeitpunkt Traces ab. Bei der Uhrzeit kann es sich um relative Werte wie „vor 5 Minuten“, „Gestern“ oder um einen formatierten Zeitstempel wie „2018-01-01 10:10:10“ handeln. Der Standardwert ist 'vor 10 Minuten'.

`--tail`

Schließt die Trace-Ausgabe ab. Dadurch wird das Endzeitargument ignoriert und es werden weiterhin Traces angezeigt, sobald sie verfügbar sind.

`--trace-id` *TEXT*

Die eindeutige Kennung für eine Röntgenspur.

Beispiele

Führen Sie den folgenden Befehl aus, um X-Ray-Traces nach ID abzurufen.

```
$ sam traces --trace-id tracing-id-1 --trace-id tracing-id-2
```

Führen Sie den folgenden Befehl aus, um X-Ray-Traces zu verfolgen, sobald sie verfügbar sind.

```
$ sam traces --tail
```

sam validate

Diese Seite enthält Referenzinformationen für den AWS Serverless Application Model Befehl Command Line Interface (AWS SAMCLI)`sam validate`.

Eine Einführung in das finden AWS SAMCLI Sie unter Was ist der AWS SAM CLI?

Der `sam validate` Befehl überprüft, ob eine AWS SAM Vorlagendatei gültig ist.

Usage

```
$ sam validate <options>
```

Optionen

`--config-env` *TEXT*

Der Umgebungsname, der die Standardparameterwerte in der zu verwendenden Konfigurationsdatei angibt. Der Standardwert ist „default“. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--config-file` *PATH*

Der Pfad und der Dateiname der Konfigurationsdatei, die die zu verwendenden Standardparameterwerte enthält. Der Standardwert ist „samconfig.toml“ im Stammverzeichnis

des Projektverzeichnisses. Weitere Informationen zu Konfigurationsdateien finden Sie unter AWS SAMCLIKonfigurationsdatei.

`--debug`

Aktiviert die Debug-Protokollierung, um die von den generierten Debug-Nachrichten zu drucken und Zeitstempel anzuzeigen. AWS SAMCLI

`--lint`

Führt die Linting-Validierung für die Vorlage durch. cfn-lint Erstellen Sie eine `cfnlintrc` Konfigurationsdatei, um zusätzliche Parameter anzugeben. Weitere Informationen finden Sie unter cfn-lint im CloudFormation GitHub Repository.

`--profile` *TEXT*

Das spezifische Profil aus Ihrer Anmeldeinformationsdatei, das die Anmeldeinformationen abruft. AWS

`--region` *TEXT*

Die AWS Region, in der die Bereitstellung erfolgen soll. Beispiel: us-east-1.

`--save-params`

Speichern Sie die Parameter, die Sie in der Befehlszeile angeben, in der AWS SAM Konfigurationsdatei.

`--template-file, --template, -t` *PATH*

Die AWS SAM Vorlagendatei. Der Standardwert ist `template.[yaml|yml]`.

Wenn sich Ihre Vorlage in Ihrem aktuellen Arbeitsverzeichnis befindet und benannt ist`template.[yaml|yml|json]`, ist diese Option nicht erforderlich.

Wenn Sie gerade ausgeführt habensam build, ist diese Option nicht erforderlich.

Beispiel

Ein Beispiel für die Verwendung dieses Befehls zur Validierung einer Vorlage finden Sie unterAWS SAM Vorlagendateien validieren.

Ein Beispiel zur Verwendung dieses Befehls mit cfn-lint finden Sie unter. Validieren Sie Ihre AWS SAM Bewerbungen mit CloudFormation Linter

AWS SAMCLIVerwaltung

Dieser Abschnitt enthält Informationen darüber, wie Sie Ihre Version von verwalten und anpassen können AWS SAMCLI. Dies beinhaltet Informationen darüber, wie Sie AWS SAMCLI Befehlsparameterwerte mithilfe einer Konfigurationsdatei auf Projektebene konfigurieren können. Es enthält auch Informationen zur Verwaltung verschiedener Versionen von Ihrem AWS SAMCLI, zum Einrichten von AWS Anmeldeinformationen, sodass Sie in Ihrem Namen AWS Dienste aufrufen AWS SAM können, und zu verschiedenen Möglichkeiten, die Sie anpassen können. AWS SAM Dieser Abschnitt endet mit einem Abschnitt zur allgemeinen AWS SAM Problembehandlung.

Topics

- AWS SAMCLIKonfigurationsdatei
- AWS SAMCLIVersionen verwalten
- AWS Zugangsdaten einrichten
- Telemetrie in der AWS SAMCLI
- AWS SAMCLIProblembehandlung

AWS SAMCLIKonfigurationsdatei

Die AWS Serverless Application Model Befehlszeilenschnittstelle (AWS SAMCLI) unterstützt eine Konfigurationsdatei auf Projektebene, mit der Sie AWS SAMCLI Befehlsparameterwerte konfigurieren können.

Eine Dokumentation zum Erstellen und Verwenden von Konfigurationsdateien finden Sie unter. Konfiguration der AWS SAMCLI

Themen

- Standardeinstellungen für die Konfigurationsdatei
- Unterstützte Konfigurationsdateiformate
- Geben Sie eine Konfigurationsdatei an
- Grundlagen der Konfigurationsdatei
- Regeln für Parameterwerte
- Rangfolge der Konfiguration
- Konfigurationsdateien erstellen und ändern

Standardeinstellungen für die Konfigurationsdatei

AWS SAM verwendet die folgenden Standardeinstellungen für die Konfigurationsdatei:

- Name (Name – `samconfig`.
- Standort — Im Stammverzeichnis Ihres Projekts. Dies ist derselbe Speicherort wie Ihre `template.yaml` Datei.
- Format —`TOML`. Weitere Informationen finden Sie unter TOML in der TOMLDokumentation.

Im Folgenden finden Sie ein Beispiel für eine Projektstruktur, die den Namen und den Speicherort der Standardkonfigurationsdatei enthält:

```
sam-app
### README.md
### __init__.py
### events
### hello_world
### samconfig.toml
### template.yaml
### tests
```

Im Folgenden sehen Sie ein Beispiel für eine `samconfig.toml`-Datei:

```
...
version = 0.1

[default]
[default.global]
[default.global.parameters]
stack_name = "sam-app"

[default.build.parameters]
cached = true
parallel = true

[default.deploy.parameters]
capabilities = "CAPABILITY_IAM"
confirm_changeset = true
resolve_s3 = true

[default.sync.parameters]
```

```
watch = true

[default.local_start_api.parameters]
warm_containers = "EAGER"

[prod]
[prod.sync]
[prod.sync.parameters]
watch = false
```

Unterstützte Konfigurationsdateiformate

`TOML`und `[YAML|YML]` Formate werden unterstützt. Sehen Sie sich die folgende grundlegende Syntax an:

TOML

```
version = 0.1
[environment]
[environment.command]
[environment.command.parameters]
option = parameter value
```

YAML

```
version: 0.1
environment:
  command:
    parameters:
      option: parameter value
```

Geben Sie eine Konfigurationsdatei an

Standardmäßig AWS SAMCLI sucht der in der folgenden Reihenfolge nach einer Konfigurationsdatei:

1. Benutzerdefinierte Konfigurationsdatei — Wenn Sie die `--config-file` Option zur Angabe eines Dateinamens und eines Speicherorts verwenden, AWS SAMCLI sucht der zuerst nach dieser Datei.
2. **`samconfig.toml`**Standarddatei — Dies ist der Standardname und das Standardformat der Konfigurationsdatei, die sich im Stammverzeichnis Ihres Projekts befinden. Wenn Sie keine

benutzerdefinierte Konfigurationsdatei angeben, AWS SAMCLI sucht der als Nächstes nach dieser Datei.

3. **`samconfig.[yaml|yml]`**Datei — Wenn die `samconfig.toml` nicht im Stammverzeichnis Ihres Projekts existiert, AWS SAMCLI sucht der nach dieser Datei.

Im Folgenden finden Sie ein Beispiel für die Angabe einer benutzerdefinierten Konfigurationsdatei mithilfe der `--config-file` Option:

```
$ sam deploy --config-file myconfig.yaml
```

> **Note**
>
> Der `--config-file` Parameter muss sich auf den Speicherort der AWS SAM Vorlagendatei beziehen, da er den Kontext bestimmen AWS SAMCLI muss, in dem die Konfiguration angewendet wird. Die `samconfig.toml` Datei verwaltet die Konfigurationseinstellungen für Ihre Version von AWS SAMCLI, und die CLI sucht nach der `samconfig.toml` Datei (oder dem überschriebenen Konfigurationsdateiparameter) im entsprechenden Ordner der `template.yaml` Datei.

Grundlagen der Konfigurationsdatei

Umgebung

Eine Umgebung ist ein benannter Bezeichner, der einen eindeutigen Satz von Konfigurationseinstellungen enthält. Sie können mehrere Umgebungen in einer einzigen AWS SAM Anwendung haben.

Der Standardumgebungsname lautet`default`.

Verwenden Sie die AWS SAMCLI `--config-env` Option, um die zu verwendende Umgebung anzugeben.

Befehl

Der Befehl ist der AWS SAMCLI Befehl, für den Parameterwerte angegeben werden.

Verwenden Sie den `global` Bezeichner, um Parameterwerte für alle Befehle anzugeben.

Wenn Sie auf einen AWS SAMCLI Befehl verweisen, ersetzen Sie Leerzeichen ( ) und Bindestriche () durch Unterstriche (-). _ Im Folgenden sind einige Beispiele aufgeführt:

- `build`
- `local_invoke`
- `local_start_api`

Parameters

Parameter werden als Schlüssel-Wert-Paare angegeben.

- Der Schlüssel ist der Name der AWS SAMCLI Befehlsoption.
- Der Wert ist der anzugebende Wert.

Verwenden Sie bei der Angabe von Schlüsseln den Namen der Befehlsoption in Langform und ersetzen Sie Bindestriche () durch Unterstriche (-). _ Im Folgenden sind einige Beispiele aufgeführt:

- `region`
- `stack_name`
- `template_file`

Regeln für Parameterwerte

TOML

- Boolesche Werte können oder sein. `true` `false` Beispiel, `confirm_changeset = true`.
- Verwenden Sie für Zeichenkettenwerte Anführungszeichen ()`""`. Beispiel, `region = "us-west-2"`.
- Verwenden Sie für Listenwerte Anführungszeichen (`""`) und trennen Sie die einzelnen Werte durch ein Leerzeichen ( ). Beispiel: `capabilities = "CAPABILITY_IAM CAPABILITY_NAMED_IAM"`.
- Bei Werten, die eine Liste von Schlüssel-Wert-Paaren enthalten, sind die Paare durch Leerzeichen getrennt ( ), und der Wert jedes Paares ist von codierten Anführungszeichen () umgeben. `\" \"` Beispiel, `tags = "project=\"my-application\" stage=\"production\""`.

- Bei Parameterwerten, die mehrfach angegeben werden können, ist der Wert ein Array von Argumenten. Beispiel: `image_repositories = ["my-function-1=image-repo-1", "my-function-2=image-repo-2"]`.

YAML

- Boolesche Werte können oder sein. `true` `false` Beispiel, `confirm_changeset: true`.
- Bei Einträgen, die einen einzelnen Zeichenkettenwert enthalten, sind Anführungszeichen (`""`) optional. Beispiel, `region: us-west-2`. Dazu gehören Einträge, die mehrere Schlüssel-Wert-Paare enthalten, die als einzelne Zeichenfolge bereitgestellt werden. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam deploy --tags "foo=bar hello=world"
```

```
default:
  deploy:
    parameters:
      tags: foo=bar hello=world
```

- Bei Einträgen, die eine Werteliste enthalten, oder bei Einträgen, die mehrfach in einem einzigen Befehl verwendet werden können, geben Sie sie als Liste von Zeichenfolgen an.

  Im Folgenden wird ein Beispiel gezeigt:

```
$ sam remote invoke --parameter "InvocationType=Event" --parameter "LogType=None"
```

```
default:
  remote_invoke:
    parameters:
        parameter:
        - InvocationType=Event
        - LogType=None
```

Rangfolge der Konfiguration

Bei der Konfiguration von Werten gilt die folgende Priorität:

- Parameterwerte, die Sie in der Befehlszeile angeben, haben Vorrang vor den entsprechenden Werten in der Konfigurationsdatei und im `Parameters` Abschnitt der Vorlagendatei.
- Wenn die `--parameter-overrides` Option in der Befehlszeile oder in Ihrer Konfigurationsdatei zusammen mit dem `parameter_overrides` Schlüssel verwendet wird, haben ihre Werte Vorrang vor Werten im `Parameters` Abschnitt der Vorlagendatei.
- In Ihrer Konfigurationsdatei haben die für einen bestimmten Befehl bereitgestellten Einträge Vorrang vor globalen Einträgen. Im folgenden Beispiel verwendet der `sam deploy` Befehl den Stack-Namen`my-app-stack`.

  TOML

  ```
  [default.global.parameters]
  stack_name = "common-stack"

  [default.deploy.parameters]
  stack_name = "my-app-stack"
  ```

  YAML

  ```
  default:
    global:
      parameters:
        stack_name: common-stack
    deploy:
      parameters:
        stack_name: my-app-stack
  ```

Konfigurationsdateien erstellen und ändern

Konfigurationsdateien erstellen

Wenn Sie eine Anwendung mit erstellen`sam init`, wird eine `samconfig.toml` Standarddatei erstellt. Sie können Ihre Konfigurationsdatei auch manuell erstellen.

Konfigurationsdateien ändern

Sie können Ihre Konfigurationsdateien manuell ändern. Außerdem werden die konfigurierten Werte während eines AWS SAMCLI interaktiven Ablaufs in Klammern (`[ ]`) angezeigt. Wenn Sie diese Werte ändern, AWS SAMCLI wird Ihre Konfigurationsdatei aktualisiert.

Im Folgenden finden Sie ein Beispiel für einen interaktiven Ablauf mit dem `sam deploy --guided` Befehl:

```
$ sam deploy --guided

Configuring SAM deploy
======================

    Looking for config file [samconfig.toml] :  Found
    Reading default arguments  :  Success

    Setting default arguments for 'sam deploy'
    =========================================
    Stack Name [sam-app]: ENTER
    AWS Region [us-west-2]: ENTER
    #Shows you resources changes to be deployed and require a 'Y' to initiate deploy
    Confirm changes before deploy [Y/n]: n
    #SAM needs permission to be able to create roles to connect to the resources in
 your template
    Allow SAM CLI IAM role creation [Y/n]: ENTER
    #Preserves the state of previously provisioned resources when an operation fails
    Disable rollback [y/N]: ENTER
    HelloWorldFunction may not have authorization defined, Is this okay? [y/N]: y
    Save arguments to configuration file [Y/n]: ENTER
    SAM configuration file [samconfig.toml]: ENTER
    SAM configuration environment [default]: ENTER
```

Wenn Sie Ihre Konfigurationsdatei ändern, AWS SAMCLI behandelt der globale Werte wie folgt:

- Wenn der Parameterwert im `global` Abschnitt Ihrer Konfigurationsdatei vorhanden ist, wird der Wert AWS SAMCLI nicht in den spezifischen Befehlsabschnitt geschrieben.
- Wenn der Parameterwert `global` sowohl im Befehlsabschnitt als auch in einem bestimmten Befehlsabschnitt vorhanden ist, AWS SAMCLI wird der spezifische Eintrag zugunsten des globalen Werts gelöscht.

AWS SAMCLIVersionen verwalten

Verwalten Sie Ihre Versionen der AWS Serverless Application Model Befehlszeilenschnittstelle (AWS SAMCLI) durch Upgrade, Downgrade und Deinstallation. Optional können Sie den Nightly Build herunterladen und installieren. AWS SAMCLI

Topics

- Aktualisierung des AWS SAMCLI
- Deinstallation des AWS SAMCLI
- Wechseln Sie von der Verwendung Homebrew zur Verwaltung von AWS SAMCLI
- Den AWS SAMCLI Nightly-Build verwalten
- Installation des AWS SAMCLI in einer virtuellen Umgebung mit pip
- Verwaltung des AWS SAMCLI mit Homebrew
- Fehlerbehebung

Aktualisierung des AWS SAMCLI

Linux

Um das AWS SAMCLI auf Linux zu aktualisieren, folgen Sie den Installationsanweisungen unterInstallation des AWS SAMCLI, fügen Sie jedoch die `--update` Option wie folgt zum Installationsbefehl hinzu:

```
sudo ./sam-installation/install --update
```

macOS

Die AWS SAMCLI muss mit derselben Methode aktualisiert werden, mit der sie installiert wurde. Wir empfehlen, dass Sie das Paketinstallationsprogramm verwenden, um das zu installieren und zu aktualisieren AWS SAMCLI.

Um das AWS SAMCLI mit dem Paketinstallationsprogramm zu aktualisieren, installieren Sie die neueste Paketversion. Detaillierte Anweisungen finden Sie unter Installation des AWS SAMCLI.

Windows

Um das zu aktualisieren AWS SAMCLI, wiederholen Sie die Windows-Installationsschritte unter Installieren Sie das AWS SAMCLI erneut.

Deinstallation des AWS SAMCLI

Linux

Um das AWS SAMCLI unter Linux zu deinstallieren, müssen Sie den Symlink und das Installationsverzeichnis löschen, indem Sie die folgenden Befehle ausführen:

1. Suchen Sie den Symlink und die Installationspfade.

   - Suchen Sie den Symlink mit dem which folgenden Befehl:

     ```
     which sam
     ```

     Die Ausgabe zeigt den Pfad, in dem sich die AWS SAM Binärdateien befinden, zum Beispiel:

     ```
      /usr/local/bin/sam
     ```

   - Suchen Sie mit dem folgenden Befehl nach dem Verzeichnis, auf das der ls Symlink verweist:

     ```
     ls -l /usr/local/bin/sam
     ```

     Im folgenden Beispiel lautet `/usr/local/aws-sam-cli` das Installationsverzeichnis.

     ```
      lrwxrwxrwx 1 ec2-user ec2-user 49 Oct 22 09:49 /usr/local/bin/sam -> /usr/local/
     aws-sam-cli/current/bin/sam
     ```

2. Löschen Sie den Symlink.

   ```
   sudo rm /usr/local/bin/sam
   ```

3. Löschen Sie das Installationsverzeichnis.

   ```
   sudo rm -rf /usr/local/aws-sam-cli
   ```

macOS

Deinstallieren Sie AWS SAMCLI das mit derselben Methode, mit der es installiert wurde. Wir empfehlen, dass Sie das Paketinstallationsprogramm verwenden, um das zu installieren AWS SAMCLI.

Wenn Sie das AWS SAMCLI mit dem Paketinstallationsprogramm installiert haben, gehen Sie zur Deinstallation wie folgt vor.

Um das zu deinstallieren AWS SAMCLI

1. Entfernen Sie das AWS SAMCLI Programm, indem Sie Folgendes ändern und ausführen:

```
$ sudo rm -rf /path-to/aws-sam-cli
```

a. *sudo*— Wenn Ihr Benutzer Schreibberechtigungen für den Ort hat, an dem AWS SAMCLI das Programm installiert ist, sudo ist dies nicht erforderlich. Andernfalls ist sudo erforderlich.

b. */path-to*— Pfad zu dem Ort, an dem Sie das AWS SAMCLI Programm installiert haben. Der Standardspeicherort ist `/usr/local`.

2. Entfernen Sie das, AWS SAMCLI `$PATH` indem Sie Folgendes ändern und ausführen:

```
$ sudo rm -rf /path-to-symlink-directory/sam
```

a. *sudo*— Wenn Ihr Benutzer Schreibberechtigungen hat`$PATH`, sudo ist dies nicht erforderlich. Andernfalls ist sudo erforderlich.

b. *path-to-symlink-directory*— Ihre `$PATH` Umgebungsvariable. Der Standardspeicherort ist `/usr/local/bin`.

3. Stellen Sie sicher, dass das deinstalliert AWS SAMCLI ist, indem Sie Folgendes ausführen:

```
$ sam --version
command not found: sam
```

Windows

Gehen Sie folgendermaßen vor, um das AWS SAMCLI mithilfe der Windows-Einstellungen zu deinstallieren:

1. Suchen Sie im Startmenü nach „Programme hinzufügen oder entfernen“.
2. Wählen Sie das Ergebnis mit dem Namen AWS SAM Command Line Interface und anschließend Deinstallieren, um das Deinstallationsprogramm zu starten.
3. Bestätigen Sie, dass Sie das AWS SAMCLI deinstallieren möchten.

Wechseln Sie von der Verwendung Homebrew zur Verwaltung von AWS SAMCLI

Wenn Sie Homebrew die Installation und Aktualisierung von verwenden AWS SAMCLI, empfehlen wir die Verwendung einer AWS unterstützten Methode. Folgen Sie diesen Anweisungen, um zu einer unterstützten Methode zu wechseln.

Um von der Verwendung von zu wechseln Homebrew

1. Folgen Sie den Anweisungen unterDeinstallation einer Homebrew installierten CLI AWS SAM, um die Homebrew verwaltete Version zu deinstallieren.
2. Folgen Sie den Anweisungen unterInstallieren Sie das AWS SAMCLI, um die AWS SAM CLI mit einer unterstützten Methode zu installieren.

Den AWS SAMCLI Nightly-Build verwalten

Sie können den AWS SAMCLI Nightly Build herunterladen und installieren. Er enthält eine Vorabversion des AWS SAMCLI Codes, die möglicherweise weniger stabil ist als die Produktionsversion. Nach der Installation können Sie den Nightly Build mit dem `sam-nightly` Befehl verwenden. Sie können sowohl die Produktions- als auch die Nightly-Build-Version von gleichzeitig installieren und verwenden. AWS SAMCLI

> **Note**
>
> Der Nightly-Build enthält keine Vorabversion des Build-Images. Aus diesem Grund wird beim Erstellen Ihrer serverlosen Anwendung mit dieser --use-container Option die neueste Produktionsversion des Build-Images verwendet.

Installation des AWS SAMCLI Nightly-Builds

Folgen Sie diesen Anweisungen, um den AWS SAMCLI Nightly-Build zu installieren.

Linux

Sie können die Nightly-Build-Version von AWS SAMCLI auf der Linux x86_64-Plattform mit dem Paket-Installer installieren.

Um den Nightly Build zu installieren AWS SAMCLI

1. Laden Sie das Paketinstallationsprogramm aus sam-cli-nightlydem aws-sam-cli GitHubRepository herunter.
2. Folgen Sie den Schritten zur Installation des AWS SAMCLI, um das Nightly Build-Paket zu installieren.

macOS

Sie können die Nightly Build-Version von AWS SAMCLI on macOS mit dem Nightly Build Package Installer installieren.

Um den Nightly Build zu installieren AWS SAMCLI

1. Laden Sie das Paketinstallationsprogramm für Ihre Plattform aus sam-cli-nightlydem aws-sam-cli GitHubRepository herunter.
2. Folgen Sie den Schritten zur Installation des AWS SAMCLI, um das Nightly Build-Paket zu installieren.

Windows

Die Nightly Build-Version von AWS SAMCLI ist über diesen Download-Link verfügbar: AWS SAMCLINightly Build. Um den Nightly Build unter Windows zu installieren, führen Sie dieselben Schritte aus wie inInstallieren Sie das AWS SAMCLI, verwenden Sie jedoch stattdessen den Download-Link für den Nightly Build.

Führen Sie den Befehl aus, um zu überprüfen, ob Sie die Nightly Build-Version installiert haben. sam-nightly --version Die Ausgabe dieses Befehls hat zum Beispiel das `1.X.Y.dev<YYYYMMDDHHmm>` folgende Format:

```
SAM CLI, version 1.20.0.dev202103151200
```

Wechseln Sie vom Homebrew Paketinstallationsprogramm

Wenn Sie den AWS SAMCLI Nightly-Build Homebrew zur Installation und zum Upgrade verwenden und zur Verwendung des Paketinstallationsprogramms wechseln möchten, gehen Sie wie folgt vor.

Um vom Homebrew Paket-Installer zu wechseln

1. Deinstallieren Sie den Homebrew installierten AWS SAMCLI Nightly Build.

   ```
   $ brew uninstall aws-sam-cli-nightly
   ```

2. Stellen Sie sicher, dass der AWS SAMCLI Nightly Build deinstalliert wurde, indem Sie Folgendes ausführen:

   ```
   $ sam-nightly --version
   ```

```
zsh: command not found: sam-nightly
```

3. Folgen Sie den Schritten im vorherigen Abschnitt, um den AWS SAMCLI Nightly Build zu installieren.

Installation des AWS SAMCLI in einer virtuellen Umgebung mit pip

Wir empfehlen, das native Paketinstallationsprogramm zu verwenden, um das zu installieren AWS SAMCLI. Wenn Sie es verwenden müssenpip, empfehlen wir Ihnen, das AWS SAMCLI in einer virtuellen Umgebung zu installieren. Dies gewährleistet eine saubere Installationsumgebung und eine isolierte Umgebung, falls Fehler auftreten.

> **Note**
>
> Ab dem 24. Oktober 2023 stellt AWS SAM CLI das den Support für Python 3.7 ein. Weitere Informationen hierzu finden Sie unter AWS SAMCLIEinstellung der Unterstützung für Python 3.7.

Um das AWS SAMCLI in einer virtuellen Umgebung zu installieren

1. Erstellen Sie aus einem Startverzeichnis Ihrer Wahl eine virtuelle Umgebung und geben Sie ihr einen Namen.

   Linux / macOS

   ```
   $ mkdir project
   $ cd project
   $ python3 -m venv venv
   ```

   Windows

   ```
   > mkdir project
   > cd project
   > py -3 -m venv venv
   ```

2. Aktivieren Sie die virtuelle Umgebung

Linux / macOS

```
$ . venv/bin/activate
```

Die Eingabeaufforderung ändert sich und zeigt Ihnen, dass Ihre virtuelle Umgebung aktiv ist.

```
(venv) $
```

Windows

```
> venv\Scripts\activate
```

Die Aufforderung ändert sich und zeigt Ihnen, dass Ihre virtuelle Umgebung aktiv ist.

```
(venv) >
```

3. Installieren Sie das AWS SAMCLI in Ihrer virtuellen Umgebung.

```
(venv) $ pip install --upgrade aws-sam-cli
```

4. Stellen Sie sicher, dass das AWS SAMCLI korrekt installiert ist.

```
(venv) $ sam --version
SAM CLI, version 1.94.0
```

5. Sie können den Befehl `deactivate` verwenden, um die virtuelle Umgebung zu beenden. Wenn Sie eine neue Sitzung starten, müssen Sie die Umgebung erneut aktivieren.

Verwaltung des AWS SAMCLI mit Homebrew

> **Note**
>
> Ab September 2023 AWS wird das AWS verwaltete Homebrew Installationsprogramm für AWS SAMCLI (`aws/tap/aws-sam-cli`) nicht mehr verwaltet. Um die Nutzung fortzusetzenHomebrew, können Sie das von der Community verwaltete Installationsprogramm (`aws-sam-cli`) verwenden. Ab September 2023 wird jeder Homebrew Befehl, auf den verwiesen `aws/tap/aws-sam-cli` wird, umgeleitet`aws-sam-cli`.

Wir empfehlen Ihnen, unsere unterstützten Installations - und Upgrade-Methoden zu verwenden.

Installation des AWS SAMCLI mit Homebrew

> **Note**
>
> Diese Anweisungen verwenden das von der Community verwaltete AWS SAMCLI Homebrew Installationsprogramm. Weitere Unterstützung finden Sie im Homebrew-Core-Repository.

Um das zu installieren AWS SAMCLI

1. Führen Sie Folgendes aus:

   ```
   $ brew install aws-sam-cli
   ```

2. Überprüfen Sie die Installation:

   ```
   $ sam --version
   ```

   Nach erfolgreicher Installation von sollten Sie eine Ausgabe wie die folgende sehen: AWS SAMCLI

   ```
   SAM CLI, version 1.94.0
   ```

Aktualisierung der AWS SAMCLI Verwendung von Homebrew

Führen Sie den folgenden Befehl ausHomebrew, um die AWS SAMCLI Verwendung zu aktualisieren:

```
$ brew upgrade aws-sam-cli
```

Deinstallation einer Homebrew installierten CLI AWS SAM

Wenn das mit installiert AWS SAMCLI wurdeHomebrew, gehen Sie wie folgt vor, um es zu deinstallieren.

Um das zu deinstallieren AWS SAMCLI

1. Führen Sie Folgendes aus:

```
$ brew uninstall aws-sam-cli
```

2. Stellen Sie sicher, dass das deinstalliert AWS SAMCLI ist, indem Sie Folgendes ausführen:

```
$ sam --version
command not found: sam
```

Wechseln Sie zum von der Community verwalteten Homebrew Installationsprogramm

Wenn Sie das AWS verwaltete Homebrew Installationsprogramm (`aws/tap/aws-sam-cli`) verwenden und es vorziehen, es weiterhin zu verwendenHomebrew, empfehlen wir, zum von der Community verwalteten Homebrew Installationsprogramm (`aws-sam-cli`) zu wechseln.

Um in einem einzigen Befehl zu wechseln, führen Sie den folgenden Befehl aus:

```
$ brew uninstall aws-sam-cli && brew untap aws/tap && brew cleanup aws/tap && brew
 update && brew install aws-sam-cli
```

Folgen Sie diesen Anweisungen, um jeden Befehl einzeln auszuführen.

Um zum von der Community verwalteten Homebrew Installationsprogramm zu wechseln

1. Deinstallieren Sie die AWS verwaltete Homebrew Version von AWS SAMCLI:

```
$ brew uninstall aws-sam-cli
```

2. Stellen Sie sicher, dass das deinstalliert AWS SAMCLI wurde:

```
$ which sam
sam not found
```

3. Entfernen Sie den AWS verwalteten AWS SAMCLI Wasserhahn:

```
$ brew untap aws/tap
```

Wenn Sie eine Fehlermeldung wie die folgende erhalten, fügen Sie die `--force` Option hinzu und versuchen Sie es erneut.

```
Error: Refusing to untap aws/tap because it contains the following installed
 formulae or casks:
aws-sam-cli-nightly
```

4. Entfernen Sie die zwischengespeicherten Dateien für das AWS verwaltete Installationsprogramm:

```
$ brew cleanup aws/tap
```

5. Update Homebrew und alle Formeln:

```
$ brew update
```

6. Installieren Sie die von der Community verwaltete Version von: AWS SAMCLI

```
$ brew install aws-sam-cli
```

7. Stellen Sie sicher, dass AWS SAMCLI das erfolgreich installiert wurde:

```
$ sam --version
SAM CLI, version 1.94.0
```

Fehlerbehebung

Falls Sie bei der Installation oder Verwendung von auf Fehler stoßen AWS SAMCLI, finden Sie weitere Informationen unterAWS SAMCLIProblembehandlung.

AWS Zugangsdaten einrichten

Für die AWS SAM Befehlszeilenschnittstelle (CLI) müssen Sie AWS Anmeldeinformationen festlegen, damit sie in Ihrem Namen AWS Dienste aufrufen kann. Zum Beispiel AWS SAMCLI ruft der Amazon S3 auf und CloudFormation.

Möglicherweise haben Sie bereits AWS Anmeldeinformationen für die Arbeit mit AWS Tools wie einem der AWS SDKs oder dem festgelegt AWS CLI. Falls nicht, finden Sie in diesem Thema die empfohlenen Vorgehensweisen zum Einrichten von AWS Anmeldeinformationen.

Um AWS Anmeldeinformationen festzulegen, benötigen Sie die Zugriffsschlüssel-ID und Ihren geheimen Zugriffsschlüssel für den IAM-Benutzer, den Sie konfigurieren möchten. Informationen zu Zugriffsschlüsseln IDs und geheimen Zugriffsschlüsseln finden Sie unter Managing Access Keys for IAM Users im IAM-Benutzerhandbuch.

Stellen Sie als Nächstes fest, ob Sie das AWS CLI installiert haben. Folgen Sie dann den Anweisungen in einem der folgenden Abschnitte:

Mit dem AWS CLI

Wenn Sie das AWS CLI installiert haben, verwenden Sie den `aws configure` Befehl und folgen Sie den Anweisungen:

```
$ aws configure
AWS Access Key ID [None]: your_access_key_id
AWS Secret Access Key [None]: your_secret_access_key
Default region name [None]:
Default output format [None]:
```

Informationen zu dem `aws configure` Befehl finden Sie unter Schnellkonfiguration von AWS CLI im AWS Command Line Interface Benutzerhandbuch.

Ich verwende nicht den AWS CLI

Wenn Sie das nicht AWS CLI installiert haben, können Sie entweder eine Anmeldeinformationsdatei erstellen oder Umgebungsvariablen festlegen:

- Anmeldeinformationsdatei — Sie können Anmeldeinformationen in der AWS Anmeldeinformationsdatei auf Ihrem lokalen System festlegen. Diese Datei muss sich an einem der folgenden Speicherorte befinden:
  - `~/.aws/credentials`unter Linux oder macOS
  - `C:\Users\`*`USERNAME`*`\.aws\credentials` (Windows)

  Diese Datei sollte Zeilen im folgenden Format enthalten:

  ```
  [default]
  aws_access_key_id = your_access_key_id
  aws_secret_access_key = your_secret_access_key
  ```

- Umgebungsvariablen — Sie können die Umgebungsvariablen `AWS_ACCESS_KEY_ID` und die `AWS_SECRET_ACCESS_KEY` Umgebungsvariablen festlegen.

  Verwenden Sie den Befehl export, um diese Variablen unter Linux oder macOS festzulegen:

  ```
  export AWS_ACCESS_KEY_ID=your_access_key_id
  export AWS_SECRET_ACCESS_KEY=your_secret_access_key
  ```

  Verwenden Sie den Befehl set, um diese Variablen unter Windows festzulegen:

  ```
  set AWS_ACCESS_KEY_ID=your_access_key_id
  set AWS_SECRET_ACCESS_KEY=your_secret_access_key
  ```

Telemetrie in der AWS SAMCLI

Bei AWS entwickeln und lancieren wir Dienstleistungen auf der Grundlage dessen, was wir aus Interaktionen mit Kunden lernen. Wir nutzen Kundenfeedback, um unser Produkt weiterzuentwickeln. Telemetrie ist eine zusätzliche Information, die uns hilft, die Bedürfnisse unserer Kunden besser zu verstehen, Probleme zu diagnostizieren und Funktionen bereitzustellen, die das Kundenerlebnis verbessern.

Die AWS SAM Befehlszeilenschnittstelle (CLI) sammelt Telemetriedaten, z. B. allgemeine Nutzungsmetriken, System- und Umgebungsinformationen sowie Fehler. Einzelheiten zu der Art der erfassten Telemetrie finden Sie unter Arten von erfassten Informationen.

Die AWS SAMCLI sammelt keine personenbezogenen Daten wie Benutzernamen oder E-Mail-Adressen. Außerdem werden keine sensiblen Informationen auf Projektebene extrahiert.

Kunden haben die Kontrolle darüber, ob die Telemetrie aktiviert ist, und sie können ihre Einstellungen jederzeit ändern. Wenn die Telemetrie aktiviert bleibt, werden Telemetriedaten im Hintergrund AWS SAMCLI gesendet, ohne dass eine zusätzliche Kundeninteraktion erforderlich ist.

Schalten Sie die Telemetrie für eine Sitzung aus

In macOS- und Linux-Betriebssystemen können Sie die Telemetrie für eine einzelne Sitzung ausschalten. Um die Telemetrie für Ihre aktuelle Sitzung zu deaktivieren, führen Sie den folgenden Befehl aus, um die Umgebungsvariable `SAM_CLI_TELEMETRY` auf `false` festzulegen. Wiederholen Sie den Befehl für jedes neue Terminal oder jede neue Sitzung.

```
export SAM_CLI_TELEMETRY=0
```

Telemetrie für Ihr Profil in allen Sitzungen ausschalten

Führen Sie die folgenden Befehle aus, um die Telemetrie für alle Sitzungen zu deaktivieren, wenn Sie das AWS SAMCLI auf Ihrem Betriebssystem ausführen.

Aktivieren von Telemetrie unter Linux

1. Führen Sie Folgendes aus:

   ```
   echo "export SAM_CLI_TELEMETRY=0" >>~/.profile
   ```

2. Führen Sie Folgendes aus:

   ```
   source ~/.profile
   ```

Aktivieren von Telemetrie unter macOS

1. Führen Sie Folgendes aus:

   ```
   echo "export SAM_CLI_TELEMETRY=0" >>~/.profile
   ```

2. Führen Sie Folgendes aus:

   ```
   source ~/.profile
   ```

Aktivieren von Telemetrie unter Windows

Mit dem folgenden Befehl können Sie die Umgebungsvariable vorübergehend für die Lebensdauer des Terminalfensters festlegen:

Wenn Sie die Befehlszeile verwenden:

```
set SAM_CLI_TELEMETRY=0
```

Bei Verwendung von PowerShell:

```
$env:SAM_CLI_TELEMETRY=0
```

Verwenden Sie den folgenden Befehl, um die Umgebungsvariable entweder in der Befehlszeile oder PowerShell dauerhaft festzulegen:

```
setx SAM_CLI_TELEMETRY 0
```

> **Note**
>
> Änderungen werden erst wirksam, wenn das Terminal geschlossen und erneut geöffnet wurde.

Arten von erfassten Informationen

- Nutzungsinformationen — Die generischen Befehle und Unterbefehle, die Kunden ausführen.
- Fehler und Diagnoseinformationen — Status und Dauer der Befehle, die Kunden ausführen, einschließlich Exit-Codes, Namen interner Ausnahmen und Fehler beim Herstellen einer Verbindung zu Docker.
- System- und Umgebungsinformationen — Die Python-Version, das Betriebssystem (Windows, Linux oder macOS), die Umgebung, in der der AWS SAMCLI ausgeführt wird (z. B. ein AWS IDE-Toolkit oder ein Terminal), und Hashwerte der Nutzungsattribute. AWS CodeBuild

Weitere Informationen

Die AWS SAMCLI gesammelten Telemetriedaten entsprechen den AWS Datenschutzrichtlinien. Weitere Informationen finden Sie hier:

- AWS Nutzungsbedingungen
- Häufig gestellte Fragen zum Datenschutz

AWS SAMCLIProblembehandlung

Dieser Abschnitt enthält Einzelheiten zur Behebung von Fehlermeldungen bei der Verwendung, Installation und Verwaltung der AWS Serverless Application Model Befehlszeilenschnittstelle (AWS SAMCLI).

Topics

- Fehlerbehebung
- Fehlermeldungen
- Warnmeldungen

Fehlerbehebung

Anleitungen zur Fehlerbehebung in Bezug auf AWS SAMCLI finden Sie unterBehebung von Installationsfehlern.

Fehlermeldungen

Curl-Fehler: „curl: (6) Konnte nicht auflösen:...“

Beim Versuch, den API Gateway Gateway-Endpunkt aufzurufen, wird der folgende Fehler angezeigt:

```
curl: (6) Could not resolve: endpointdomain (Domain name not found)
```

Das bedeutet, dass Sie versucht haben, eine Anfrage an eine ungültige Domain zu senden. Dies kann passieren, wenn Ihre serverlose Anwendung nicht erfolgreich bereitgestellt werden konnte oder wenn Sie einen Tippfehler in Ihrem curl Befehl haben. Stellen Sie mithilfe der CloudFormation Konsole oder des sicher, dass die Anwendung erfolgreich bereitgestellt wurde AWS CLI, und stellen Sie sicher, dass Ihr curl Befehl korrekt ist.

Fehler: Es können keine genauen Ressourceninformationen mit dem angegebenen Stack-Namen gefunden werden

Wenn `sam remote invoke` Sie den Befehl in einer Anwendung ausführen, die eine einzelne Lambda-Funktionsressource enthält, wird der folgende Fehler angezeigt:

```
Error: Can't find exact resource information with given <stack-name>. Please provide
 full resource ARN or --stack-name to resolve the ambiguity.
```

Mögliche Ursache: Sie haben die **`--stack-name`** Option nicht angegeben.

> Wenn ein Funktions-ARN nicht als Argument bereitgestellt wird, erfordert der `sam remote invoke` Befehl, dass die `--stack-name` Option angegeben wird.

Lösung: Geben Sie die **`--stack-name`** Option an.

Im Folgenden wird ein Beispiel gezeigt:

```
$ sam remote invoke --stack-name sam-app

Invoking Lambda Function HelloWorldFunction



START RequestId: 40593abb-e1ad-4d99-87bd-ac032e364e82 Version: $LATEST
END RequestId: 40593abb-e1ad-4d99-87bd-ac032e364e82
REPORT RequestId: 40593abb-e1ad-4d99-87bd-ac032e364e82  Duration: 11.31 ms
  Billed Duration: 12 ms  Memory Size: 128 MB     Max Memory Used: 67 MB  Init
 Duration: 171.71 ms
{"statusCode":200,"body":"{\"message\":\"hello world\"}"}%
```

Fehler: Ressourceninformationen können nicht aus dem Stacknamen gefunden werden

Wenn `sam remote invoke` Sie den Befehl ausführen und einen Lambda-Funktions-ARN als Argument übergeben, wird der folgende Fehler angezeigt:

```
Error: Can't find resource information from stack name (<stack-name>) and resource id
 (<function-id>)
```

Mögliche Ursache: Sie haben den Wert des Stack-Namens in Ihrer **`samconfig.toml`** Datei definiert.

Der AWS SAMCLI erste überprüft Ihre `samconfig.toml` Datei auf einen Stacknamen. Falls angegeben, wird das Argument als logischer ID-Wert übergeben.

Lösung: Übergeben Sie stattdessen die logische ID der Funktion.

Sie können die logische ID der Funktion als Argument anstelle des ARN der Funktion übergeben.

Lösung: Entfernen Sie den Wert des Stack-Namens aus Ihrer Konfigurationsdatei.

Sie können den Wert des Stack-Namens aus Ihrer Konfigurationsdatei entfernen. Dadurch wird verhindert, dass Ihre Funktion ARN als logischen ID-Wert übergeben wird. AWS SAMCLI

Führen Sie es aus, `sam build` nachdem Sie Ihre Konfigurationsdatei geändert haben.

Fehler: Verwaltete Ressourcen konnten nicht erstellt werden: Anmeldeinformationen konnten nicht gefunden werden

Beim Ausführen des sam deploy Befehls wird der folgende Fehler angezeigt:

```
Error: Failed to create managed resources: Unable to locate credentials
```

Das bedeutet, dass Sie keine AWS Anmeldeinformationen eingerichtet haben, mit denen AWS SAMCLI Sie AWS Serviceanfragen tätigen können. Um dieses Problem zu beheben, müssen Sie AWS Anmeldeinformationen einrichten. Weitere Informationen finden Sie unter AWS Zugangsdaten einrichten.

Fehler: FileNotFoundError in Windows

Wenn Sie Befehle AWS SAMCLI unter Windows ausführen, wird möglicherweise der folgende Fehler angezeigt:

```
Error: FileNotFoundError
```

Mögliche Ursache: Sie interagieren AWS SAMCLI möglicherweise mit Dateipfaden, die die maximale Pfadbeschränkung von Windows überschreiten.

Lösung: Um dieses Problem zu beheben, muss das neue Verhalten für lange Pfade aktiviert werden. Informationen dazu finden Sie unter Enable Long Paths in Windows 10, Version 1607 und höher in der Dokumentation zur Entwicklung von Microsoft Windows-Apps.

Fehler: Der Abhängigkeitsauflöser von pip...

Beispiel-Fehlertext:

```
ERROR: pip's dependency resolver does not currently take into account all the packages
 that are installed. This behaviour is the source of the following dependency
 conflicts.
aws-sam-cli 1.58.0 requires aws-sam-translator==1.51.0, but you have aws-sam-translator
 1.58.0 which is incompatible.
aws-sam-cli 1.58.0 requires typing-extensions==3.10.0.0, but you have typing-extensions
 4.4.0 which is incompatible.
```

Mögliche Ursache: Bei der Installation von Paketen können Abhängigkeiten zwischen Paketen zu Konflikten führen. pip

Jede Version des `aws-sam-cli` Pakets hängt von einer Version des `aws-sam-translator` Pakets ab. Zum Beispiel kann `aws-sam-cli` v1.58.0 von v1.51.0 abhängen`aws-sam-translator`.

Wenn Sie das Paket AWS SAMCLI verwenden und dann ein anderes Paket installierenpip, das von einer neueren Version von abhängt`aws-sam-translator`, passiert Folgendes:

- Die neuere Version von `aws-sam-translator` wird installiert.
- Die aktuelle Version von `aws-sam-cli` und die neuere Version von sind `aws-sam-translator` möglicherweise nicht kompatibel.
- Wenn Sie den verwenden AWS SAMCLI, tritt der Abhängigkeitsauflösungsfehler auf.

Lösungen:

1. Verwenden Sie das AWS SAMCLI native Paketinstallationsprogramm.
    a. Deinstallieren Sie das AWS SAMCLI mit Pip. Detaillierte Anweisungen finden Sie unter Deinstallation des AWS SAMCLI.
    b. Installieren Sie das AWS SAMCLI mit dem nativen Paketinstallationsprogramm. Detaillierte Anweisungen finden Sie unter Installieren Sie das AWS SAMCLI.
    c. Falls erforderlich, führen Sie ein Upgrade AWS SAMCLI mit dem systemeigenen Paketinstallationsprogramm durch. Detaillierte Anweisungen finden Sie unter Aktualisierung des AWS SAMCLI.
2. Wenn Sie die CLI verwenden müssenpip, empfehlen wir Ihnen, die AWS SAM CLI in einer virtuellen Umgebung zu installieren. Dies gewährleistet eine saubere Installationsumgebung und eine isolierte Umgebung, falls Fehler auftreten. Detaillierte Anweisungen finden Sie unter Installation des AWS SAMCLI in einer virtuellen Umgebung mit pip.

Fehler: Kein solcher Befehl 'remote'

Beim Ausführen des `sam remote invoke` Befehls wird der folgende Fehler angezeigt:

```
$ sam remote invoke ...
2023-06-20 08:15:07 Command remote not available
Usage: sam [OPTIONS] COMMAND [ARGS]...
Try 'sam -h' for help.
```

```
Error: No such command 'remote'.
```

Mögliche Ursache: Ihre Version von AWS SAMCLI ist veraltet.

Der AWS SAMCLI `sam remote invoke` Befehl wurde mit AWS SAMCLI Version 1.88.0 veröffentlicht. Sie können Ihre Version überprüfen, indem Sie den `sam --version` Befehl ausführen.

Lösung: Führen Sie ein Upgrade AWS SAMCLI auf die neueste Version durch.

Detaillierte Anweisungen finden Sie unter Aktualisierung des AWS SAMCLI.

Fehler: Das lokale Ausführen von AWS SAM-Projekten erfordertDocker. Hast du es installiert?

Beim Ausführen des sam local start-api Befehls wird der folgende Fehler angezeigt:

```
Error: Running AWS SAM projects locally requires Docker. Have you got it installed?
```

Das bedeutet, dass Sie es nicht Docker richtig installiert haben. Dockerist erforderlich, um Ihre Anwendung lokal zu testen. Um dieses Problem zu beheben, folgen Sie den Anweisungen zur Installation von Docker für Ihren Entwicklungshost. Weitere Informationen finden Sie unter Installieren von Docker.

Fehler: Sicherheitseinschränkungen nicht erfüllt

Beim Ausführen sam deploy --guided wird Ihnen die Frage angezeigt*`Function` may not have authorization defined, Is this okay? [y/N]*. Wenn Sie auf diese Aufforderung mit **N** (der Standardantwort) antworten, wird der folgende Fehler angezeigt:

```
Error: Security Constraints Not Satisfied
```

Die Aufforderung informiert Sie darüber, dass die Anwendung, die Sie bereitstellen möchten, möglicherweise über eine öffentlich zugängliche Amazon API Gateway, die ohne Autorisierung konfiguriert ist. Wenn Sie **N** auf diese Aufforderung antworten, sagen Sie, dass dies nicht in Ordnung ist.

Um dieses Problem zu beheben, haben Sie die folgenden Optionen:

- Konfigurieren Sie Ihre Anwendung mit Autorisierung. Informationen zur Konfiguration der Autorisierung finden Sie unterKontrollieren Sie den API-Zugriff mit Ihrer AWS SAM Vorlage.

- Wenn Sie beabsichtigen, einen öffentlich zugänglichen API-Endpunkt ohne Autorisierung einzurichten, starten Sie Ihre Bereitstellung neu und beantworten Sie diese Frage mit, **Y** um anzugeben, dass Sie mit der Bereitstellung einverstanden sind.

Nachricht: Fehlendes Authentifizierungstoken

Beim Versuch, den API Gateway Gateway-Endpunkt aufzurufen, wird der folgende Fehler angezeigt:

```
{"message":"Missing Authentication Token"}
```

Das bedeutet, dass Sie versucht haben, eine Anfrage an die richtige Domain zu senden, aber die URI ist nicht erkennbar. Um dieses Problem zu beheben, überprüfen Sie die vollständige URL und aktualisieren Sie den curl Befehl mit der richtigen URL.

Warnmeldungen

Warnung:... AWS wird das Homebrew Installationsprogramm für AWS SAM ... nicht mehr verwalten

Bei der Installation AWS SAMCLI von Homebrew Using wird die folgende Warnmeldung angezeigt:

```
Warning: ... AWS will no longer maintain the Homebrew installer for AWS SAM (aws/tap/
aws-sam-cli).
    For AWS supported installations, use the first party installers ...
```

Mögliche Ursache: Der Homebrew Support wird nicht AWS mehr aufrechterhalten.

Ab September 2023 AWS wird das Homebrew Installationsprogramm für das nicht mehr gewartet AWS SAMCLI.

Lösung: Verwenden Sie eine AWS unterstützte Installationsmethode.

- AWS Unterstützte Installationsmethoden finden Sie unterInstallieren Sie das AWS SAMCLI.

Lösung: Verwenden Sie das von der Community verwaltete InstallationsprogrammHomebrew, um die Nutzung fortzusetzen.

- Sie können das von der Community verwaltete Homebrew Installationsprogramm nach eigenem Ermessen verwenden. Detaillierte Anweisungen finden Sie unter Verwaltung des AWS SAMCLI mit Homebrew.

AWS SAM Steckverbinderreferenz

Dieser Abschnitt enthält Referenzinformationen für den Connector-Ressourcentyp AWS Serverless Application Model (AWS SAM). Eine Einführung in Konnektoren finden Sie unterVerwaltung von Ressourcenberechtigungen mit AWS SAM Steckverbinder.

Unterstützte Quell- und Zielressourcentypen für Konnektoren

Der `AWS::Serverless::Connector` Ressourcentyp unterstützt eine ausgewählte Anzahl von Verbindungen zwischen Quell- und Zielressourcen. Verwenden Sie bei der Konfiguration von Konnektoren in Ihrer AWS SAM Vorlage die folgende Tabelle, um auf unterstützte Verbindungen und die Eigenschaften zu verweisen, die für jeden Quell- und Zielressourcentyp definiert werden müssen. Weitere Informationen zur Konfiguration von Konnektoren in Ihrer Vorlage finden Sie unterAWS::Serverless::Connector.

Verwenden Sie die `Id` Eigenschaft sowohl für Quell- als auch für Zielressourcen, sofern sie in derselben Vorlage definiert sind. Optional `Qualifier` kann eine hinzugefügt werden, um den Umfang Ihrer definierten Ressource einzuschränken. Wenn sich die Ressource nicht in derselben Vorlage befindet, verwenden Sie eine Kombination unterstützter Eigenschaften.

Um neue Verbindungen anzufordern, reichen Sie ein neues Problem im serverless-application-model AWS GitHubRepository ein.

Source type (Quellentyp)	Zieltyp	Berechtigungen	Eigenschaften der Quelle	Eigenschaften des Ziels
`AWS::ApiGateway::RestApi`	`AWS::Lambda::Function`	`Write`	`Id`oder `Qualifier` `ResourceId` , und `Type`	`Id`oder `Arn` und `Type`
`AWS::ApiGateway::RestApi`	`AWS::Serverless::Function`	`Write`	`Id`oder `Qualifier` `ResourceId` , und `Type`	`Id`oder `Arn` und `Type`
`AWS::ApiGatewayV2::Api`	`AWS::Lambda::Function`	`Write`	`Id`oder `Qualifier`	`Id`oder `Arn` und `Type`

Source type (Quellentyp)	Zieltyp	Berechtigungen	Eigenschaften der Quelle	Eigenschaften des Ziels
			`ResourceId` , und `Type`	
`AWS::ApiGatewayV2::Api`	`AWS::Serverless::Function`	`Write`	`Id` oder `Qualifier` `ResourceId` , und `Type`	`Id` oder `Arn` und `Type`
`AWS::AppSync::DataSource`	`AWS::DynamoDB::Table`	`Read`	`Id` oder `RoleName` und `Type`	`Id` oder `Arn` und `Type`
`AWS::AppSync::DataSource`	`AWS::DynamoDB::Table`	`Write`	`Id` oder `RoleName` und `Type`	`Id` oder `Arn` und `Type`
`AWS::AppSync::DataSource`	`AWS::Events::EventBus`	`Write`	`Id` oder `RoleName` und `Type`	`Id` oder `Arn` und `Type`
`AWS::AppSync::DataSource`	`AWS::Lambda::Function`	`Write`	`Id` oder `RoleName` und `Type`	`Id` oder `Arn` und `Type`
`AWS::AppSync::DataSource`	`AWS::Serverless::Function`	`Write`	`Id` oder `RoleName` und `Type`	`Id` oder `Arn` und `Type`
`AWS::AppSync::DataSource`	`AWS::Serverless::SimpleTable`	`Read`	`Id` oder `RoleName` und `Type`	`Id` oder `Arn` und `Type`
`AWS::AppSync::DataSource`	`AWS::Serverless::SimpleTable`	`Write`	`Id` oder `RoleName` und `Type`	`Id` oder `Arn` und `Type`

Source type (Quellentyp)	Zieltyp	Berechtigungen	Eigenschaften der Quelle	Eigenschaften des Ziels
AWS::AppSync::GraphQLApi	AWS::Lambda::Function	Write	Idoder ResourceId und Type	Idoder Arn und Type
AWS::AppSync::GraphQLApi	AWS::Serverless::Function	Write	Idoder ResourceId und Type	Idoder Arn und Type
AWS::DynamoDB::Table	AWS::Lambda::Function	Read	Idoder Arn und Type	Idoder RoleName und Type
AWS::DynamoDB::Table	AWS::Serverless::Function	Read	Idoder Arn und Type	Idoder RoleName und Type
AWS::Events::Rule	AWS::Events::EventBus	Write	Idoder RoleName und Type	Idoder Arn und Type
AWS::Events::Rule	AWS::Lambda::Function	Write	Idoder Arn und Type	Idoder Arn und Type
AWS::Events::Rule	AWS::Serverless::Function	Write	Idoder Arn und Type	Idoder Arn und Type
AWS::Events::Rule	AWS::Serverless::StateMachine	Write	Idoder RoleName und Type	Idoder Arn und Type
AWS::Events::Rule	AWS::SNS::Topic	Write	Idoder Arn und Type	Idoder Arn und Type

Source type (Quellentyp)	Zieltyp	Berechtigungen	Eigenschaften der Quelle	Eigenschaften des Ziels
AWS::Events::Rule	AWS::SQS::Queue	Write	Idoder Arn und Type	Idoder ArnQueueUrl, und Type
AWS::Events::Rule	AWS::StepFunctions::StateMachine	Write	Idoder RoleName und Type	Idoder Arn und Type
AWS::Lambda::Function	AWS::DynamoDB::Table	Read, Write	Idoder RoleName und Type	Idoder Arn und Type
AWS::Lambda::Function	AWS::Events::EventBus	Write	Idoder RoleName und Type	Idoder Arn und Type
AWS::Lambda::Function	AWS::Lambda::Function	Write	Idoder RoleName und Type	Idoder Arn und Type
AWS::Lambda::Function	AWS::Location::PlaceIndex	Read	Idoder RoleName und Type	Idoder Arn und Type
AWS::Lambda::Function	AWS::S3::Bucket	Read, Write	Idoder RoleName und Type	Idoder Arn und Type
AWS::Lambda::Function	AWS::Serverless::Function	Write	Idoder RoleName und Type	Idoder Arn und Type

Source type (Quellentyp)	Zieltyp	Berechtigungen	Eigenschaften der Quelle	Eigenschaften des Ziels
AWS::Lambda::Function	AWS::Serverless::SimpleTable	Read, Write	Idoder RoleName und Type	Idoder Arn und Type
AWS::Lambda::Function	AWS::Serverless::StateMachine	Read, Write	Idoder RoleName und Type	Idoder ArnName, und Type
AWS::Lambda::Function	AWS::SNS::Topic	Write	Idoder RoleName und Type	Idoder Arn und Type
AWS::Lambda::Function	AWS::SQS::Queue	Read, Write	Idoder RoleName und Type	Idoder Arn und Type
AWS::Lambda::Function	AWS::StepFunctions::StateMachine	Read, Write	Idoder RoleName und Type	Idoder ArnName, und Type
AWS::S3::Bucket	AWS::Lambda::Function	Write	Idoder Arn und Type	Idoder Arn und Type
AWS::S3::Bucket	AWS::Serverless::Function	Write	Idoder Arn und Type	Idoder Arn und Type
AWS::Serverless::Api	AWS::Lambda::Function	Write	Idoder Qualifier ResourceId , und Type	Idoder Arn und Type

Source type (Quellentyp)	Zieltyp	Berechtigungen	Eigenschaften der Quelle	Eigenschaften des Ziels
AWS::Serverless::Api	AWS::Serverless::Function	Write	Idoder Qualifier ResourceId , und Type	Idoder Arn und Type
AWS::Serverless::Function	AWS::DynamoDB::Table	Read, Write	Idoder RoleName und Type	Idoder Arn und Type
AWS::Serverless::Function	AWS::Events::EventBus	Write	Idoder RoleName und Type	Idoder Arn und Type
AWS::Serverless::Function	AWS::Lambda::Function	Write	Idoder RoleName und Type	Idoder Arn und Type
AWS::Serverless::Function	AWS::S3::Bucket	Read, Write	Idoder RoleName und Type	Idoder Arn und Type
AWS::Serverless::Function	AWS::Serverless::Function	Write	Idoder RoleName und Type	Idoder Arn und Type
AWS::Serverless::Function	AWS::Serverless::SimpleTable	Read, Write	Idoder RoleName und Type	Idoder Arn und Type
AWS::Serverless::Function	AWS::Serverless::StateMachine	Read, Write	Idoder RoleName und Type	Idoder ArnName, und Type

Source type (Quellentyp)	Zieltyp	Berechtigungen	Eigenschaften der Quelle	Eigenschaften des Ziels
AWS::Serverless::Function	AWS::SNS::Topic	Write	Idoder RoleName und Type	Idoder Arn und Type
AWS::Serverless::Function	AWS::SQS::Queue	Read, Write	Idoder RoleName und Type	Idoder Arn und Type
AWS::Serverless::Function	AWS::StepFunctions::StateMachine	Read, Write	Idoder RoleName und Type	Idoder ArnName, und Type
AWS::Serverless::HttpApi	AWS::Lambda::Function	Write	Idoder Qualifier ResourceId , und Type	Idoder Arn und Type
AWS::Serverless::HttpApi	AWS::Serverless::Function	Write	Idoder Qualifier ResourceId , und Type	Idoder Arn und Type
AWS::Serverless::SimpleTable	AWS::Lambda::Function	Read	Idoder Arn und Type	Idoder RoleName und Type
AWS::Serverless::SimpleTable	AWS::Serverless::Function	Read	Idoder Arn und Type	Idoder RoleName und Type
AWS::Serverless::StateMachine	AWS::DynamoDB::Table	Read, Write	Idoder RoleName und Type	Idoder Arn und Type

Source type (Quellentyp)	Zieltyp	Berechtigungen	Eigenschaften der Quelle	Eigenschaften des Ziels
AWS::Serverless::StateMachine	AWS::Events::EventBus	Write	Idoder RoleName und Type	Idoder Arn und Type
AWS::Serverless::StateMachine	AWS::Lambda::Function	Write	Idoder RoleName und Type	Idoder Arn und Type
AWS::Serverless::StateMachine	AWS::S3::Bucket	Read, Write	Idoder RoleName und Type	Idoder Arn und Type
AWS::Serverless::StateMachine	AWS::Serverless::Function	Write	Idoder RoleName und Type	Idoder Arn und Type
AWS::Serverless::StateMachine	AWS::Serverless::SimpleTable	Read, Write	Idoder RoleName und Type	Idoder Arn und Type
AWS::Serverless::StateMachine	AWS::Serverless::StateMachine	Read, Write	Idoder RoleName und Type	Idoder ArnName, und Type
AWS::Serverless::StateMachine	AWS::SNS::Topic	Write	Idoder RoleName und Type	Idoder Arn und Type

Source type (Quellentyp)	Zieltyp	Berechtigungen	Eigenschaften der Quelle	Eigenschaften des Ziels
AWS::Serverless::StateMachine	AWS::SQS::Queue	Write	Idoder RoleName und Type	Idoder Arn und Type
AWS::Serverless::StateMachine	AWS::StepFunctions::StateMachine	Read, Write	Idoder RoleName und Type	Idoder ArnName, und Type
AWS::SNS::Topic	AWS::Lambda::Function	Write	Idoder Arn und Type	Idoder Arn und Type
AWS::SNS::Topic	AWS::Serverless::Function	Write	Idoder Arn und Type	Idoder Arn und Type
AWS::SNS::Topic	AWS::SQS::Queue	Write	Idoder Arn und Type	Idoder ArnQueueUrl, und Type
AWS::SQS::Queue	AWS::Lambda::Function	Read, Write	Idoder Arn und Type	Idoder RoleName und Type
AWS::SQS::Queue	AWS::Serverless::Function	Read, Write	Idoder Arn und Type	Idoder RoleName und Type
AWS::StepFunctions::StateMachine	AWS::DynamoDB::Table	Read, Write	Idoder RoleName und Type	Idoder Arn und Type

Source type (Quellentyp)	Zieltyp	Berechtigungen	Eigenschaften der Quelle	Eigenschaften des Ziels
AWS::StepFunctions::StateMachine	AWS::Events::EventBus	Write	Idoder RoleName und Type	Idoder Arn und Type
AWS::StepFunctions::StateMachine	AWS::Lambda::Function	Write	Idoder RoleName und Type	Idoder Arn und Type
AWS::StepFunctions::StateMachine	AWS::S3::Bucket	Read, Write	Idoder RoleName und Type	Idoder Arn und Type
AWS::StepFunctions::StateMachine	AWS::Serverless::Function	Write	Idoder RoleName und Type	Idoder Arn und Type
AWS::StepFunctions::StateMachine	AWS::Serverless::SimpleTable	Read, Write	Idoder RoleName und Type	Idoder Arn und Type
AWS::StepFunctions::StateMachine	AWS::Serverless::StateMachine	Read, Write	Idoder RoleName und Type	Idoder ArnName, und Type
AWS::StepFunctions::StateMachine	AWS::SNS::Topic	Write	Idoder RoleName und Type	Idoder Arn und Type

Source type (Quellentyp)	Zieltyp	Berechtigungen	Eigenschaften der Quelle	Eigenschaften des Ziels
`AWS::StepFunctions::StateMachine`	`AWS::SQS::Queue`	`Write`	`Id`oder `RoleName` und `Type`	`Id`oder `Arn` und `Type`
`AWS::StepFunctions::StateMachine`	`AWS::StepFunctions::StateMachine`	`Read`, `Write`	`Id`oder `RoleName` und `Type`	`Id`oder `ArnName`, und `Type`

Von Konnektoren erstellte IAM-Richtlinien

In diesem Abschnitt werden die AWS Identity and Access Management (IAM-) Richtlinien dokumentiert, die AWS SAM bei der Verwendung von Connectoren erstellt werden.

`AWS::DynamoDB::Table` auf `AWS::Lambda::Function`

Art der Richtlinie

Der `AWS::Lambda::Function` Rolle ist eine vom Kunden verwaltete Richtlinie zugeordnet.

Zugriffskategorien

`Read`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:DescribeStream",
        "dynamodb:GetRecords",
        "dynamodb:GetShardIterator",
        "dynamodb:ListStreams"
      ],
      "Resource": [
        "%{Source.Arn}/stream/*"
```

```
      ]
    }
  ]
}
```

`AWS::Events::Rule` auf `AWS::SNS::Topic`

Art der Richtlinie

`AWS::SNS::TopicPolicy` angehängt an den `AWS::SNS::Topic`.

Zugriffskategorien

`Write`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "events.amazonaws.com"
      },
      "Resource": "%{Destination.Arn}",
      "Action": "sns:Publish",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "%{Source.Arn}"
        }
      }
    }
  ]
}
```

`AWS::Events::Rule` auf `AWS::Events::EventBus`

Art der Richtlinie

Der `AWS::Events::Rule` Rolle ist eine vom Kunden verwaltete Richtlinie zugeordnet.

Zugriffskategorien

`Write`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:PutEvents"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

`AWS::Events::Rule` auf `AWS::StepFunctions::StateMachine`

Art der Richtlinie

Der `AWS::Events::Rule` Rolle ist eine vom Kunden verwaltete Richtlinie zugeordnet.

Zugriffskategorien

`Write`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:StartExecution"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

`AWS::Events::Rule` auf `AWS::Lambda::Function`

Art der Richtlinie

`AWS::Lambda::Permission`angehängt an den`AWS::Lambda::Function`.

Zugriffskategorien

`Write`

```
{
  "Action": "lambda:InvokeFunction",
  "Principal": "events.amazonaws.com",
  "SourceArn": "%{Source.Arn}"
}
```

`AWS::Events::Rule` auf `AWS::SQS::Queue`

Art der Richtlinie

`AWS::SQS::QueuePolicy`angehängt an den`AWS::SQS::Queue`.

Zugriffskategorien

`Write`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "events.amazonaws.com"
      },
      "Resource": "%{Destination.Arn}",
      "Action": "sqs:SendMessage",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "%{Source.Arn}"
        }
      }
    }
  ]
}
```

`AWS::Lambda::Function` auf `AWS::Lambda::Function`

Art der Richtlinie

Der `AWS::Lambda::Function` Rolle ist eine vom Kunden verwaltete Richtlinie zugeordnet.

Zugriffskategorien

`Write`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeAsync",
        "lambda:InvokeFunction"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

`AWS::Lambda::Function` auf `AWS::S3::Bucket`

Art der Richtlinie

Der `AWS::Lambda::Function` Rolle ist eine vom Kunden verwaltete Richtlinie zugeordnet.

Zugriffskategorien

`Read`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectAcl",
        "s3:GetObjectLegalHold",
        "s3:GetObjectRetention",
        "s3:GetObjectTorrent",
        "s3:GetObjectVersion",
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionForReplication",
        "s3:GetObjectVersionTorrent",
```

```
        "s3:ListBucket",
        "s3:ListBucketMultipartUploads",
        "s3:ListBucketVersions",
        "s3:ListMultipartUploadParts"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}/*"
      ]
    }
  ]
}
```

`Write`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:AbortMultipartUpload",
        "s3:DeleteObject",
        "s3:DeleteObjectVersion",
        "s3:PutObject",
        "s3:PutObjectLegalHold",
        "s3:PutObjectRetention",
        "s3:RestoreObject"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}/*"
      ]
    }
  ]
}
```

`AWS::Lambda::Function` auf `AWS::DynamoDB::Table`

Art der Richtlinie

Der `AWS::Lambda::Function` Rolle ist eine vom Kunden verwaltete Richtlinie zugeordnet.

Zugriffskategorien

Read

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:Query",
        "dynamodb:Scan",
        "dynamodb:BatchGetItem",
        "dynamodb:ConditionCheckItem",
        "dynamodb:PartiQLSelect"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}/index/*"
      ]
    }
  ]
}
```

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:PutItem",
        "dynamodb:UpdateItem",
        "dynamodb:DeleteItem",
        "dynamodb:BatchWriteItem",
        "dynamodb:PartiQLDelete",
        "dynamodb:PartiQLInsert",
        "dynamodb:PartiQLUpdate"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}/index/*"
      ]
    }
  ]
```

```
}
```

`AWS::Lambda::Function` auf `AWS::SQS::Queue`

Art der Richtlinie

Der `AWS::Lambda::Function` Rolle ist eine vom Kunden verwaltete Richtlinie zugeordnet.

Zugriffskategorien

`Read`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sqs:ReceiveMessage",
        "sqs:GetQueueAttributes"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

`Write`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sqs:DeleteMessage",
        "sqs:SendMessage",
        "sqs:ChangeMessageVisibility",
        "sqs:PurgeQueue"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
```

```
  ]
}
```

`AWS::Lambda::Function` auf `AWS::SNS::Topic`

Art der Richtlinie

Der `AWS::Lambda::Function` Rolle ist eine vom Kunden verwaltete Richtlinie zugeordnet.

Zugriffskategorien

`Write`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

`AWS::Lambda::Function` auf `AWS::StepFunctions::StateMachine`

Art der Richtlinie

Der `AWS::Lambda::Function` Rolle ist eine vom Kunden verwaltete Richtlinie zugeordnet.

Zugriffskategorien

`Write`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:StartExecution",
```

```
        "states:StartSyncExecution"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "states:StopExecution"
      ],
      "Resource": [
        "arn:${AWS::Partition}:states:${AWS::Region}:${AWS::AccountId}:execution:
%{Destination.Name}:*"
      ]
    }
  ]
}
```

Read

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:DescribeStateMachine",
        "states:ListExecutions"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "states:DescribeExecution",
        "states:DescribeStateMachineForExecution",
        "states:GetExecutionHistory"
      ],
      "Resource": [
        "arn:${AWS::Partition}:states:${AWS::Region}:${AWS::AccountId}:execution:
%{Destination.Name}:*"
```

```
      ]
    }
  ]
}
```

`AWS::Lambda::Function` auf `AWS::Events::EventBus`

Art der Richtlinie

Der `AWS::Lambda::Function` Rolle ist eine vom Kunden verwaltete Richtlinie zugeordnet.

Zugriffskategorien

`Write`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:PutEvents"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

`AWS::Lambda::Function` auf `AWS::Location::PlaceIndex`

Art der Richtlinie

Der `AWS::Lambda::Function` Rolle ist eine vom Kunden verwaltete Richtlinie zugeordnet.

Zugriffskategorien

`Read`

```
{
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Action": [
        "geo:DescribePlaceIndex",
        "geo:GetPlace",
        "geo:SearchPlaceIndexForPosition",
        "geo:SearchPlaceIndexForSuggestions",
        "geo:SearchPlaceIndexForText"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

`AWS::ApiGatewayV2::Api` auf `AWS::Lambda::Function`

Art der Richtlinie

`AWS::Lambda::Permission`angehängt an den`AWS::Lambda::Function`.

Zugriffskategorien

`Write`

```
{
  "Action": "lambda:InvokeFunction",
  "Principal": "apigateway.amazonaws.com",
  "SourceArn": "arn:${AWS::Partition}:execute-api:${AWS::Region}:${AWS::AccountId}:
%{Source.ResourceId}/%{Source.Qualifier}"
}
```

`AWS::ApiGateway::RestApi` auf `AWS::Lambda::Function`

Art der Richtlinie

`AWS::Lambda::Permission`angehängt an den`AWS::Lambda::Function`.

Zugriffskategorien

`Write`

```
{
  "Action": "lambda:InvokeFunction",
  "Principal": "apigateway.amazonaws.com",
```

```
  "SourceArn": "arn:${AWS::Partition}:execute-api:${AWS::Region}:${AWS::AccountId}:
%{Source.ResourceId}/%{Source.Qualifier}"
}
```

`AWS::SNS::Topic` auf `AWS::SQS::Queue`

Art der Richtlinie

`AWS::SQS::QueuePolicy`angehängt an den`AWS::SQS::Queue`.

Zugriffskategorien

`Write`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.amazonaws.com"
      },
      "Resource": "%{Destination.Arn}",
      "Action": "sqs:SendMessage",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "%{Source.Arn}"
        }
      }
    }
  ]
}
```

`AWS::SNS::Topic` auf `AWS::Lambda::Function`

Art der Richtlinie

`AWS::Lambda::Permission`angehängt an den`AWS::Lambda::Function`.

Zugriffskategorien

`Write`

```
{
  "Action": "lambda:InvokeFunction",
```

```
  "Principal": "sns.amazonaws.com",
  "SourceArn": "%{Source.Arn}"
}
```

`AWS::SQS::Queue` auf `AWS::Lambda::Function`

Art der Richtlinie

Der `AWS::Lambda::Function` Rolle ist eine vom Kunden verwaltete Richtlinie zugeordnet.

Zugriffskategorien

`Write`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sqs:DeleteMessage"
      ],
      "Resource": [
        "%{Source.Arn}"
      ]
    }
  ]
}
```

`Read`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sqs:ReceiveMessage",
        "sqs:GetQueueAttributes"
      ],
      "Resource": [
        "%{Source.Arn}"
      ]
    }
  ]
```

```
}
```

`AWS::S3::Bucket` auf `AWS::Lambda::Function`

Art der Richtlinie

`AWS::Lambda::Permission`angehängt an den`AWS::Lambda::Function`.

Zugriffskategorien

`Write`

```
{
  "Action": "lambda:InvokeFunction",
  "Principal": "s3.amazonaws.com",
  "SourceArn": "%{Source.Arn}",
  "SourceAccount": "${AWS::AccountId}"
}
```

`AWS::StepFunctions::StateMachine` auf `AWS::Lambda::Function`

Art der Richtlinie

Der `AWS::StepFunctions::StateMachine` Rolle ist eine vom Kunden verwaltete Richtlinie zugeordnet.

Zugriffskategorien

`Write`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeAsync",
        "lambda:InvokeFunction"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
```

```
}
```

`AWS::StepFunctions::StateMachine` auf `AWS::SNS::Topic`

Art der Richtlinie

Der `AWS::StepFunctions::StateMachine` Rolle ist eine vom Kunden verwaltete Richtlinie zugeordnet.

Zugriffskategorien

`Write`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

`AWS::StepFunctions::StateMachine` auf `AWS::SQS::Queue`

Art der Richtlinie

Der `AWS::StepFunctions::StateMachine` Rolle ist eine vom Kunden verwaltete Richtlinie zugeordnet.

Zugriffskategorien

`Write`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
```

```
        "sqs:SendMessage"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

`AWS::StepFunctions::StateMachine` auf `AWS::S3::Bucket`

Art der Richtlinie

Der `AWS::StepFunctions::StateMachine` Rolle ist eine vom Kunden verwaltete Richtlinie zugeordnet.

Zugriffskategorien

`Read`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectAcl",
        "s3:GetObjectLegalHold",
        "s3:GetObjectRetention",
        "s3:GetObjectTorrent",
        "s3:GetObjectVersion",
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionForReplication",
        "s3:GetObjectVersionTorrent",
        "s3:ListBucket",
        "s3:ListBucketMultipartUploads",
        "s3:ListBucketVersions",
        "s3:ListMultipartUploadParts"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}/*"
      ]
    }
```

```
  ]
}
```

`Write`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:AbortMultipartUpload",
        "s3:DeleteObject",
        "s3:DeleteObjectVersion",
        "s3:PutObject",
        "s3:PutObjectLegalHold",
        "s3:PutObjectRetention",
        "s3:RestoreObject"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}/*"
      ]
    }
  ]
}
```

`AWS::StepFunctions::StateMachine` auf `AWS::DynamoDB::Table`

Art der Richtlinie

Der `AWS::StepFunctions::StateMachine` Rolle ist eine vom Kunden verwaltete Richtlinie zugeordnet.

Zugriffskategorien

`Read`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:GetItem",
```

```
        "dynamodb:Query",
        "dynamodb:Scan",
        "dynamodb:BatchGetItem",
        "dynamodb:ConditionCheckItem",
        "dynamodb:PartiQLSelect"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}/index/*"
      ]
    }
  ]
}
```

`Write`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:PutItem",
        "dynamodb:UpdateItem",
        "dynamodb:DeleteItem",
        "dynamodb:BatchWriteItem",
        "dynamodb:PartiQLDelete",
        "dynamodb:PartiQLInsert",
        "dynamodb:PartiQLUpdate"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}/index/*"
      ]
    }
  ]
}
```

`AWS::StepFunctions::StateMachine` auf `AWS::StepFunctions::StateMachine`

Art der Richtlinie

Der `AWS::StepFunctions::StateMachine` Rolle ist eine vom Kunden verwaltete Richtlinie zugeordnet.

Zugriffskategorien

Read

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:DescribeExecution"
      ],
      "Resource": [
        "arn:${AWS::Partition}:states:${AWS::Region}:${AWS::AccountId}:execution:
%{Destination.Name}:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "events:DescribeRule"
      ],
      "Resource": [
        "arn:${AWS::Partition}:events:${AWS::Region}:${AWS::AccountId}:rule/
StepFunctionsGetEventsForStepFunctionsExecutionRule"
      ]
    }
  ]
}
```

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:StartExecution"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    },
    {
```

```
      "Effect": "Allow",
      "Action": [
        "states:StopExecution"
      ],
      "Resource": [
        "arn:${AWS::Partition}:states:${AWS::Region}:${AWS::AccountId}:execution:
%{Destination.Name}:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "events:PutTargets",
        "events:PutRule"
      ],
      "Resource": [
        "arn:${AWS::Partition}:events:${AWS::Region}:${AWS::AccountId}:rule/
StepFunctionsGetEventsForStepFunctionsExecutionRule"
      ]
    }
  ]
}
```

`AWS::StepFunctions::StateMachine` auf `AWS::Events::EventBus`

Art der Richtlinie

Der `AWS::StepFunctions::StateMachine` Rolle ist eine vom Kunden verwaltete Richtlinie zugeordnet.

Zugriffskategorien

`Write`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:PutEvents"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
```

```
    }
  ]
}
```

`AWS::AppSync::DataSource` auf `AWS::DynamoDB::Table`

Art der Richtlinie

Der `AWS::AppSync::DataSource` Rolle ist eine vom Kunden verwaltete Richtlinie zugeordnet.

Zugriffskategorien

`Read`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:Query",
        "dynamodb:Scan",
        "dynamodb:BatchGetItem",
        "dynamodb:ConditionCheckItem",
        "dynamodb:PartiQLSelect"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}/index/*"
      ]
    }
  ]
}
```

`Write`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:PutItem",
        "dynamodb:UpdateItem",
```

```
        "dynamodb:DeleteItem",
        "dynamodb:BatchWriteItem",
        "dynamodb:PartiQLDelete",
        "dynamodb:PartiQLInsert",
        "dynamodb:PartiQLUpdate"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}/index/*"
      ]
    }
  ]
}
```

`AWS::AppSync::DataSource` auf `AWS::Lambda::Function`

Art der Richtlinie

Der `AWS::AppSync::DataSource` Rolle ist eine vom Kunden verwaltete Richtlinie zugeordnet.

Zugriffskategorien

`Write`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeAsync",
        "lambda:InvokeFunction"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}:*"
      ]
    }
  ]
}
```

`AWS::AppSync::DataSource` auf `AWS::Events::EventBus`

Art der Richtlinie

Der `AWS::AppSync::DataSource` Rolle ist eine vom Kunden verwaltete Richtlinie zugeordnet.

Zugriffskategorien

`Write`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:PutEvents"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

`AWS::AppSync::GraphQLApi` auf `AWS::Lambda::Function`

Art der Richtlinie

`AWS::Lambda::Permission`angehängt an den`AWS::Lambda::Function`.

Zugriffskategorien

`Write`

```
{
  "Action": "lambda:InvokeFunction",
  "Principal": "appsync.amazonaws.com",
  "SourceArn": "arn:${AWS::Partition}:appsync:${AWS::Region}:${AWS::AccountId}:apis/
%{Source.ResourceId}"
}
```

Installation von Docker zur Verwendung mit dem AWS SAMCLI

Dockerist eine Anwendung, die Container auf Ihrem Computer ausführt. Mit AWS SAM kann eine lokale Umgebung bereitgestellt werdenDocker, die einem Container ähnelt, um Ihre serverlosen Anwendungen zu erstellen, zu testen und zu debuggen. AWS Lambda

> **Note**
>
> Dockerist nur erforderlich, um Ihre Anwendungen lokal zu testen und mithilfe dieser Option Bereitstellungspakete zu erstellen. `--use-container`

Topics

- Installation von Docker
- Nächste Schritte

Installation von Docker

Folgen Sie diesen Anweisungen, um die Installation Docker auf Ihrem Betriebssystem durchzuführen.

Linux

Docker ist auf vielen verschiedenen Betriebssystemen verfügbar, einschließlich der meisten modernen Linux-Distributionen wie CentOSDebian, und. Ubuntu Informationen zur Installation Docker auf Ihrem speziellen Betriebssystem finden Sie unter Get Docker auf der Docker Docs-Website.

Zur Installation Docker auf Amazon Linux 2 oder Amazon Linux 2023

1. Aktualisieren Sie die installierten Pakete und den Cache der Paketverwaltung auf Ihrer Instance.

   ```
   $ sudo yum update -y
   ```

2. Installieren Sie das neueste Docker Community Edition-Paket.

   - Führen Sie für Amazon Linux 2 Folgendes aus:

     ```
     $ sudo amazon-linux-extras install docker
     ```

   - Führen Sie für Amazon Linux 2023 Folgendes aus:

     ```
     $ sudo yum install -y docker
     ```

3. Starten Sie den Service Docker.

   ```
   $ sudo service docker start
   ```

4. Fügen Sie der `docker` Gruppe das `ec2-user` hinzu, damit Sie Docker Befehle ausführen können, ohne zu verwendensudo.

   ```
   $ sudo usermod -a -G docker ec2-user
   ```

5. Holen Sie sich die neuen `docker` Gruppenberechtigungen, indem Sie sich abmelden und erneut anmelden. Schließen Sie dazu Ihr aktuelles SSH-Terminalfenster und stellen Sie in einer neuen Instanz erneut eine Verbindung zu Ihrer Instance her. Ihre neue SSH-Sitzung sollte über die entsprechenden `docker` Gruppenberechtigungen verfügen.

6. Stellen Sie sicher, dass die Docker-Befehle ausführen `ec2-user` können, ohne sie zu verwenden. sudo

   ```
   $ docker ps
   ```

   Sie sollten die folgende Ausgabe sehen, die bestätigt, dass Docker installiert ist und läuft:

   ```
   CONTAINER ID        IMAGE               COMMAND             CREATED
   STATUS              PORTS               NAMES
   ```

> **Note**
>
> Um unter Linux Lambda-Funktionen mit einer anderen Befehlssatzarchitektur als Ihrem Host-Computer zu erstellen und auszuführen, müssen Sie zusätzliche Schritte konfigurierenDocker. Um beispielsweise `arm64` Funktionen auf einem `x86_64` Computer auszuführen, können Sie den folgenden Befehl ausführen, um den Docker Daemon zu konfigurieren:. `docker run --rm --privileged multiarch/qemu-user-static --reset -p yes`

Falls bei der Installation Probleme auftretenDocker, finden Sie weitere Informationen unterBehebung von Installationsfehlern. Oder lesen Sie auf der Docker Docs-Website den Abschnitt zur Fehlerbehebung unter Schritte nach der Installation für Linux.

macOS

> **Note**
>
> DockerDesktop wird offiziell unterstützt, aber ab AWS SAMCLI Version 1.47.0 können Sie Alternativen verwenden, sofern diese die Runtime verwenden. Docker

1. Installieren Docker

   Das AWS SAMCLI unterstützt die Docker Ausführung auf macOS Sierra 10.12 oder höher. Informationen zur Installation finden Sie Docker unter DockerDesktop für Mac installieren auf der Docker Docs-Website.
2. Konfigurieren Sie Ihre gemeinsam genutzten Laufwerke

   Das AWS SAMCLI setzt voraus, dass das Projektverzeichnis oder ein beliebiges übergeordnetes Verzeichnis in einem gemeinsam genutzten Laufwerk aufgeführt ist. Dies kann die gemeinsame Nutzung von Dateien erfordern. Weitere Informationen finden Sie in der DockerDokumentation unter dem Thema Volume-Mount erfordert Dateifreigabe zur Problembehebung.
3. Überprüfen der Installation

   Stellen Sie nach Docker der Installation sicher, dass es funktioniert. Stellen Sie außerdem sicher, dass Sie Docker Befehle von der Befehlszeile aus ausführen können (z. B.`docker ps`). Sie müssen keine Container installieren, abrufen oder abrufen — dies wird bei Bedarf automatisch AWS SAMCLI erledigt.

Falls bei der Installation Probleme auftretenDocker, finden Sie weitere Tipps zur Problembehebung auf der Docs-Website im Abschnitt Problembehandlung und Diagnose. Docker

Windows

> **Note**
>
> AWS SAM unterstützt offiziell Docker Desktop. Ab AWS SAMCLI Version 1.47.0 können Sie jedoch Alternativen verwenden, sofern diese die Docker Runtime verwenden.

1. Installieren Docker.

DockerDesktop unterstützt das neueste Windows-Betriebssystem. Für ältere Versionen von Windows ist die Docker Toolbox verfügbar. Wählen Sie Ihre Version von Windows für die richtigen Docker Installationsschritte:

- Informationen zur Installation Docker für Windows 10 finden Sie auf der Docker Docs-Website unter Docker Desktop für Windows installieren.
- Informationen zur Installation Docker für frühere Versionen von Windows finden Sie unter The Docker Toolbox im Docker GitHub Toolbox-Repository.

2. Konfigurieren Sie Ihre gemeinsam genutzten Laufwerke.

   Das AWS SAMCLI setzt voraus, dass das Projektverzeichnis oder ein beliebiges übergeordnetes Verzeichnis in einem gemeinsam genutzten Laufwerk aufgeführt ist. In einigen Fällen müssen Sie Ihr Laufwerk teilen, Docker damit es ordnungsgemäß funktioniert.

3. Überprüfen Sie die Installation.

   Stellen Sie nach Docker der Installation sicher, dass es funktioniert. Stellen Sie außerdem sicher, dass Sie Docker Befehle von der Befehlszeile aus ausführen können (z. B.docker ps). Sie müssen keine Container installieren, abrufen oder abrufen — dies wird bei Bedarf automatisch AWS SAMCLI erledigt.

Falls bei der Installation Probleme auftretenDocker, finden Sie weitere Tipps zur Problembehebung auf der Docs-Website im Abschnitt Problembehandlung und Diagnose. Docker

Nächste Schritte

Informationen zur Installation von finden AWS SAMCLI Sie unterInstallieren Sie das AWS SAMCLI.

Installation Finch zur Verwendung mit der AWS SAM CLI

Finch AWS ist ein Container-Entwicklungstool, das eine Alternative zu Docker MacOS- und Linux-Systemen bietet und besonders vorteilhaft ist. Weitere Informationen finden Sie unter Finch unter GitHub. Mit AWS SAM kann eine lokale Umgebung bereitgestellt werdenFinch, die einem Container AWS Lambda zum Erstellen, Testen und Debuggen Ihrer serverlosen Anwendungen ähnelt.

> **Note**
>
> Sie müssen Finch Ihre Anwendungen lokal testen, Bereitstellungspakete mithilfe der `--use-container` Option erstellen und OCI-Image-Funktionen erstellen. Ab AWS SAM CLI-Version 1.145.0 unterstützt AWS SAM CLI Finch als alternative Container-Laufzeit zu. Docker

Topics

- Installation von Finch
- Konfiguration Finch mit AWS SAM CLI
- Die Installation wird überprüft
- Stoppen von Finch
- Fehlerbehebung
- Nächste Schritte

Installation von Finch

Folgen Sie diesen Anweisungen, um die Installation Finch auf Ihrem Betriebssystem durchzuführen.

Voraussetzungen

> **Note**
>
> FinchDie Integration in AWS SAM CLI ist auf macOS- und Linux-Systemen verfügbar. Windows-Benutzer sollten aus Finch Funktionsgründen die Verwendung einer virtuellen Maschine oder einer Linux-Umgebung in Betracht ziehen.

macOS-Versionen:

- macOS 15 Sequoia
- macOS 14 Sonoma

> **Note**
>
> Finchunterstützt die neuesten beiden Versionen von macOS.

Linux:

- Linux Kernel v4.x +

Finch installieren

macOS

Installieren Sie Finch unter macOS wie folgt

1. Installieren Sie Finch mit der `brew` CLI. Wenn Sie dazu aufgefordert werden, geben Sie Ihr macOS-Passwort ein.

   ```
   $ brew install finch
   ```

2. Überprüfen Sie die installierte Finch Version.

   ```
   $ finch --version
   ```

3. Initialisieren Sie die Finch virtuelle Maschine (Erstinstallation).

   ```
   $ finch vm init
   ```

4. Starten Sie die Finch virtuelle Maschine.

   ```
   $ finch vm start
   ```

5. Überprüfen Sie den Status der Finch virtuellen Maschine.

   ```
   $ finch vm status
   ```

6. Stellen Sie sicher, dass das korrekt Finch läuft.

   ```
   $ finch --info
   $ finch ps
   ```

Weitere Informationen finden Sie unter Finch auf macOS installieren.

Linux

Sie können die Installation Finch unter Amazon Linux, Ubuntu oder mithilfe einer generischen Installationsmethode durchführen. Weitere Informationen finden Sie unter Finch unter Linux installieren.

Konfiguration Finch mit AWS SAM CLI

Nach der Installation Finch erkennt die AWS SAM CLI automatisch und verwendet Finch sie als Container-Runtime, wenn sie Docker nicht ausgeführt wird.

> **Note**
>
> Finchdient als Fallback fürDocker. Wenn Sie beide Finch installiert haben Docker und ausgeführt werden, priorisiert AWS SAM CLI Docker standardmäßig.

Konfiguration der Administratoreinstellungen (macOS)

Sie können AWS SAM CLI so konfigurieren, dass eine bestimmte Container-Laufzeit als Standard verwendet wird, indem Sie eine Administrator-Einstellungsdatei erstellen. Diese systemweite Einstellung setzt das standardmäßige Fallback-Verhalten außer Kraft.

Verfügbare Konfigurationsoptionen

- FinchAls bevorzugte Container-Laufzeit festlegen:

```
$ sudo /usr/libexec/PlistBuddy -c "Add :DefaultContainerRuntime string finch" /
Library/Preferences/com.amazon.samcli.plist
```

- DockerAls bevorzugte Container-Laufzeit festlegen:

```
$ sudo /usr/libexec/PlistBuddy -c "Add :DefaultContainerRuntime string docker" /
Library/Preferences/com.amazon.samcli.plist
```

- Einstellung entfernen (zum Standardverhalten zurückkehren):

```
$ sudo /usr/libexec/PlistBuddy -c "Remove :DefaultContainerRuntime" /Library/
Preferences/com.amazon.samcli.plist
```

> **Note**
>
> Wenn keine Administratoreinstellung festgelegt ist, verwendet AWS SAM CLI Docker standardmäßig, sofern verfügbar, und Finch als Fallback, wenn sie nicht ausgeführt Docker wird.

Unterschiede beim Einhängen von Verzeichnissen (macOS)

Unter macOS schlagen AWS SAM lokale Befehle fehl, wenn Sie Ihr Projekt außerhalb Ihres Home-Verzeichnisses (~) oder suchen/`Volumes`. Dieses Design schränkt das automatische Einbinden von Verzeichnissen aus Sicherheitsgründen ohne Ihr Wissen ein.

Optionen zur Schadensbegrenzung

- Option 1: Öffnen Sie die Finch-Konfiguration in einem Texteditor `~/.finch/finch.yaml` und fügen Sie Ihren Projektpfad unter dem `additional_directories` Abschnitt hinzu. Starten Sie dann die Finch virtuelle Maschine neu. Weitere Informationen finden Sie in der Finch-Dokumentation.

  ```
  $ finch vm stop
  $ finch vm start
  ```

- Option 2: Verschieben Sie Ihr Projekt in Ihr Home-Verzeichnis.

Die Installation wird überprüft

Stellen Sie nach der Installation und dem Start sicherFinch, dass es mit AWS SAM CLI funktioniert.

Grundlegende Überprüfung

Führen Sie den folgenden Befehl aus, um zu überprüfen, ob AWS SAM CLI Folgendes verwenden kannFinch:

```
$ sam build --use-container
```

Sie sollten sehen, wie AWS SAM CLI verwendet wirdFinch, um Ihre Anwendung in einer Containerumgebung zu erstellen.

> **Note**
>
> Finchdient als Ausweichlösung fürDocker. Wenn ausgeführt Docker wird, priorisiert AWS SAM CLI Docker standardmäßig.

Testen Finch als Fallback

Um Finch als Fallback zu testen, wenn Docker es läuft, müssen Sie zuerst aufhörenDocker.

Um die Finch Fallback-Funktionalität zu testen

1. DockerVerwenden Sie nicht mehr die entsprechenden Befehle für Ihr Betriebssystem.

   macOS:

   ```
   $ docker stop $(docker ps -q)
   $ pkill -f "Docker Desktop"
   ```

   Verify Docker wurde gestoppt.

   ```
   $ docker ps
   ```

   Linux:

   ```
   $ docker stop $(docker ps -q)
   $ sudo systemctl stop docker
   ```

   Verify Docker wurde gestoppt.

   ```
   $ sudo systemctl status docker
   ```

2. Führen Sie den `sam build --use-container` Befehl aus, um zu überprüfen, ob AWS SAM CLI ihn automatisch Finch als Fallback verwendet.

Stoppen von Finch

Wenn Sie den Vorgang beenden müssenFinch, verwenden Sie die folgenden Befehle:

macOS:

```
$ finch vm stop
$ sudo pkill -f finch
```

Verify Finch wurde gestoppt.

```
$ finch vm status
```

Linux:

```
$ sudo systemctl stop finch
$ sudo pkill -f finch
```

Verify Finch wurde gestoppt.

```
$ sudo systemctl status finch
```

Fehlerbehebung

Wenn Sie auf Probleme stoßen mitFinch:

- Stellen Sie sicher, dass Ihr Betriebssystem die Voraussetzungen erfüllt.
- Stellen Sie sicher, dass Sie die Virtualisierung auf Ihrem System aktiviert haben.
- Stellen Sie für macOS sicher, dass Sie über ausreichend Festplattenspeicher für die virtuelle Maschine verfügen.
- Stellen Sie für Linux sicher, dass Sie über die erforderlichen Berechtigungen zum Ausführen von `systemctl` Befehlen verfügen.
- Wenn Sie beim Übertragen von Images mit mehreren Architekturen an Amazon Elastic Container Registry (Amazon ECR) CloudFormation Bereitstellungsfehler mit der Fehlermeldung „Medientyp nicht unterstützt“ feststellen, erstellen Sie ein einzelnes Image-Artefakt, anstatt Images mit mehreren Architekturen zu verwenden. Dies liegt daran, Finch dass sowohl der Bildindex als auch das Bild an Amazon ECR übertragen werden, während nur das Docker Bild übertragen wird.
- Wenn plattformübergreifendes Erstellen fehlschlägt, weil die Architektur der Zielfunktion nicht mit der Architektur des Host-Computers unter Linux übereinstimmt, aktivieren Sie die plattformübergreifende Emulation, indem Sie Folgendes ausführen. `sudo finch run --privileged --rm tonistiigi/binfmt:master --install all`

Weitere Informationen zur Fehlerbehebung finden Sie in der Finch-Dokumentation oder im. AWS SAMCLIProblembehandlung

Nächste Schritte

Nachdem Sie es Finch installiert haben, können Sie es mit AWS SAM CLI verwenden, um:

- Erstellen Sie serverlose Anwendungen mithilfe von Containern mit`sam build --use-container`. Weitere Informationen finden Sie unter the section called “gleicher Build”.
- Testen Sie Lambda-Funktionen lokal mit`sam local invoke`. Weitere Informationen finden Sie unter the section called “sam local invoke”.
- Starten Sie ein lokales API Gateway mit`sam local start-api`. Weitere Informationen finden Sie unter the section called “Sam lokale Start-API”.
- Starten Sie einen lokalen Lambda-Endpunkt mit`sam local start-lambda`. Weitere Informationen finden Sie unter the section called “Sam Local Start-Lambda”.

Weitere Informationen zur Verwendung von AWS SAM CLI mit Containern finden Sie unterStandard-Build mit AWS SAM.

Bildrepositorien für AWS SAM

AWS SAM vereinfacht Continuous Integrations- und Continuous Delivery (CI/CD) -Aufgaben für serverlose Anwendungen mithilfe von Build-Container-Images. Die bereitgestellten Images enthalten die AWS SAM Befehlszeilenschnittstelle (CLI) und Build-Tools für eine Reihe unterstützter AWS Lambda Laufzeiten. AWS SAM Dadurch ist es einfacher, serverlose Anwendungen mit dem zu erstellen und zu verpacken. AWS SAMCLI Sie können diese Images mit CI/CD Systemen verwenden, um die Erstellung und Bereitstellung von AWS SAM Anwendungen zu automatisieren. Beispiele finden Sie unter Bereitstellen mit CI/CD Systemen und Pipelines.

AWS SAM URIs für Build-Container-Images sind mit der Version des in diesem Image AWS SAMCLI enthaltenen Tags versehen. Wenn Sie den URI ohne Tags angeben, wird die neueste Version verwendet. `public.ecr.aws/sam/build-nodejs20.x`Verwendet beispielsweise das neueste Bild. `public.ecr.aws/sam/build-nodejs20.x:1.24.1`Verwendet jedoch das Image, das die AWS SAM CLI-Version 1.24.1 enthält.

Ab Version 1.33.0 von sind `x86_64` sowohl als auch `arm64` Container-Images für unterstützte Laufzeiten verfügbar. AWS SAMCLI Weitere Informationen finden Sie unter Lambda-Laufzeiten im Entwicklerhandbuch. AWS Lambda

> **Note**
>
> Vor Version 1.22.0 von DockerHub war es das Standard-Repository AWS SAMCLI, aus dem das Container-Image AWS SAMCLI abgerufen wurde. Ab Version 1.22.0 wurde das Standard-Repository in Amazon Elastic Container Registry Public (Amazon ECR Public) geändert. Um ein Container-Image aus einem anderen Repository als dem aktuellen Standard-Repository abzurufen, können Sie den gleicher Build Befehl mit der --build-image Option verwenden. Die Beispiele am Ende dieses Themas zeigen, wie Anwendungen mithilfe von DockerHub Repository-Images erstellt werden.

URIs des Bild-Repositorys

In der folgenden Tabelle sind die URIs der Amazon ECR Public-Build-Container-Images aufgeführt, die Sie zum Erstellen und Verpacken serverloser Anwendungen verwenden können. AWS SAM

> **Note**
>
> Amazon ECR Public wurde DockerHub ab Version 1.22.0 ersetzt. AWS SAMCLI Wenn Sie eine frühere Version von verwenden, empfehlen wir Ihnen AWS SAMCLI, ein Upgrade durchzuführen.

Laufzeit	Amazon ECR Public
Benutzerdefinierte Laufzeit (AL2023)	public.ecr. aws/sam/build-provided.al2023
Benutzerdefinierte Laufzeit (AL2)	public.ecr. aws/sam/build-provided.al2
Benutzerdefinierte Laufzeit	public.ecr. aws/sam/build-bereitgestellt
Java 25	public.ecr. aws/sam/build-java25
Java 21	public.ecr. aws/sam/build-java21
Java 17	public.ecr. aws/sam/build-java17
Java 17 (AL2023)	public.ecr. aws/sam/build-java17.al2023

Laufzeit	Amazon ECR Public
Java 11	public.ecr. aws/sam/build-java11
Java 11 (AL2023)	public.ecr. aws/sam/build-java11.al2023
Java 8 (AL2023)	public.ecr. aws/sam/build-java8.al2023
Java 8	public.ecr. aws/sam/build-java8
.NET 9	public.ecr. aws/sam/build-dotnet9
.NET 8	public.ecr. aws/sam/build-dotnet8
.NET 7	public.ecr. aws/sam/build-dotnet7
.NET 6	public.ecr. aws/sam/build-dotnet6
Node.js 24	public.ecr. aws/sam/build-nodejs24.x
Node.js 22	public.ecr. aws/sam/build-nodejs22.x
Node.js 20	public.ecr. aws/sam/build-nodejs20.x
Node.js 18	public.ecr. aws/sam/build-nodejs18.x
Node.js 16	public.ecr. aws/sam/build-nodejs16.x
Python 3.14	public.ecr. aws/sam/build-python3.14
Python 3.13	public.ecr. aws/sam/build-python3.13
Python 3.12	public.ecr. aws/sam/build-python3.12
Python 3.11	public.ecr. aws/sam/build-python3.11
Python 3.10	public.ecr. aws/sam/build-python3.10
Python 3.9	public.ecr. aws/sam/build-python3.9
Python 3.8	public.ecr. aws/sam/build-python3.8

Laufzeit	Amazon ECR Public
Ruby 4.0	public.ecr. aws/sam/build-ruby4.0
Ruby 3.4	public.ecr. aws/sam/build-ruby3.4
Ruby 3.3	public.ecr. aws/sam/build-ruby3.3
Ruby 3.2	public.ecr. aws/sam/build-ruby3.2

Beispiele

Die folgenden beiden Beispielbefehle erstellen Anwendungen mithilfe von Container-Images aus dem Image-Repository:

Erstellen Sie eine Node.js 24 Anwendung mithilfe eines Container-Images, das aus Amazon ECR abgerufen wurde:

```
$ sam build --use-container --build-image public.ecr.aws/sam/build-nodejs24.x
```

Erstellen Sie eine Funktionsressource mithilfe des Python 3.14 Container-Images, das aus Amazon ECR abgerufen wurde:

```
$ sam build --use-container --build-image Function1=public.ecr.aws/sam/build-python3.14
```

Schrittweise Bereitstellung serverloser Anwendungen mit AWS SAM

AWS Serverless Application Model (AWS SAM) ist integriert CodeDeploy, um schrittweise AWS Lambda Bereitstellungen zu ermöglichen. AWS SAM Erledigt mit nur wenigen Konfigurationszeilen Folgendes für Sie:

- Stellt neue Versionen Ihrer Lambda-Funktion bereit und erstellt automatisch Aliase, die auf die neue Version verweisen.
- Leitet den Kundenverkehr schrittweise auf die neue Version um, bis Sie davon überzeugt sind, dass sie wie erwartet funktioniert. Wenn ein Update nicht ordnungsgemäß funktioniert, können Sie die Änderungen rückgängig machen.

- Definiert Testfunktionen vor und nach dem Datenverkehr, um zu überprüfen, ob der neu bereitgestellte Code korrekt konfiguriert ist und ob Ihre Anwendung erwartungsgemäß funktioniert.
- Macht die Bereitstellung automatisch rückgängig, wenn CloudWatch Alarme ausgelöst werden.

> **Note**
>
> Wenn Sie schrittweise Bereitstellungen über Ihre AWS SAM Vorlage aktivieren, wird automatisch eine CodeDeploy Ressource für Sie erstellt. Sie können die CodeDeploy Ressource direkt über die AWS-Managementkonsole anzeigen.

Beispiel

Das folgende Beispiel zeigt die Verwendung von CodeDeploy , um Kunden schrittweise auf Ihre neu bereitgestellte Version der Lambda-Funktion umzustellen:

```
Resources:
MyLambdaFunction:
  Type: AWS::Serverless::Function
  Properties:
    Handler: index.handler
    Runtime: nodejs20.x
    CodeUri: s3://bucket/code.zip

    AutoPublishAlias: live

    DeploymentPreference:
      Type: Canary10Percent10Minutes
      Alarms:
        # A list of alarms that you want to monitor
        - !Ref AliasErrorMetricGreaterThanZeroAlarm
        - !Ref LatestVersionErrorMetricGreaterThanZeroAlarm
      Hooks:
        # Validation Lambda functions that are run before & after traffic shifting
        PreTraffic: !Ref PreTrafficLambdaFunction
        PostTraffic: !Ref PostTrafficLambdaFunction
```

Diese Überarbeitungen der AWS SAM Vorlage haben folgende Auswirkungen:

- `AutoPublishAlias`: Durch Hinzufügen dieser Eigenschaft und Angabe eines Aliasnamens: AWS SAM
  - Erkennt, wenn neuer Code bereitgestellt wird, basierend auf Änderungen an der Amazon S3 S3-URI der Lambda-Funktion.
  - Erstellt und veröffentlicht eine aktualisierte Version dieser Funktion mit dem neuesten Code.
  - Erstellt einen Alias mit einem von Ihnen angegebenen Namen (sofern nicht bereits ein Alias vorhanden ist) und verweist auf die aktualisierte Version der Lambda-Funktion. Funktionsaufrufe sollten den Alias-Qualifizierer verwenden, um dies zu nutzen. Wenn Sie mit der Versionierung und Aliasnamen von Lambda-Funktionen nicht vertraut sind, finden Sie weitere Informationen unter AWS Lambda Funktionsversionierung und Aliase.
- `Deployment Preference Type`: Im vorherigen Beispiel werden 10 Prozent Ihres Kundenverkehrs sofort auf Ihre neue Version umgestellt. Nach 10 Minuten wird der gesamte Verkehr auf die neue Version umgestellt. Wenn Ihre Tests vor oder nach dem Datenverkehr jedoch fehlschlagen oder wenn ein CloudWatch Alarm ausgelöst wird, wird Ihre Bereitstellung CodeDeploy zurückgesetzt. Sie können auf folgende Weise angeben, wie der Datenverkehr zwischen den Versionen verlagert werden soll:
  - `Canary`: Der Datenverkehr wird in zwei Inkrementen verschoben. Sie können aus vordefinierten Canary-Optionen wählen. Die Optionen geben den Prozentsatz des Datenverkehrs an, der im ersten Schritt auf Ihre aktualisierte Lambda-Funktionsversion umgestellt wurde, und das Intervall in Minuten, bevor der verbleibende Verkehr in der zweiten Stufe verschoben wird.
  - `Linear`: Der Datenverkehr wird in gleichmäßigen Inkrementen um die gleiche Anzahl von Minuten zwischen den einzelnen Inkrementen verschoben. Sie können aus vordefinierten linearen Optionen wählen, die den Prozentsatz des Datenverkehrs angeben, der in jedem Inkrement verschoben wird, und die Anzahl der Minuten zwischen den einzelnen Schritten.
  - `AllAtOnce`: Der gesamte Datenverkehr wird gleichzeitig von der ursprünglichen Lambda-Funktion auf die aktualisierte Lambda-Funktionsversion umgestellt.

  In der folgenden Tabelle sind weitere Optionen zur Verkehrsverlagerung aufgeführt, die über die im Beispiel verwendete hinaus verfügbar sind.

Bereitstellungspräferenztyp
Canary10Percent30Minutes
Canary10Percent5Minutes

Bereitstellungspräferenztyp
Canary10Percent10Minutes
Canary10Percent15Minutes
Linear 10 10 Minuten PercentEvery
Linear 10 1 Minute PercentEvery
Linear 10 2 Minuten PercentEvery
Linear 10 3 Minuten PercentEvery
AllAtOnce

- `Alarms`: Dabei handelt es sich um CloudWatch Alarme, die durch alle Fehler ausgelöst werden, die bei der Bereitstellung ausgelöst wurden. Wenn sie auftreten, wird Ihre Bereitstellung automatisch rückgängig gemacht. Dies ist beispielsweise der Fall, wenn der aktualisierte Code, den Sie bereitstellen, Fehler in der Anwendung verursacht. Ein anderes Beispiel ist, wenn einige AWS Lambdaoder benutzerdefinierte CloudWatch Messwerte, die Sie angegeben haben, den Alarmschwellenwert überschritten haben.
- `Hooks`: Dabei handelt es sich um Testfunktionen vor und nach dem Verkehr, die Prüfungen durchführen, bevor die Verkehrsverlagerung auf die neue Version beginnt und nachdem die Verkehrsverlagerung abgeschlossen ist.
  - `PreTraffic`: Ruft vor Beginn der Verkehrsverlagerung die CodeDeploy Lambda-Funktion Pre-Traffic Hook auf. Diese Lambda-Funktion muss zurückrufen CodeDeploy und deren Erfolg oder Misserfolg anzeigen. Wenn die Funktion fehlschlägt, wird sie abgebrochen und ein Fehler wird an gemeldet. CloudFormation Wenn die Funktion erfolgreich ist, wird mit der CodeDeploy Verkehrsverlagerung fortgefahren.
  - `PostTraffic`: Ruft nach Abschluss der Verkehrsverlagerung die CodeDeploy Lambda-Funktion nach dem Traffic Hook auf. Dies ähnelt dem Pre-Traffic-Hook, bei dem die Funktion erneut aufrufen muss, um einen Erfolg oder CodeDeploy Misserfolg zu melden. Verwenden Sie Post-Traffic-Hooks für Integrationstests oder andere Validierungsaktivitäten.

  Weitere Informationen finden Sie unter SAM-Referenz für sichere Bereitstellungen.

Schrittweise erstmalige Bereitstellung einer Lambda-Funktion

Bei der schrittweisen Bereitstellung einer Lambda-Funktion CodeDeploy ist eine zuvor bereitgestellte Funktionsversion erforderlich, von der der Datenverkehr verlagert werden soll. Daher sollte Ihre erste Bereitstellung in zwei Schritten durchgeführt werden:

- Schritt 1: Stellen Sie Ihre Lambda-Funktion bereit und erstellen Sie automatisch Aliase mit. `AutoPublishAlias`
- Schritt 2: Führen Sie Ihre schrittweise Bereitstellung mit durch. `DeploymentPreference`

Wenn Sie Ihre erste schrittweise Bereitstellung in zwei Schritten durchführen, erhalten Sie CodeDeploy eine frühere Lambda-Funktionsversion, von der aus Sie den Datenverkehr verlagern können.

Schritt 1: Stellen Sie Ihre Lambda-Funktion bereit

```
Resources:
MyLambdaFunction:
  Type: AWS::Serverless::Function
  Properties:
    Handler: index.handler
    Runtime: nodejs20.x
    CodeUri: s3://bucket/code.zip

    AutoPublishAlias: live
```

Schritt 2: Führen Sie Ihre schrittweise Bereitstellung durch

```
Resources:
MyLambdaFunction:
  Type: AWS::Serverless::Function
  Properties:
    Handler: index.handler
    Runtime: nodejs20.x
    CodeUri: s3://bucket/code.zip

    AutoPublishAlias: live

    DeploymentPreference:
      Type: Canary10Percent10Minutes
```

```
      Alarms:
        # A list of alarms that you want to monitor
        - !Ref AliasErrorMetricGreaterThanZeroAlarm
        - !Ref LatestVersionErrorMetricGreaterThanZeroAlarm
      Hooks:
        # Validation Lambda functions that are run before and after traffic shifting
        PreTraffic: !Ref PreTrafficLambdaFunction
        PostTraffic: !Ref PostTrafficLambdaFunction
```

Weitere Informationen

Ein praktisches Beispiel für die Konfiguration einer schrittweisen Bereitstellung finden Sie in Modul 5 — Bereitstellungen auf Canary in The Complete AWS SAM Workshop.

Wichtige Referenzhinweise für AWS SAM

Dieser Abschnitt enthält wichtige Hinweise und Ankündigungen für AWS Serverless Application Model (AWS SAM).

Themen

- Wichtige Hinweise für 2023

Wichtige Hinweise für 2023

Oktober 2023

AWS SAMCLIEinstellung der Unterstützung für Python 3.7

Veröffentlicht am 20.10.2023

Python 3.7erhielt end-of-life den Status im Juni 2023. AWS SAM CLISie werden den Support für den Python 3.7 24. Oktober 2023 einstellen. Weitere Informationen finden Sie in der Ankündigung im aws-sam-cli GitHubRepositorium.

Diese Änderung wirkt sich auf die folgenden Benutzer aus:

- Wenn Sie das AWS SAM CLI über verwenden Python 3.7 und installieren`pip`.
- Wenn Sie das `aws-sam-cli` als Bibliothek verwenden und Ihre Anwendung damit erstellenPython 3.7.

Wenn Sie das AWS SAM CLI mit einer anderen Methode installieren und verwalten, sind Sie davon nicht betroffen.

Für betroffene Benutzer empfehlen wir, dass Sie Ihre Entwicklungsumgebung auf Python 3.8 oder eine neuere Version aktualisieren.

Diese Änderung hat keinen Einfluss auf die Unterstützung der Python 3.7 AWS Lambda Laufzeitumgebung. Weitere Informationen finden Sie unter Runtime Deprecation Policy im AWS Lambda Developer Guide.

AWS SAM CLI Terraform Support

In diesem Abschnitt wird die Verwendung der AWS Serverless Application Model Befehlszeilenschnittstelle (AWS SAMCLI) mit Ihren Terraform Projekten und der Terraform Cloud behandelt.

Erstellen Sie ein GitHub Problem, um Feedback zu geben und Funktionsanfragen einzureichen.

Topics

- Erste Schritte mit Terraform Unterstützung für AWS SAM CLI
- Verwendung der AWS SAM CLI mit Terraform zum lokalen Debuggen und Testen
- Verwendung der AWS SAM CLI mit Serverless.tf zum lokalen Debuggen und Testen
- AWS SAM CLI mit Terraform Referenz
- Was ist AWS SAM CLI Unterstützung für Terraform?

Erste Schritte mit Terraform Unterstützung für AWS SAM CLI

Dieses Thema behandelt die ersten Schritte mit der Verwendung der AWS Serverless Application Model Befehlszeilenschnittstelle (AWS SAMCLI) mitTerraform.

Erstellen Sie ein GitHub Problem, um Feedback zu geben und Funktionsanfragen einzureichen.

Topics

- AWS SAM CLI Terraform Voraussetzungen
- Verwenden AWS SAM CLI Befehle mit Terraform
- Richten Sie ein für Terraform projects
- Richten Sie ein für Terraform Cloud

AWS SAM CLI Terraform Voraussetzungen

Erfüllen Sie alle Voraussetzungen, um das in AWS SAMCLI Ihren Terraform Projekten verwenden zu können.

1. Installieren oder aktualisieren Sie das AWS SAM CLI

   Um zu überprüfen, ob Sie das AWS SAMCLI installiert haben, führen Sie Folgendes aus:

```
$ sam --version
```

Wenn das bereits installiert AWS SAMCLI ist, wird in der Ausgabe eine Version angezeigt. Informationen zum Upgrade auf die neueste Version finden Sie unterAktualisierung des AWS SAMCLI.

Anweisungen zur Installation von AWS SAMCLI Along mit allen zugehörigen Voraussetzungen finden Sie unterInstallieren Sie das AWS SAMCLI.

2. Installieren Terraform

   Gehen Sie wie folgt vor, um zu überprüfen, ob Sie es Terraform installiert haben:

   ```
   $ terraform -version
   ```

   Informationen zur Installation Terraform finden Sie unter Installieren Terraform in der Terraform Registrierung.

3. Installieren Docker für lokale Tests

   Die AWS SAMCLI Anforderungen Docker für lokale Tests. Informationen zur Installation Docker finden Sie unterInstallation von Docker zur Verwendung mit dem AWS SAMCLI.

Verwenden AWS SAM CLI Befehle mit Terraform

Wenn Sie einen unterstützten AWS SAMCLI Befehl ausführen, verwenden Sie die `--hook-name` Option und geben Sie den `terraform` Wert an. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam local invoke --hook-name terraform
```

Sie können diese Option in Ihrer AWS SAMCLI Konfigurationsdatei wie folgt konfigurieren:

```
hook_name = "terraform"
```

Richten Sie ein für Terraform projects

Führen Sie die Schritte in diesem Thema aus, um das AWS SAMCLI mit Terraform Projekten zu verwenden.

Wenn Sie Ihre AWS Lambda Artefakte außerhalb Ihres Terraform Projekts erstellen, ist kein zusätzliches Setup erforderlich. Informationen Verwendung der AWS SAM CLI mit Terraform zum lokalen Debuggen und Testen zum Einstieg in die Verwendung von finden Sie unter AWS SAMCLI.

Wenn Sie Ihre Lambda-Artefakte in Ihren Terraform Projekten erstellen, müssen Sie Folgendes tun:

1. Installieren Sie Python 3.8 oder neuer
2. Installieren Sie das Make Tool.
3. Definieren Sie die Build-Logik Ihrer Lambda-Artefakte in Ihrem Terraform Projekt.
4. Definieren Sie eine `sam metadata` Ressource, die AWS SAMCLI Sie über Ihre Build-Logik informiert.
5. Verwenden Sie den AWS SAMCLI `sam build` Befehl, um Ihre Lambda-Artefakte zu erstellen.

Installieren Python 3.8 oder neuer

Python3.8 oder neuer ist für die Verwendung mit dem erforderlich AWS SAMCLI. Wenn Sie ausführen`sam build`, werden die Creates AWS SAMCLI ausgeführt`makefiles`, die Python Befehle zum Erstellen Ihrer Lambda-Artefakte enthalten.

Eine Installationsanleitung finden Sie unter Python herunterladen im Python-Anfängerhandbuch.

Stellen Sie sicher, dass Python 3.8 oder neuer zu Ihrem Computerpfad hinzugefügt wurde, indem Sie Folgendes ausführen:

```
$ python --version
```

Die Ausgabe sollte eine Version von Python anzeigen, die 3.8 oder neuer ist.

Installieren Sie das Marke tool

GNUMakeist ein Tool, das die Generierung von ausführbaren Dateien und anderen Nicht-Quelldateien für Ihr Projekt steuert. Die Kreationen AWS SAMCLI, `makefiles` die auf dieses Tool angewiesen sind, um Ihre Lambda-Artefakte zu erstellen.

Wenn Sie es nicht auf Ihrem lokalen Computer Make installiert haben, installieren Sie es, bevor Sie fortfahren.

Für Windows können Sie die Installation mit Chocolatey durchführen. Eine Anleitung finden Sie unter Verwenden von Chocolatey im Abschnitt So installieren und verwenden Sie „Make“ in Windows

Definieren Sie die Build-Logik für Lambda-Artefakte

Verwenden Sie den `null_resource` Terraform Ressourcentyp, um Ihre Lambda-Build-Logik zu definieren. Im Folgenden finden Sie ein Beispiel, das ein benutzerdefiniertes Build-Skript verwendet, um eine Lambda-Funktion zu erstellen.

```
resource "null_resource" "build_lambda_function" {
    triggers = {
        build_number = "${timestamp()}"
    }

    provisioner "local-exec" {
        command =  substr(pathexpand("~"), 0, 1) == "/"? "./
py_build.sh \"${local.lambda_src_path}\" \"${local.building_path}\"
 \"${local.lambda_code_filename}\" Function" : "powershell.exe -File .\\PyBuild.ps1
 ${local.lambda_src_path} ${local.building_path} ${local.lambda_code_filename}
 Function"
    }
}
```

Definieren Sie eine dieselben Metadaten Ressource

Die `sam metadata` Ressource ist ein `null_resource` Terraform Ressourcentyp, der die AWS SAMCLI Informationen bereitstellt, die sie zum Auffinden Ihrer Lambda-Artefakte benötigt. Für jede Lambda-Funktion oder -Layer in Ihrem Projekt ist eine eindeutige `sam metadata` Ressource erforderlich. Weitere Informationen zu diesem Ressourcentyp finden Sie unter null_resource in der Registrierung. Terraform

Um eine zu definieren dieselben Metadaten Ressource

1. Benennen Sie Ihre Ressource, beginnend mit`sam_metadata_`, um die Ressource als sam metadata Ressource zu identifizieren.
2. Definieren Sie Ihre Lambda-Artefakteigenschaften innerhalb des `triggers` Blocks Ihrer Ressource.
3. Geben Sie mit dem Argument an`null_resource`, dass Ihre Lambda-Build-Logik enthalten ist. `depends_on`

   Das Folgende ist eine Beispielvorlage:

   ```
   resource "null_resource" "sam_metadata_..." {
   ```

```
  triggers = {
    resource_name = resource_name
    resource_type = resource_type
    original_source_code = original_source_code
    built_output_path = built_output_path
  }
  depends_on = [
    null_resource.build_lambda_function # ref to your build logic
  ]
}
```

Das Folgende ist eine `sam metadata` Beispielressource:

```
resource "null_resource" "sam_metadata_aws_lambda_function_publish_book_review" {
    triggers = {
        resource_name = "aws_lambda_function.publish_book_review"
        resource_type = "ZIP_LAMBDA_FUNCTION"
        original_source_code = "${local.lambda_src_path}"
        built_output_path = "${local.building_path}/${local.lambda_code_filename}"
    }
    depends_on = [
        null_resource.build_lambda_function
    ]
}
```

Der Inhalt Ihrer `sam metadata` Ressource variiert je nach Lambda-Ressourcentyp (Funktion oder Layer) und Verpackungstyp (ZIP oder Bild). Weitere Informationen sowie Beispiele finden Sie untersam-Metadaten-Ressource.

Wenn Sie eine `sam metadata` Ressource konfigurieren und einen unterstützten AWS SAMCLI Befehl verwenden, generiert der die Metadatendatei, bevor der AWS SAMCLI Befehl ausgeführt AWS SAMCLI wird. Sobald Sie diese Datei generiert haben, können Sie die `--skip-prepare-infra` Option mit zukünftigen AWS SAMCLI Befehlen verwenden, um den Prozess der Metadatengenerierung zu überspringen und Zeit zu sparen. Diese Option sollte nur verwendet werden, wenn Sie keine Infrastrukturänderungen vorgenommen haben, z. B. neue Lambda-Funktionen oder neue API-Endpunkte erstellt haben.

Verwenden Sie die AWS SAM CLI um Ihre Lambda-Artefakte zu bauen

Verwenden Sie den AWS SAMCLI `sam build` Befehl, um Ihre Lambda-Artefakte zu erstellen. Wenn Sie das ausführen`sam build`, AWS SAMCLI führt das Folgendes aus:

1. Sucht in Ihrem Terraform Projekt nach `sam metadata` Ressourcen, um mehr über Ihre Lambda-Ressourcen zu erfahren und sie zu finden.
2. Initiiert Ihre Lambda-Build-Logik, um Ihre Lambda-Artefakte zu erstellen.
3. Erstellt ein `.aws-sam` Verzeichnis, das Ihr Terraform Projekt zur Verwendung mit den Befehlen organisiert. AWS SAMCLI `sam local`

Um mit Sam Build zu bauen

1. Führen Sie in dem Verzeichnis, das Ihr Terraform Root-Modul enthält, Folgendes aus:

   ```
   $ sam build --hook-name terraform
   ```

2. Um eine bestimmte Lambda-Funktion oder -Layer zu erstellen, führen Sie Folgendes aus

   ```
   $ sam build --hook-name terraform lambda-resource-id
   ```

   Die Lambda-Ressourcen-ID kann der Name der Lambda-Funktion oder die vollständige Terraform Ressourcenadresse sein, z. B. oder. `aws_lambda_function.list_books` `module.list_book_function.aws_lambda_function.this[0]`

Wenn sich Ihr Funktionsquellcode oder andere Terraform Konfigurationsdateien außerhalb des Verzeichnisses befinden, das Ihr Terraform Root-Modul enthält, müssen Sie den Speicherort angeben. Verwenden Sie die `--terraform-project-root-path` Option, um den absoluten oder relativen Pfad zum Verzeichnis der obersten Ebene anzugeben, das diese Dateien enthält. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam build --hook-name terraform --terraform-project-root-path ~/projects/terraform/
demo
```

Erstellen Sie mithilfe eines Containers

Wenn Sie den AWS SAMCLI `sam build` Befehl ausführen, können Sie den so konfigurieren, AWS SAMCLI dass Ihre Anwendung mithilfe eines lokalen Docker Containers erstellt wird.

> **Note**
>
> Sie müssen es Docker installiert und konfiguriert haben. Detaillierte Anweisungen finden Sie unter Installation von Docker zur Verwendung mit dem AWS SAMCLI.

Um mit einem Container zu bauen

1. Erstellen Sie einen`Dockerfile`, der die Make Werkzeuge TerraformPython, und enthält. Sie sollten auch die Laufzeit Ihrer Lambda-Funktion mit einbeziehen.

   Das Folgende ist ein Beispiel: `Dockerfile`

```
FROM public.ecr.aws/amazonlinux/amazonlinux:2

RUN yum -y update \
    && yum install -y unzip tar gzip bzip2-devel ed gcc gcc-c++ gcc-gfortran \
    less libcurl-devel openssl openssl-devel readline-devel xz-devel \
    zlib-devel glibc-static libcxx libcxx-devel llvm-toolset-7 zlib-static \
    && rm -rf /var/cache/yum

RUN yum -y install make \
    && yum -y install zip

RUN yum install -y yum-utils \
    && yum-config-manager --add-repo https://rpm.releases.hashicorp.com/
AmazonLinux/hashicorp.repo \
    && yum -y install terraform \
    && terraform --version

# AWS Lambda Builders
RUN amazon-linux-extras enable python3.8
RUN yum clean metadata && yum -y install python3.8
RUN curl -L get-pip.io | python3.8
RUN pip3 install aws-lambda-builders
RUN ln -s /usr/bin/python3.8 /usr/bin/python3
RUN python3 --version

VOLUME /project
WORKDIR /project

ENTRYPOINT ["sh"]
```

2. Verwenden Sie docker build es, um Ihr Docker Image zu erstellen.

   Im Folgenden wird ein Beispiel gezeigt:

   ```
   $ docker build --tag terraform-build:v1 <path-to-directory-containing-Dockerfile>
   ```

3. Führen Sie den AWS SAMCLI `sam build` Befehl mit den `--build-image` Optionen `--use-container` und aus.

   Im Folgenden wird ein Beispiel gezeigt:

   ```
   $ sam build --use-container --build-image terraform-build:v1
   ```

Nächste Schritte

Informationen zum Einstieg in AWS SAMCLI Ihre Terraform Projekte finden Sie unterVerwendung der AWS SAM CLI mit Terraform zum lokalen Debuggen und Testen.

Richten Sie ein für Terraform Cloud

Wir empfehlen, dass Sie eine neuere Version verwendenTerraform v1.6.0. Wenn Sie eine ältere Version verwenden, müssen Sie lokal eine Terraform Plandatei generieren. Die lokale Plandatei enthält die AWS SAM CLI Informationen, die sie für lokale Tests und Debugging benötigt.

Um eine lokale Plandatei zu generieren

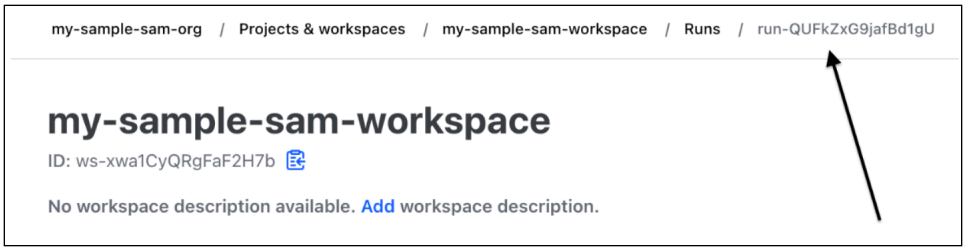
> **Note**
>
> Diese Schritte sind für Terraform v1.6.0 oder neuer nicht erforderlich. Informationen zum Einstieg in die Verwendung von AWS SAM CLI mit Terraform Cloud finden Sie unterVerwenden mit AWS SAMCLI Terraform.

1. Konfigurieren Sie ein API-Token — Die Art des Tokens hängt von Ihrer Zugriffsebene ab. Weitere Informationen finden Sie in der Terraform Cloud Dokumentation unter API-Token.
2. Legen Sie Ihre API-Token-Umgebungsvariable fest — Das Folgende ist ein Beispiel aus der Befehlszeile:

```
$ export TOKEN="<api-token-value>"
```

3. Rufen Sie Ihre Lauf-ID ab — Suchen Sie in der Terraform Cloud Konsole die Run-ID für den Terraform Lauf, den Sie mit dem verwenden möchten AWS SAMCLI.

   Die Run-ID befindet sich im Breadcrumb-Pfad Ihres Laufs.



4. Rufen Sie die Plandatei ab — Rufen Sie mit Ihrem API-Token Ihre lokale Plandatei ab. Das Folgende ist ein Beispiel aus der Befehlszeile:

```
curl \
   --header "Authorization: Bearer $TOKEN" \
   --header "Content-Type: application/vnd.api+json" \
   --location \
   https://app.terraform.io/api/v2/runs/<run ID>/plan/json-output \
   > custom_plan.json
```

Sie sind jetzt bereit, das AWS SAMCLI mit zu verwendenTerraform Cloud. Wenn Sie einen unterstützten AWS SAMCLI Befehl verwenden, verwenden Sie die `--terraform-plan-file` Option, um den Namen und Pfad Ihrer lokalen Plandatei anzugeben. Im Folgenden wird ein Beispiel gezeigt:

```
$ sam local invoke --hook-name terraform --terraform-plan-file custom-plan.json
```

Das Folgende ist ein Beispiel für die Verwendung des `sam local start-api` Befehls:

```
$ sam local start-api --hook-name terraform --terraform-plan-file custom-plan.json
```

Eine Beispielanwendung, die Sie mit diesen Beispielen verwenden können, finden Sie unter api_gateway_v2_tf_cloud im aws-samples-Repository. GitHub

Nächste Schritte

Informationen zum Einstieg in die Verwendung von AWS SAMCLI with Terraform Cloud finden Sie unterVerwendung der AWS SAM CLI mit Terraform zum lokalen Debuggen und Testen.

Verwendung der AWS SAM CLI mit Terraform zum lokalen Debuggen und Testen

In diesem Thema wird beschrieben, wie Sie unterstützte AWS Serverless Application Model Command Line Interface (AWS SAMCLI) -Befehle mit Ihren Terraform Projekten verwenden undTerraform Cloud.

Erstellen Sie ein GitHub Problem, um Feedback zu geben und Funktionsanfragen einzureichen.

Topics

- Lokales Testen mit gleicher lokaler Aufruf
- Lokales Testen mit dieselbe lokale Start-API
- Lokales Testen mit sam lokales Start-Lambda
- Terraform Einschränkungen

Lokales Testen mit gleicher lokaler Aufruf

> **Note**
>
> Um den zum lokalen Testen AWS SAMCLI zu verwenden, müssen Sie Docker installiert und konfiguriert haben. Detaillierte Anweisungen finden Sie unter Installation von Docker zur Verwendung mit dem AWS SAMCLI.

Im Folgenden finden Sie ein Beispiel dafür, wie Sie Ihre Lambda-Funktion lokal testen, indem Sie ein Ereignis übergeben:

```
$ sam local invoke --hook-name terraform hello_world_function -e events/event.json -
```

Weitere Informationen zur Verwendung dieses Befehls finden Sie unterEinführung in das Testen mit sam local invoke.

Lokales Testen mit dieselbe lokale Start-API

Um es `sam local start-api` mit zu verwendenTerraform, führen Sie Folgendes aus:

```
$ sam local start-api --hook-name terraform
```

Im Folgenden wird ein Beispiel gezeigt:

```
$ sam local start-api --hook-name terraform




Running Prepare Hook to prepare the current application


Executing prepare hook of hook "terraform"


Initializing Terraform application


...
Creating terraform plan and getting JSON output


....
Generating metadata file




Unresolvable attributes discovered in project, run terraform apply to resolve them.




Finished generating metadata file. Storing in...
Prepare hook completed and metadata file generated at: ...
```

```
Mounting HelloWorldFunction at http://127.0.0.1:3000/hello [GET]


Mounting None at http://127.0.0.1:3000/hello [POST]


You can now browse to the above endpoints to invoke your functions. You do not need
 to restart/reload SAM CLI while working on your functions, changes will be reflected
 instantly/automatically. If you
used sam build before running local commands, you will need to re-run sam build for the
 changes to be picked up. You only need to restart SAM CLI if you update your AWS SAM
 template
2023-06-26 13:21:20  * Running on http://127.0.0.1:3000/ (Press CTRL+C to quit)
```

Weitere Informationen zu diesem Befehl finden Sie unterEinführung in das Testen mit dieselbe lokale Start-API.

Lambda-Funktionen, die Lambda-Autorisierer verwenden

Für Lambda-Funktionen, die für die Verwendung von Lambda-Autorisierern konfiguriert sind, AWS SAMCLI wird automatisch Ihr Lambda-Autorisierer aufgerufen, bevor Sie Ihren Lambda-Funktionsendpunkt aufrufen.

- Weitere Informationen zu dieser Funktion finden Sie unter. AWS SAMCLI Lambda-Funktionen, die Lambda-Autorisierer verwenden
- Weitere Informationen zur Verwendung von Lambda-Autorisierern finden Sie unter Terraform Resource: aws_api_gateway_authorizer In der Registrierung. Terraform

Lokales Testen mit sam lokales Start-Lambda

Das Folgende ist ein Beispiel für das lokale Testen Ihrer Lambda-Funktion mit dem (): AWS Command Line Interface AWS CLI

1. Verwenden Sie das AWS SAMCLI, um eine lokale Testumgebung zu erstellen:

   ```
   $ sam local start-lambda --hook-name terraform hello_world_function
   ```

2. Verwenden Sie den AWS CLI , um Ihre Funktion lokal aufzurufen:

```
$ aws lambda invoke --function-name hello_world_function --endpoint-
url http://127.0.0.1:3001/ response.json --cli-binary-format raw-in-base64-out --
payload file://events/event.json
```

Weitere Informationen zu diesem Befehl finden Sie unterEinführung in das Testen mit sam local start-lambda.

Terraform Einschränkungen

Die folgenden Einschränkungen gelten bei der Verwendung von AWS SAMCLI withTerraform:

- Lambda-Funktionen, die mit mehreren Ebenen verknüpft sind.
- Terraformlokale Variablen, die Verbindungen zwischen Ressourcen definieren.
- Referenzierung einer Lambda-Funktion, die noch nicht erstellt wurde. Dazu gehören Funktionen, die im Body-Attribut der REST-API-Ressource definiert sind.

Um diese Einschränkungen zu umgehen, können Sie ausführen, `terraform apply` wenn eine neue Ressource hinzugefügt wird.

Verwendung der AWS SAM CLI mit Serverless.tf zum lokalen Debuggen und Testen

Das AWS Serverless Application Model Command Line Interface (AWS SAMCLI) kann mit Serverless.tf Modulen zum lokalen Debuggen und Testen Ihrer Funktionen und Ebenen verwendet werden. AWS Lambda Die folgenden AWS SAMCLI Befehle werden unterstützt:

- `sam build`
- `sam local invoke`
- `sam local start-api`
- `sam local start-lambda`

> **Note**
>
> Serverless.tf Version 4.6.0 und neuer unterstützen die AWS SAMCLI Integration.

Um AWS SAMCLI mit der Verwendung von Serverless.tf Modulen zu beginnen, aktualisieren Sie auf die neueste Version von Serverless.tf und. AWS SAMCLI

Ab Version 6.0.0 von serverless.tf müssen Sie den Parameter auf setzen. `create_sam_metadata` `true` Dadurch werden die Metadatenressourcen generiert, die der Befehl benötigt. AWS SAMCLI `sam build`

Weitere Informationen Serverless.tf finden Sie im https://registry.terraform.io/modules/terraform-aws-modules/lambda/aws/latest Terraform-aws-Lambda-Modul.

AWS SAM CLI mit Terraform Referenz

Dieser Abschnitt ist die Referenz für die Verwendung der AWS Serverless Application Model Befehlszeilenschnittstelle (AWS SAMCLI) Terraform für lokales Debuggen und Testen.

Erstellen Sie ein GitHub Problem, um Feedback zu geben und Funktionsanfragen einzureichen.

AWS SAM Referenz zu den unterstützten Funktionen

Die Referenzdokumentation für AWS SAMCLI Funktionen, deren Verwendung mit Terraform unterstützt wird, finden Sie hier:

- gleicher Build
- sam local invoke
- Sam lokale Start-API
- Sam Local Start-Lambda

Terraform spezifische Referenz

Referenzdokumentation speziell für die Verwendung von AWS SAMCLI Terraform with finden Sie hier:

- sam-Metadaten-Ressource

sam-Metadaten-Ressource

Diese Seite enthält Referenzinformationen für den sam metadata resource Ressourcentyp, der für Terraform Projekte verwendet wird.

- Eine Einführung in die Verwendung der AWS Serverless Application Model Befehlszeilenschnittstelle (AWS SAMCLI) mit Terraform finden Sie unterWas ist AWS SAM CLI Unterstützung für Terraform?.
- Informationen zur Verwendung von AWS SAMCLI with Terraform finden Sie unterVerwendung der AWS SAM CLI mit Terraform zum lokalen Debuggen und Testen.

Topics

- Argumente
- Beispiele

Argumente

Argument	Description
`built_output_path`	Der Pfad zu den eingebauten Artefakten Ihrer AWS Lambda Funktion.
`docker_build_args`	Dekodierte Zeichenfolge des JSON-Objekts mit Docker-Build-Argumenten. Dieses Argument ist optional.
`docker_context`	Der Pfad zu dem Verzeichnis, das den Docker-Image-Build-Kontext enthält.
`docker_file`	Der Pfad zur Docker-Datei. Dieser Pfad ist relativ zum `docker_context` Pfad. Dieses Argument ist optional. Der Standardwert ist `Dockerfile` .
`docker_tag`	Der Wert des erstellten Docker-Image-Tags. Dieser Wert ist optional.
`depends_on`	Der Pfad zur Erstellungsressource für Ihre Lambda-Funktion oder -Layer. Weitere Informationen finden Sie unter Das depends_on Argument in der Terraform Registrierung.
`original_source_code`	Der Pfad, in dem Ihre Lambda-Funktion definiert ist. Dieser Wert kann eine Zeichenfolge, ein Array von Zeichenketten oder ein dekodiertes JSON-Objekt als Zeichenfolge sein.

Argument	Description
	• Bei String-Arrays wird nur der erste Wert verwendet, da mehrere Codepfade nicht unterstützt werden. • Für JSON-Objekte `source_code_property` muss auch der definiert werden.
`resource_name`	Der Name der Lambda-Funktion.
`resource_type`	Das Format Ihres Lambda-Funktionspakettyps. Zulässige Werte sind: • `IMAGE_LAMBDA_FUNCTION` • `LAMBDA_LAYER` • `ZIP_LAMBDA_FUNCTION`
`source_code_property`	Der Pfad zum Lambda-Ressourcencode im JSON-Objekt. Definieren Sie diese Eigenschaft, wenn `original_source_code` es sich um ein JSON-Objekt handelt.

Beispiele

sam-Metadatenressource, die mithilfe des ZIP-Pakettyps auf eine Lambda-Funktion verweist

```
# Lambda function resource
resource "aws_lambda_function" "tf_lambda_func" {
  filename = "${path.module}/python/hello-world.zip"
  handler = "index.lambda_handler"
  runtime = "python3.8"
  function_name = "function_example"
  role = aws_iam_role.iam_for_lambda.arn
  depends_on = [
    null_resource.build_lambda_function # function build logic
  ]
}

# sam metadata resource
resource "null_resource" "sam_metadata_function_example" {
  triggers = {
    resource_name = "aws_lambda_function.function_example"
    resource_type = "ZIP_LAMBDA_FUNCTION"
```

```
    original_source_code = "${path.module}/python"
    built_output_path = "${path.module}/building/function_example"
  }
  depends_on = [
    null_resource.build_lambda_function # function build logic
  ]
}
```

sam-Metadatenressource, die mithilfe des Image-Pakettyps auf eine Lambda-Funktion verweist

```
resource "null_resource" "sam_metadata_function {
  triggers = {
    resource_name = "aws_lambda_function.image_function"
    resource_type = "IMAGE_LAMBDA_FUNCTION"
    docker_context = local.lambda_src_path
    docker_file = "Dockerfile"
    docker_build_args = jsonencode(var.build_args)
    docker_tag = "latest"
  }
}
```

dieselbe Metadatenressource, die auf eine Lambda-Layer verweist

```
resource "null_resource" "sam_metadata_layer1" {
  triggers = {
    resource_name = "aws_lambda_layer_version.layer"
    resource_type = "LAMBDA_LAYER"
    original_source_code = local.layer_src
    built_output_path = "${path.module}/${layer_build_path}"
  }
  depends_on = [null_resource.layer_build]
}
```

Was ist AWS SAM CLI Unterstützung für Terraform?

Verwenden Sie die AWS Serverless Application Model Befehlszeilenschnittstelle (AWS SAMCLI) mit Ihren Terraform Projekten oder Terraform Cloud um lokales Debugging und Testen von Folgendem durchzuführen:

- AWS Lambda Funktionen und Ebenen.
- Amazon API Gateway-APIs.

Eine Einführung in finden Sie unter Was istTerraform? Terraformauf der HashiCorp Terraform Website.

Erstellen Sie ein GitHub Problem, um Feedback zu geben und Funktionsanfragen einzureichen.

> **Note**
>
> Im Rahmen des AWS SAMCLI Parsing-Schritts der Integration generieren AWS SAMCLI Prozesse und Benutzerbefehle Projektdateien und -daten. Die Befehlsausgabe sollte unverändert bleiben, aber in bestimmten Umgebungen kann die Umgebung oder der Runner zusätzliche Protokolle oder Informationen in die Ausgabe einfügen.

Topics

- Was ist der AWS SAM CLI?
- Wie verwende ich den AWS SAM CLI mit Terraform?
- Nächste Schritte

Was ist der AWS SAM CLI?

Das AWS SAMCLI ist ein Befehlszeilentool, das Sie mit AWS SAM Vorlagen und unterstützten Integrationen von Drittanbietern verwenden können, z. B. Terraform um Ihre serverlosen Anwendungen zu erstellen und auszuführen. Eine Einführung in das finden Sie unter AWS SAMCLI. Was ist der AWS SAM CLI?

Der AWS SAMCLI unterstützt die folgenden Befehle fürTerraform:

- `sam local invoke`— Initiiert lokal einen einmaligen Aufruf einer AWS Lambda Funktionsressource. Weitere Informationen zu diesem Befehl finden Sie unterEinführung in das Testen mit sam local invoke.
- `sam local start-api`— Führen Sie Ihre Lambda-Ressourcen lokal aus und testen Sie sie über einen lokalen HTTP-Serverhost. Diese Art des Testens ist hilfreich für Lambda-Funktionen, die von einem API-Gateway-Endpunkt aufgerufen werden. Weitere Informationen zu diesem Befehl finden Sie unter. Einführung in das Testen mit dieselbe lokale Start-API
- `sam local start-lambda`— Starten Sie einen lokalen Endpunkt für Ihre Lambda-Funktion, um Ihre Funktion lokal mithilfe von AWS Command Line Interface (AWS CLI) oder SDKs aufzurufen.

Weitere Informationen zu diesem Befehl finden Sie unter. Einführung in das Testen mit sam local start-lambda

Wie verwende ich den AWS SAM CLI mit Terraform?

Der Terraform Kernworkflow besteht aus drei Phasen: Schreiben , Planen und Anwenden. Mit der AWS SAMCLI Unterstützung für Terraform können Sie die Vorteile des AWS SAMCLI `sam local` Befehlssatzes nutzen und gleichzeitig Ihre Terraform Workflows weiterhin zur Verwaltung Ihrer Anwendungen verwenden AWS. Im Allgemeinen bedeutet dies Folgendes:

- Schreiben — Erstellen Sie Ihre Infrastruktur als Code mithilfe vonTerraform.
- Testen und Debuggen — Verwenden Sie das AWS SAMCLI, um Ihre Anwendungen lokal zu testen und zu debuggen.
- Plan — Sehen Sie sich vor der Anwendung eine Vorschau der Änderungen an.
- Bewerben — Stellen Sie Ihre Infrastruktur bereit.

Ein Beispiel für die Verwendung von AWS SAMCLI with Terraform finden Sie unter Better together: AWS SAMCLI und HashiCorp Terraform im AWS Compute-Blog.

Nächste Schritte

Informationen zum Erfüllen aller Voraussetzungen und zur Einrichtung Terraform finden Sie unterErste Schritte mit Terraform Unterstützung für AWS SAM CLI.

Veröffentlichen Sie Ihre Bewerbung mit dem AWS SAMCLI

Um Ihre AWS SAM Anwendung anderen Benutzern zum Suchen und Bereitstellen zur Verfügung AWS SAMCLI zu stellen, können Sie sie mit dem veröffentlichen AWS Serverless Application Repository. Um Ihre Anwendung mit dem zu veröffentlichen AWS SAMCLI, müssen Sie sie mithilfe einer AWS SAM Vorlage definieren. Sie müssen es auch lokal oder in der AWS Cloud getestet haben.

Folgen Sie den Anweisungen in diesem Thema, um eine neue Anwendung zu erstellen, eine neue Version einer vorhandenen Anwendung zu erstellen oder die Metadaten einer vorhandenen Anwendung zu aktualisieren. (Was Sie tun, hängt davon ab AWS Serverless Application Repository, ob die Anwendung bereits in der vorhanden ist und ob sich Anwendungsmetadaten ändern.) Weitere Informationen zu Anwendungsmetadaten finden Sie unterAWS SAM Eigenschaften des Vorlagen-Metadatenabschnitts.

Voraussetzungen

Bevor Sie eine Anwendung AWS Serverless Application Repository unter Verwendung von veröffentlichen AWS SAMCLI, müssen Sie über Folgendes verfügen:

- Das AWS SAMCLI ist installiert. Weitere Informationen finden Sie unter Installieren Sie das AWS SAMCLI. Führen Sie den folgenden Befehl aus, um festzustellen, ob der installiert AWS SAMCLI ist:

  ```
  sam --version
  ```

- Eine gültige AWS SAM Vorlage.
- Ihr Anwendungscode und die Abhängigkeiten, auf die die AWS SAM Vorlage verweist.
- Eine semantische Version, die nur erforderlich ist, um Ihre Anwendung öffentlich zu teilen. Dieser Wert kann so einfach wie 1,0 sein.
- Eine URL, die auf den Quellcode Ihrer Anwendung verweist.
- Die Datei `README.md`. In dieser Datei sollte beschrieben werden, wie Kunden Ihre Anwendung verwenden können und wie sie konfiguriert wird, bevor sie sie in ihren eigenen AWS Konten bereitstellen.
- Eine `LICENSE.txt` Datei, die nur benötigt wird, um Ihre Anwendung öffentlich zu teilen.

- Wenn Ihre Anwendung verschachtelte Anwendungen enthält, müssen Sie diese bereits in der AWS Serverless Application Repository veröffentlicht haben.
- Eine gültige Bucket-Richtlinie von Amazon Simple Storage Service (Amazon S3), die dem Service Leseberechtigungen für Artefakte gewährt, die Sie beim Verpacken Ihrer Anwendung auf Amazon S3 hochladen. Gehen Sie wie folgt vor, um diese Richtlinie einzurichten:

  1. Öffnen Sie die Amazon S3 S3-Konsole unter https://console.aws.amazon.com/s3/.
  2. Wählen Sie den Namen des Amazon S3 S3-Buckets, den Sie zum Verpacken Ihrer Anwendung verwendet haben.
  3. Wählen Sie Berechtigungen.
  4. Wählen Sie auf der Registerkarte Berechtigungen unter Bucket-Richtlinie die Option Bearbeiten aus.
  5. Fügen Sie auf der Seite Bucket-Richtlinie bearbeiten die folgende Richtlinienerklärung in den Policy-Editor ein. Stellen Sie sicher, dass Sie in der Richtlinienerklärung Ihren Bucket-Namen im `Resource` Element und Ihre AWS Konto-ID im `Condition` Element verwenden. Der Ausdruck im `Condition` Element stellt sicher, AWS Serverless Application Repository dass Sie nur über das angegebene AWS Konto auf Anwendungen zugreifen dürfen. Weitere Informationen zu Richtlinienerklärungen finden Sie in der Referenz zu den IAM-JSON-Richtlinienelementen im IAM-Benutzerhandbuch.

     JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service":  "serverlessrepo.amazonaws.com"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::<your-bucket-name>/*",
            "Condition" : {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                }
            }
        }
    ]
```

```
}
```

6. Wählen Sie Änderungen speichern aus.

Eine neue Anwendung veröffentlichen

Schritt 1: Fügen Sie der AWS SAM Vorlage einen **`Metadata`** Abschnitt hinzu

Fügen Sie Ihrer AWS SAM Vorlage zunächst einen `Metadata` Abschnitt hinzu. Geben Sie die Anwendungsinformationen an, die auf der veröffentlicht werden sollen AWS Serverless Application Repository.

Im Folgenden finden Sie einen `Metadata` Beispielabschnitt:

```
Metadata:
  AWS::ServerlessRepo::Application:
    Name: my-app
    Description: hello world
    Author: user1
    SpdxLicenseId: Apache-2.0
    LicenseUrl: LICENSE.txt
    ReadmeUrl: README.md
    Labels: ['tests']
    HomePageUrl: https://github.com/user1/my-app-project
    SemanticVersion: 0.0.1
    SourceCodeUrl: https://github.com/user1/my-app-project

Resources:
  HelloWorldFunction:
    Type: AWS::Lambda::Function
      Properties:
        ...
        CodeUri: source-code1
        ...
```

Weitere Informationen zu `Metadata` diesem Abschnitt der AWS SAM Vorlage finden Sie unterAWS SAM Eigenschaften des Vorlagen-Metadatenabschnitts.

Schritt 2: Verpacken Sie die Anwendung

Führen Sie den folgenden AWS SAMCLI Befehl aus, der die Artefakte der Anwendung auf Amazon S3 hochlädt und eine neue Vorlagendatei mit dem Namen `packaged.yaml` ausgibt:

```
sam package --output-template-file packaged.yaml --s3-bucket <your-bucket-name>
```

Sie verwenden die `packaged.yaml` Vorlagendatei im nächsten Schritt, um die Anwendung auf dem AWS Serverless Application Repository zu veröffentlichen. Diese Datei ähnelt der ursprünglichen Vorlagendatei (`template.yaml`), weist jedoch einen wesentlichen Unterschied auf: Die `ReadmeUrl` Eigenschaften `CodeUriLicenseUrl`, und verweisen auf den Amazon S3 S3-Bucket und Objekte, die die entsprechenden Artefakte enthalten.

Der folgende Ausschnitt aus einer `packaged.yaml`-Beispielvorlagendatei zeigt die `CodeUri`-Eigenschaft:

```
MySampleFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: s3://bucketname/fbd77a3647a4f47a352fc0bjectGUID

...
```

Schritt 3: Veröffentlichen Sie die Anwendung

Führen Sie den folgenden AWS SAMCLI Befehl aus AWS Serverless Application Repository, um eine private Version Ihrer AWS SAM Anwendung auf dem zu veröffentlichen:

```
sam publish --template packaged.yaml --region us-east-1
```

Die Ausgabe des `sam publish` Befehls enthält einen Link zu Ihrer Anwendung auf der AWS Serverless Application Repository. Sie können auch direkt zur AWS Serverless Application Repository Landingpage gehen und nach Ihrer Anwendung suchen.

Schritt 4: Teilen Sie die Anwendung (optional)

Standardmäßig ist Ihre Anwendung auf privat eingestellt, sodass sie für andere AWS Konten nicht sichtbar ist. Um Ihre Anwendung mit anderen zu teilen, müssen Sie sie entweder öffentlich machen oder einer bestimmten Liste von AWS Konten die Erlaubnis erteilen.

Informationen zur gemeinsamen Nutzung Ihrer Anwendung mithilfe von finden Sie unter Beispiele für AWS Serverless Application Repository ressourcenbasierte Richtlinien im AWS Serverless Application Repository Entwicklerhandbuch. AWS CLI Informationen zur gemeinsamen Nutzung Ihrer Anwendung mithilfe von finden Sie unter Gemeinsame Nutzung einer Anwendung im AWS Serverless Application Repository Entwicklerhandbuch. AWS-Managementkonsole

Veröffentlichen einer neuen Version einer vorhandenen Anwendung

Nachdem Sie eine Anwendung auf dem veröffentlicht haben AWS Serverless Application Repository, möchten Sie möglicherweise eine neue Version davon veröffentlichen. Möglicherweise haben Sie Ihren Lambda-Funktionscode geändert oder Ihrer Anwendungsarchitektur eine neue Komponente hinzugefügt.

Um eine Anwendung zu aktualisieren, die Sie zuvor veröffentlicht haben, veröffentlichen Sie die Anwendung erneut, indem Sie denselben Vorgang verwenden, der zuvor beschrieben wurde. Geben Sie im `Metadata` Abschnitt der AWS SAM Vorlagendatei denselben Anwendungsnamen ein, mit dem Sie sie ursprünglich veröffentlicht haben, fügen Sie jedoch einen neuen `SemanticVersion` Wert hinzu.

Stellen Sie sich beispielsweise eine Anwendung vor, die mit dem Namen `SampleApp` und einem `SemanticVersion` von veröffentlicht wurde`1.0.0`. Um diese Anwendung zu aktualisieren, muss die AWS SAM Vorlage den Namen der Anwendung `SampleApp` und ein `SemanticVersion` von `1.0.1` (oder etwas anderes als`1.0.0`) enthalten.

Weitere Themen

- AWS SAM Eigenschaften des Vorlagen-Metadatenabschnitts

AWS SAM Eigenschaften des Vorlagen-Metadatenabschnitts

`AWS::ServerlessRepo::Application`ist ein Metadatenschlüssel, mit dem Sie Anwendungsinformationen angeben können, die Sie in der veröffentlichen möchten AWS Serverless Application Repository.

> **Note**
>
> CloudFormation Systeminterne Funktionen werden vom `AWS::ServerlessRepo::Application` Metadatenschlüssel nicht unterstützt.

Eigenschaften

Diese Tabelle enthält Informationen zu den Eigenschaften des `Metadata` Abschnitts der AWS SAM Vorlage. Dieser Abschnitt ist erforderlich, um Anwendungen AWS Serverless Application Repository unter Verwendung von zu veröffentlichen AWS SAMCLI.

Eigenschaft	Typ	Erforderlich	Beschreibung
`Name`	Zeichenfolge	TRUE	Der Name der Anwendung. Minimale Länge = 1. Maximale Länge = 140. Pattern: `"[a-zA-Z0-9\\-]+";`
`Description`	Zeichenfolge	TRUE	Die Beschreibung der Anwendung. Minimale Länge = 1. Maximale Länge = 256.
`Author`	Zeichenfolge	TRUE	Der Name des Autors, der die Anwendung veröffentlicht. Minimale Länge = 1. Maximale Länge = 127. Pattern: `"^[a-z0-9](([a-z0-9]|-(?!-))*[a-z0-9])?$";`
`SpdxLicenseId`	Zeichenfolge	FALSE	Eine gültige Lizenz-ID. Eine Liste der gültigen Lizenzkennungen finden Sie in der SPDX-Lizenzliste auf der Software Package Data Exchange (SPDX) - Website.

Eigenschaft	Typ	Erforderlich	Beschreibung
`LicenseUrl`	Zeichenfolge	FALSE	Der Verweis auf eine lokale Lizenzdatei oder ein Amazon S3 S3-Link zu einer Lizenzdatei, der dem spdxLicenseID-Wert Ihrer Anwendung entspricht. Eine AWS SAM Vorlagendatei, die nicht mit dem `sam package` Befehl gepackt wurde, kann einen Verweis auf eine lokale Datei für diese Eigenschaft enthalten. Damit eine Anwendung mit dem `sam publish` Befehl veröffentlicht werden kann, muss diese Eigenschaft jedoch ein Verweis auf einen Amazon S3 S3-Bucket sein. Maximale Größe: 5 MB. Sie müssen einen Wert für diese Eigenschaft angeben, um Ihre Anwendung öffentlich zu machen. Beachten Sie, dass Sie diese Eigenschaft nicht aktualisieren können, nachdem Ihre Anwendung veröffentlicht wurde. Um einer Anwendung eine Lizenz hinzuzufügen, müssen Sie sie entweder zuerst löschen oder eine neue Anwendung mit einem anderen Namen veröffentlichen.

Eigenschaft	Typ	Erforderlich	Beschreibung
`ReadmeUrl`	Zeichenfolge	FALSE	Der Verweis auf eine lokale Readme-Datei oder ein Amazon S3 S3-Link zur Readme-Datei, die eine detailliertere Beschreibung der Anwendung und ihrer Funktionsweise enthält. Eine AWS SAM Vorlagendatei, die nicht mit dem `sam package` Befehl gepackt wurde, kann einen Verweis auf eine lokale Datei für diese Eigenschaft enthalten . Um mit dem `sam publish` Befehl veröffentlicht zu werden, muss diese Eigenschaft jedoch ein Verweis auf einen Amazon S3 S3-Bucket sein. Maximale Größe: 5 MB.
`Labels`	Zeichenfolge	FALSE	Die Bezeichnungen, die das Auffinden von Anwendungen in den Suchergebnissen verbessern. Minimale Länge = 1. Maximale Länge = 127. Maximale Anzahl der Etiketten: 10. Pattern: `"^[a-zA-Z0-9+\\-_:\\/@]+$";`
`HomePageUrl`	Zeichenfolge	FALSE	Eine URL mit weiteren Informationen über die Anwendung, z. B. den Speicherort Ihres GitHub Repositorys für die Anwendung.
`SemanticVersion`	Zeichenfolge	FALSE	Die semantische Version der Anwendung. Die Semantic Versioning-Spezifikation finden Sie auf der Semantic Versioning-Website. Sie müssen einen Wert für diese Eigenschaft angeben, um Ihre Anwendung öffentlich zu machen.
`SourceCodeUrl`	Zeichenfolge	FALSE	Ein Link zu einem öffentlichen Repository für den Quellcode Ihrer Anwendung.

Anwendungsfälle

In diesem Abschnitt werden die Anwendungsfälle für die Veröffentlichung von Anwendungen zusammen mit den `Metadata` Eigenschaften aufgeführt, die für diesen Anwendungsfall verarbeitet werden. Eigenschaften, die für einen bestimmten Anwendungsfall nicht aufgeführt sind, werden ignoriert.

- Eine neue Anwendung erstellen — Eine neue Anwendung wird erstellt, wenn in der keine Anwendung AWS Serverless Application Repository mit einem passenden Namen für ein Konto vorhanden ist.
  - `Name`
  - `SpdxLicenseId`
  - `LicenseUrl`
  - `Description`
  - `Author`
  - `ReadmeUrl`
  - `Labels`
  - `HomePageUrl`
  - `SourceCodeUrl`
  - `SemanticVersion`
  - Der Inhalt der AWS SAM Vorlage (z. B. alle Eventquellen, Ressourcen und Lambda-Funktionscode)

- Anwendungsversion erstellen — Eine Anwendungsversion wird erstellt, wenn in der bereits eine Anwendung AWS Serverless Application Repository mit einem passenden Namen für ein Konto vorhanden SemanticVersion ist und diese sich ändert.
  - `Description`
  - `Author`
  - `ReadmeUrl`
  - `Labels`
  - `HomePageUrl`
  - `SourceCodeUrl`

  - `SemanticVersion`
  - Der Inhalt der AWS SAM Vorlage (z. B. alle Eventquellen, Ressourcen und Lambda-Funktionscode)

- Aktualisierung einer Anwendung — Eine Anwendung wird aktualisiert, wenn in der bereits eine Anwendung AWS Serverless Application Repository mit einem passenden Namen für ein Konto vorhanden SemanticVersion ist und die sich nicht ändert.
  - `Description`
  - `Author`
  - `ReadmeUrl`
  - `Labels`
  - `HomePageUrl`

Beispiel

Im Folgenden finden Sie einen `Metadata` Beispielabschnitt:

```
Metadata:
  AWS::ServerlessRepo::Application:
    Name: my-app
    Description: hello world
    Author: user1
    SpdxLicenseId: Apache-2.0
    LicenseUrl: LICENSE.txt
    ReadmeUrl: README.md
    Labels: ['tests']
    HomePageUrl: https://github.com/user1/my-app-project
    SemanticVersion: 0.0.1
    SourceCodeUrl: https://github.com/user1/my-app-project
```

Dokumentenverlauf für AWS SAM

In der folgenden Tabelle werden die wichtigen Änderungen in den einzelnen Versionen des AWS Serverless Application Model Entwicklerhandbuchs beschrieben. Um Benachrichtigungen über Aktualisierungen dieser Dokumentation zu erhalten, können Sie einen RSS-Feed abonnieren.

- Letzte Aktualisierung der Dokumentation: 20. Juni 2024

Änderung	Beschreibung	Datum
AWS SAMCLIUnterstützung für die `AWS::LanguageExtensions` Transformation wurde hinzugefügt	Das erweitert AWS SAMCLI jetzt`Fn::ForEach` ,`Fn::Length` , und `Fn::FindInMap` mit `DefaultValue` lokal`Fn::ToJsonString` , also,,,`sam build`,,`sam package`,`sam deploy`,,`sam sync`,`sam validate`,,, und die `sam local` Befehle funktionieren mit Vorlagen, die die `AWS::LanguageExtensions` Transformation verwenden. Weitere Informationen finden Sie unter Unterstützung für CloudFormation Spracherweiterungen.	13. Mai 2026
Der gesamte Inhalt des Entwicklerhandbuchs wurde neu strukturiert und aktualisiert	Der Leitfaden wurde neu organisiert und neu strukturiert, um die Auffindbarkeit und Benutzerfreundlichkeit zu verbessern. Aktualisierte und verbesserte Titel. Bei der Einführung von Themen und	20. Juni 2024

	Konzepten wurden zusätzliche Details bereitgestellt.	
AWS SAMCLIUnterstützung für Ruby 3.3 hinzugefügt	Ruby 3.3 ist jetzt als Laufzeit- und Image-Repository verfügbar. Einzelheiten finden Sie unter Image-Repositorys und Sam Init.	4. April 2024
Befehlsoptionen wurden hinzugefügt AWS SAMCLI	Neue Optionen sind für den Befehl sam local start-api:`--ssl-cert-file PATH` , verfügbar. `--ssl-key-file PATH` Zusätzlich `--add-host LIST` ist die neue Befehlszeilenoption für sam local invoke, sam localstart-api und sam local start-lambda verfügbar	20. März 2024
Unterstützung für.NET 8 wurde hinzugefügt AWS SAMCLI	.NET 8 ist jetzt als Laufzeit- und Image-Repository verfügbar. Laufzeiten und Image-Repositorys für.NET Core 3.1, Node.js 14, Node.js 12, Python 3.7, Ruby 2.7 werden nicht mehr unterstützt. Siehe Image-Repositorys und Sam Init.	22. Februar 2024
Das Installationsprogramm für das AWS SAMCLI arm64-Paket wurde hinzugefügt Linux	Anweisungen finden Sie unter Installation von. AWS SAMCLI	6. Dezember 2023

Die Option --watch-exclude für den AWS SAMCLI Befehl sam sync wurde hinzugefügt	Schließt Dateien und Ordner von der Initiierung einer Synchronisation aus. Weitere Informationen finden Sie unter Angeben von Dateien und Ordnern, die keine Synchronisierung initiieren.	6. Dezember 2023
Die Option --build-in-source wurde für den Befehl sam sync hinzugefügt AWS SAMCLI	Erstellen Sie Ihr Projekt in Ihrem Quellordner, um den Erstellungsprozess zu beschleunigen. Weitere Informationen finden Sie unter Beschleunigen Sie die Build-Zeiten, indem Sie Ihr Projekt im Quellordner erstellen.	6. Dezember 2023
Die Option --build-in-source wurde für den Befehl sam build hinzugefügt AWS SAMCLI	Erstellen Sie Ihr Projekt in Ihrem Quellordner, um den Erstellungsprozess zu beschleunigen. Weitere Informationen finden Sie unter Beschleunigen Sie die Build-Zeiten, indem Sie Ihr Projekt im Quellordner erstellen.	6. Dezember 2023
Neue Ressourcenunterstützung für den Befehl AWS SAMCLI Remote Invoke hinzugefügt	Zur Verwendung `sam remote invoke` mit Kinesis Data Streams Streams-Anwendungen, Amazon SQS SQS-Warteschlangen und Step Functions Functions-Zustandsmaschinen. Weitere Informationen finden Sie unter Sam Remote Invoke verwenden.	15. November 2023

Neuer AWS SAMCLI Remote-Test-Event-Befehl für gemeinsam nutzbare Testereignisse hinzugefügt	Verwenden Sie die AWS SAM CLI, um auf gemeinsam nutzbare Testereignisse aus der EventBridge Schemareg istry zuzugreifen und diese zu verwalten, um Ihre Lambda-Fu nktionen in der zu testen. AWS Cloud Weitere Informationen finden Sie unter Sam remote test-event verwenden.	3. Oktober 2023
AWS SAMCLIUnterstützun g für Terraform ist jetzt allgemein verfügbar	Weitere Informationen zum AWS SAMCLI Support für Terraform finden Sie unter AWS SAMCLITerraformSup port.	5. September 2023
AWS SAMCLIUnterstützun g wurde hinzugefügt für Terraform Cloud	Das unterstützt AWS SAMCLI jetzt lokale Tests fürTerraform Cloud. Weitere Informationen finden Sie unter Einrichten für Terraform Cloud.	5. September 2023
Unterstützung für YAML Dateiformate für die AWS SAMCLI Konfigurationsdatei hinzugefügt	Das unterstützt AWS SAMCLI jetzt das `[.yaml|.yml]` - Dateiformat. Die Seiten zur AWS SAM CLIKonfig uration AWS SAMCLI und zur Konfigurationsdatei wurden aktualisiert.	18. Juli 2023

AWS SAMCLIsam local start-apiBefehlsunterstützung für hinzugefügt Terraform	Wozu dient AWS SAMCLI die UnterstützungTerraform? Der Abschnitt wurde aktualisiert und enthält nun auch AWS SAMCLI `sam local start-api` Befehlsunterstützung fürTerraform.	6. Juli 2023
Neuer Befehl zum Aufrufen AWS SAMCLI per Fernzugriff hinzugefügt	Informationen zum Einstieg finden Sie `sam remote invoke` unter Sam Remote Invoke verwenden.	22. Juni 2023
AWS AppSyncGraphQL APIServerloser Ressourcentyp hinzugefügt	Erstellen Sie einen neuen `AWS::Serverless::GraphQLApi` Abschnitt, der beschreibt, wie Sie eine GraphQL API Ressource mit AWS SAM definieren.	22. Juni 2023
AWS SAMCLIUnterstützung für Ruby 3.2 wurde hinzugefügt	Aktualisieren Sie die Sam-Init-Seite, sodass sie neue Basis-Image- und Laufzeitwerte enthält. Aktualisieren Sie die Seite mit den Image-Repositorys mit Ruby 3.2 Amazon ECR URI.	6. Juni 2023
Optionale Schritte zur Integritätsprüfung des Paket-Installationsprogramms hinzugefügt AWS SAMCLI	Aktualisieren Sie die Installation der AWS SAMCLI Seite, um den optionalen Schritt widerzuspiegeln. Erstellen Überprüfen Sie die Integrität der AWS SAMCLI Installationsseite, um die Schritte zu dokumentieren.	31. Mai 2023

Die Option Sam Sync wurde hinzugefügt, um die Infrastruktursynchronisierung zu überspringen	Passen Sie an, ob bei jeder Ausführung eine CloudFormation Bereitstellung erforderlich `sam sync` ist. Weitere Informationen finden Sie unter Überspringen der ersten CloudFormation Bereitstellung.	23. März 2023
Unterstützung für den Eventquellentyp DocumentDB hinzugefügt	Die AWS SAM Vorlagenspezifikation unterstützt jetzt den `DocumentDB` Ereignisquellentyp für die `AWS::Serverless::Function` Ressource. Weitere Informationen finden Sie unter DocumentDB.	10. März 2023
Erstellen Sie Rust Lambda-Funktionen mit Cargo Lambda	Verwenden Sie die AWS SAMCLI, um Ihre Rust-Lambda-Funktionen mit Cargo Lambda zu erstellen. Weitere Informationen finden Sie unter Erstellen von Rust Lambda-Funktionen mit Cargo Lambda.	23. Februar 2023
Erstellen Sie Funktionsressourcen außerhalb von AWS SAM	Es wurde eine Anleitung zum Überspringen von Funktionen bei der Verwendung des sam build Befehls hinzugefügt. Weitere Informationen finden Sie unter Funktionen außerhalb von AWS SAM erstellen.	14. Februar 2023

Neue Syntax für eingebettete Konnektoren	Verwenden Sie die neue Syntax für eingebettete Konnektoren, um Ihre `AWS::Serverless::Connector` Ressourcen zu definieren. Weitere Informationen finden Sie unter Ressourcenberechtigungen mit AWS SAM Konnektoren verwalten.	8. Februar 2023
Es wurde ein neuer Befehl sam list für hinzugefügt AWS SAMCLI	sam listDient zum Anzeigen wichtiger Informationen über die Ressourcen in Ihrer serverlosen Anwendung. Weitere Informationen finden Sie in der Sam-Liste.	2. Februar 2023
Format- und OutExtension Build-Eigenschaften für Esbuild hinzugefügt	Das Erstellen von Node.js Lambda-Funktionen mit esbuild unterstützt `Format` und `OutExtension` erstellt jetzt Eigenschaften. Weitere Informationen finden Sie unter Node.js Lambda-Funktionen mit esbuild erstellen.	2. Februar 2023
Der Vorlagenspezifikation wurden Optionen zur Laufzeitverwaltung hinzugefügt AWS SAM	Konfigurieren Sie Laufzeitverwaltungsoptionen für Ihre Lambda-Funktionen. Weitere Informationen hierzu finden Sie unter `RuntimeManagementConfig`.	24. Januar 2023

Die Zieleigenschaft wurde zu EventSource for AWS::Serverless:: StateMachine resource hinzugefügt.	`AWS::Serverless::StateMachine` Der Ressourcentyp unterstützt die `Target` Eigenschaft für `EventBridgeRule` und die `Schedule` Ereignisquelle.	13. Januar 2023
Skalierung von SQS-Pollern für Lambda-Funktionen konfigurieren	Konfigurieren Sie die Skalierung von SQS-Pollern mit der Eigenschaft für. `ScalingConfig` `AWS::Serverless::Function` Weitere Informationen hierzu finden Sie unter `ScalingConfig` .	12. Januar 2023
Validieren Sie AWS SAM Anwendungen mit cfn-lint	Sie können cfn-lint verwenden, um Ihre AWS SAM Vorlagen über die zu validieren. AWS SAMCLI Weitere Informationen finden Sie unter Mit cfn-lint validieren.	11. Januar 2023
Überwachen Sie Ihre serverlosen Anwendungen mit Application Insights CloudWatch	Konfigurieren Sie Amazon CloudWatch Application Insights, um Ihre AWS SAM Anwendungen zu überwachen. Weitere Informationen finden Sie unter Überwachen Sie Ihre serverlosen Anwendungen mit CloudWatch Application Insights.	19. Dezember 2022

AWS SAMCLIPaket-Installer für macOS hinzugefügt	Installieren Sie das AWS SAMCLI mit dem neuen macOS-Paketinstallationspro gramm. Weitere Informationen finden Sie unter Installation von AWS SAMCLI.	6. Dezember 2022
Unterstützung für Lambda hinzugefügt SnapStart	Konfigurieren Sie SnapStart Ihre Lambda-Funktionen so, dass sie Snapshots erstellen. Dabei handelt es sich um zwischengespeicher te Zustände Ihrer initialisierten Funktionen. Weitere Informati onen hierzu finden Sie unter `AWS::Serverless::F unction` .	28. November 2022
Unterstützung für nodejs18.x hinzugefügt AWS SAMCLI	AWS SAMCLIunterstützt jetzt die Laufzeit von nodejs18.x. Weitere Informationen finden Sie unter sam init.	17. November 2022
Es wurden Anleitungen zur Konfiguration von Zugriff und Berechtigungen hinzugefügt	AWS SAM bietet zwei Optionen, die die Zugriffs- und Berechtigungsverwa ltung für Ihre serverlosen Anwendungen vereinfachen. Weitere Informationen finden Sie unter Ressourcenzugriff und Berechtigungen verwalten .	17. November 2022

Unterstützung für die Erstellung von.NET 7 Lambda-Funktionen mit nativer AOT-Kompilierung hinzugefügt	Erstellen und verpacken Sie Ihre.NET 7 Lambda-Funktionen mit und nutzen Sie die native Ahead-of-Time (AOT) -Kompilierung AWS SAM, um die Lambda-Kaltstartzeiten zu verbessern. Weitere Informationen finden Sie unter Erstellen von.NET 7 Lambda-Funktionen mit nativer AOT-Kompilierung.	15. November 2022
AWS SAMCLITerraformUnterstützung für lokales Debuggen und Testen hinzugefügt	Verwenden Sie das AWS SAMCLI in Ihren Terraform Projekten, um Ihre Lambda-Funktionen und -Layer lokal zu debuggen und zu testen. Weitere Informationen finden Sie unter AWS SAMTerraformCLI-Unterstützung.	14. November 2022
AWS SAM Unterstützung für EventBridge Scheduler hinzugefügt	Die Vorlagenspezifikation AWS Serverless Application Model (AWS SAM) bietet eine einfache, kurze Syntax, die Sie verwenden können, um Ereignisse mit dem EventBridge Scheduler für und zu planen. AWS Lambda AWS Step Functions Weitere Informationen finden Sie unter Ereignisse mit EventBridge Scheduler planen.	10. November 2022

Die AWS SAMCLI Installationsanweisungen wurden vereinfacht	AWS SAMCLIVoraussetzungen und optionale Schritte wurden auf separate Seiten verschoben. Installationsschritte für unterstützte Betriebssysteme finden Sie unter Installation von AWS SAMCLI.	04. November 2022
Es wurde ein Fix hinzugefügt, um lange Pfade für Windows 10-Benutzer zuzulassen	Das AWS SAMCLI App-Vorlagen-Repository enthält einige lange Dateipfade, die `sam init` aufgrund von Windows MAX_PATH 10-Einschränkungen zu Fehlern bei der Ausführung führen können. Weitere Informationen finden Sie unter Installation des AWS SAMCLI	04. November 2022
Der schrittweise Bereitstellungsprozess für Erstinstallationen wurde aktualisiert	Die schrittweise Bereitstellung einer Lambda-Funktion mit AWS CodeDeploy erfordert zwei Schritte. Weitere Informationen finden Sie unter Schrittweises Bereitstellen einer Lambda-Funktion zum ersten Mal.	13. Oktober 2022
Zusätzliche Unterstützung der Lambda-Ereignisfilterung für mehr Ereignistypen	`FilterCritera` Eigenschaften wurden den Typen „`MSKMQ`," und „`SelfManagedKafka` Ereignisquellentypen" hinzugefügt.	13. Oktober 2022

OpenID Connect (OIDC) - Unterstützung für Pipeline hinzugefügt AWS SAM	AWS SAM unterstützt die OpenID Connect (OIDC) - Benutzerauthentifizierung für Bitbucket-, GitHub Actions- und Continuous-Integrations- und GitLab Continuous-Delivery-Plattformen (). CI/CD Weitere Informationen findest du unter OIDC-Benutzerkonten mit Pipeline verwenden. AWS SAM	13. Oktober 2022
Hinweis zu Eigenschaften JwtConfiguration	Ein Hinweis zur Definition `issuer` und zu `audience` Eigenschaften wurde unter `JwtConfiguration` für hinzugefügt `OAuth2Authorizer`.	7. Oktober 2022
Neue Eigenschaften für Function und StateMachine EventSource	`Enabled`und `State` Eigenschaften wurden zur `CloudWatchEvent` Ereignisquelle für hinzugefügt `AWS::Serverless::Function`. `State`Eigenschaft zur `Schedule` Ereignisquelle für `AWS::Serverless::Function` und hinzugefügt `AWS::Serverless::StateMachine`.	6. Oktober 2022

AWS SAM Konnektoren sind jetzt allgemein verfügbar	Konnektoren sind ein AWS SAM abstrakter RessourcentypAWS::Serverless::Connector , der als bezeichnet wird und eine einfache und sichere Methode zur Bereitstellung von Berechtigungen zwischen Ihren serverlosen Anwendungsressourcen bietet. Weitere Informationen finden Sie unter Ressourcenberechtigungen mit AWS Serverless Application Model Konnektoren verwalten.	6. Oktober 2022
Neue Sam-Sync-Optionen wurden dem hinzugefügt AWS SAMCLI	--dependency-layer und --use-container Optionen wurden hinzugefügt sam sync.	20. September 2022
Neue Sam-Deploy-Optionen wurden dem hinzugefügt AWS SAMCLI	--on-failure Option hinzugefügt zu sam deploy.	09. September 2022
Esbuild-Unterstützung ist jetzt allgemein verfügbar	Um Node.js Lambda-Funktionen zu erstellen und zu verpacken, können Sie den AWS SAMCLI mit dem JavaScript Esbuild-Bundler verwenden.	01. September 2022

Telemetrie aktualisiert AWS SAMCLI	Die Beschreibung der gesammelten System- und Umgebungsinformationen wurde aktualisiert und enthält nun Hashwerte von Nutzungsattributen.	01. September 2022
Unterstützung für lokale Umgebungsvariablen hinzugefügt AWS SAMCLI	Verwenden Sie Umgebungsvariablen mit AWS SAMCLI, wenn Sie Lambda-Funktionen lokal aufrufen und API Gateway lokal ausführen.	01. September 2022
Support für Lambda-Befehlssatzarchitekturen	Verwenden Sie die AWS SAMCLI, um Lambda-Funktionen und Lambda-Schichten für unsere `x86_64` `arm64` Befehlssatzarchitekturen zu erstellen. Weitere Informationen finden Sie unter den Architectures-Eigenschaften des `AWS::Serverless::Function` Ressourcentyps und der CompatibleArchitecturesEigenschaft des Ressourcentyps. `AWS::Serverless::LayerVersion`	1. Oktober 2021

Generieren von Beispiel-Pipeline-Konfigurationen	Verwenden Sie die AWS SAMCLI, um mithilfe der neuen `sam pipeline init`Befehle `sam pipeline bootstrap` und Beispiel-Pipelines für mehrere CI/CD Systeme zu generieren. Weitere Informationen finden Sie unter Generieren von CI/CD Beispiel-Pipelines.	21. Juli 2021
AWS SAMCLIAWS CDK Integration (Vorschau, Phase 2)	In Phase 2 der öffentlichen Vorschauversion können Sie die jetzt AWS SAMCLI zum Verpacken und Bereitstellen von AWS CDK Anwendungen verwenden. Sie können eine AWS CDK Beispielanwendung auch direkt mit dem herunterl aden AWS SAMCLI. Weitere Informationen finden Sie unter AWS Cloud Development Kit (AWS CDK) (Vorschau).	13. Juli 2021
Support für RabbitMQ als Eventquelle für Funktionen	Unterstützung für RabbitMQ als Ereignisquelle für serverlose Funktionen hinzugefügt. Weitere Informati onen finden Sie in der `SourceAccessConfig urations` Eigenschaft der `MQ` Ereignisquelle des Ressourcentyps. `AWS::Serv erless::Function`	7. Juli 2021

Bereitstellung serverloser Anwendungen mithilfe von Amazon ECR Build Container-Images	Verwenden Sie Amazon ECR Build Container-Images, um serverlose Anwendungen mit gängigen CI/CD Systemen wie Jenkins AWS CodePipeline und Actions bereitzustellen. GitLab CI/CD GitHub Weitere Informationen finden Sie unter Bereitstellen serverloser Anwendungen.	24. Juni 2021
Debuggen von AWS SAM Anwendungen mit Toolkits AWS	AWS Toolkits unterstützen jetzt schrittweises Debuggen mit mehr Kombinationen aus integrierten Entwicklu ngsumgebungen (IDEs) und Laufzeiten. Weitere Informationen finden Sie unter Verwenden von Toolkits. AWS	20. Mai 2021
AWS SAMCLIAWS CDK Integration (Vorschau)	Sie können das jetzt verwenden AWS SAMCLI, um AWS CDK Anwendung en lokal zu testen und zu erstellen. Dies ist eine öffentlic he Vorschauversion. Weitere Informationen finden Sie unter AWS Cloud Development Kit (AWS CDK) (Vorschau).	29. April 2021
Das Standard-Container-Image-Repository wurde zu Amazon ECR Public geändert	Das Standard-Container-Image-Repository wurde von DockerHub Amazon ECR Public geändert. Weitere Informationen finden Sie unter Image-Repositorys.	6. April 2021

Nächtliche Builds AWS SAMCLI	Sie können jetzt eine Vorabversion von installieren AWS SAMCLI, die jede Nacht erstellt wird. Weitere Informationen finden Sie im Abschnitt Nightly Build des Unterthemas Betriebssystem Ihrer Wahl unter Installation von. AWS SAMCLI	25. März 2021
Unterstützung für Build-Container-Umgebungsvariablen	Sie können jetzt Umgebungsvariablen an Build-Container übergeben. Weitere Informationen finden Sie unter den `--container-env-var-file` Optionen `--container-env-var` und `sam build`.	4. März 2021
Neuer Linux-Installationsprozess	Sie können das jetzt AWS SAMCLI mit einem nativen Linux-Installationsprogramm installieren. Weitere Informationen finden Sie unter Installation von AWS SAMCLI unter Linux.	10. Februar 2021

Support für Warteschlangen mit unzustellbaren Briefen für EventBridge	Unterstützung für Warteschlangen mit uneingeschränkten Nachrichten EventBridge und `Schedule` Ereignisquellen für serverlose Funktionen und Zustandsmaschinen wurde hinzugefügt. Weitere Informationen finden Sie unter den `DeadLetterConfig` Eigenschaften `EventBridgeRule` und den `Schedule` Ereignisquellen für die Ressourcentypen und. `AWS::Serverless::Function` `AWS::Serverless::StateMachine`	29. Januar 2021
Support für benutzerdefinierte Checkpoints	Unterstützung für benutzerdefinierte Checkpoints für DynamoDB- und Kinesis-Ereignisquellen für serverlose Funktionen hinzugefügt. Weitere Informationen finden Sie unter den `FunctionResponseTypes` Eigenschaften `Kinesis`und den `DynamoDB`Datentypen des Ressourcentyps. `AWS::Serverless::Function`	29. Januar 2021

Support für taumelnde Fenster	Unterstützung für Tumbling-Fenster für DynamoDB und Kinesis-Ereignisquellen für serverlose Funktione n hinzugefügt. Weitere Informationen finden Sie unter den `TumblingW indowInSeconds` Eigenschaften `Kinesis`und den `DynamoDB`Datentypen des Ressourcentyps. `AWS::Serv erless::Function`	17. Dezember 2020
Support für warme Behälter	Unterstützung für warme Container beim lokalen Testen mit den AWS SAMCLI Befehlen `sam local start-api` und hinzugefü gt `sam local start-lam bda` . Weitere Informationen finden Sie in der `--warm-co ntainers` Option für diese Befehle.	16. Dezember 2020
Support für Lambda-Co ntainer-Images	Unterstützung für Lambda-Container-Images hinzugefü gt. Weitere Informationen finden Sie unter Anwendungen erstellen.	1. Dezember 2020

Support für Codesignatur	Unterstützung für Codesignatur und vertrauenswürdige Bereitstellungen von serverlosem Anwendungscode hinzugefügt. Weitere Informationen finden Sie unter Codesignatur für AWS SAM Anwendungen konfigurieren.	23. November 2020
Support für parallel und zwischengespeicherte Builds	Die Leistung serverloser Anwendungsbuilds wurde verbessert, indem dem `sam build`Befehl zwei Optionen hinzugefügt wurden:`--parallel` , der Funktionen und Ebenen parallel statt sequentiell erstellt, und`--cached`, der Build-Artefakte aus früheren Builds verwendet, wenn keine Änderungen vorgenommen wurden, die eine Neuerstellung erfordern.	10. November 2020

Support für Amazon MQ und gegenseitige TLS-Authentifizierung	Unterstützung für Amazon MQ als Ereignisquelle für serverlose Funktionen hinzugefügt. Weitere Informationen finden Sie unter den `MQ`Datentypen `EventSource` und des `AWS::Serverless::Function` Ressourcentyps. Außerdem wurde Unterstützung für die gegenseitige Transport Layer Security (TLS) -Authentifizierung für API-Gateway-APIs und HTTP-APIs hinzugefügt. Weitere Informationen finden Sie unter dem `DomainConfiguration` Datentyp des `AWS::Serverless::Api` Ressourcentyps oder dem `HttpApiDomainConfiguration` Datentyp des `AWS::Serverless::HttpApi` Ressourcentyps.	5. November 2020
Support für Lambda-Autorisierer für HTTP-APIs	Unterstützung für Lambda-Autorisierer für den `AWS::Serverless::HttpApi` Ressourcentyp hinzugefügt. Weitere Informationen finden Sie unter Beispiel für einen Lambda-Authorizer () `AWS::Serverless::HttpApi` .	27. Oktober 2020

Support für mehrere Konfigurationsdateien und Umgebungen	Unterstützung für mehrere Konfigurationsdateien und Umgebungen zum Speichern von Standardparameterwerten für AWS SAMCLI Befehle hinzugefügt. Weitere Informationen finden Sie in der AWS SAMCLIKonfigurationsdatei.	24. September 2020
Support für X-Ray With-Step-Funktionen und Referenzen bei der Steuerung des Zugriffs auf APIs	Unterstützung für X-Ray als Ereignisquelle für serverlose Zustandsmaschinen hinzugefügt. Weitere Informationen finden Sie unter der `Tracing`Eigenschaft des `AWS::Serverless::StateMachine` Ressourcentyps. Außerdem wurde Unterstützung für Verweise bei der Steuerung des Zugriffs auf APIs hinzugefügt. Weitere Informationen finden Sie unter `ResourcePolicyStatement` Datentyp.	17. September 2020

Support für Amazon MSK	Unterstützung für Amazon MSK als Ereignisquelle für serverlose Funktionen hinzugefügt. Dadurch können Datensätze in einem Amazon MSK-Thema Ihre Lambda-Funktion auslösen. Weitere Informationen finden Sie unter den Datentypen `EventSource` und `MSK`-Datentypen des `AWS::Serverless::Function` Ressourcentyps.	13. August 2020
Support für Amazon EFS	Unterstützung für das Mounten von Amazon EFS-Dateisystemen in lokalen Verzeichnissen hinzugefügt. Dadurch kann Ihr Lambda-Funktionscode auf gemeinsam genutzte Ressourcen zugreifen und diese ändern. Weitere Informationen finden Sie in der `FileSystemConfigs` Eigenschaft des `AWS::Serverless::Function` Ressourcentyps.	16. Juni 2020

Orchestrierung serverloser Anwendungen	Unterstützung für die Orchestrierung von Anwendungen hinzugefügt, indem Step Functions Functions-Zustandsmaschinen mithilfe AWS SAM von erstellt werden. Weitere Informationen finden Sie unter AWS Ressourcen orchestrieren mit AWS Step Functions und unter `AWS::Serverless::StateMachine` Ressourcentyp.	27. Mai 2020
Benutzerdefinierte Laufzeiten erstellen	Es wurde die Möglichkeit hinzugefügt, benutzerdefinierte Laufzeiten zu erstellen. Weitere Informationen finden Sie unter Benutzerdefinierte Laufzeiten erstellen.	21. Mai 2020
Ebenen erstellen	Es wurde die Möglichkeit hinzugefügt, einzelne `LayerVersion` Ressourcen zu erstellen. Weitere Informationen finden Sie unter Ebenen erstellen.	19. Mai 2020
Generierte CloudFormation Ressourcen	Es wurden Einzelheiten zu den AWS SAM generierten CloudFormation Ressourcen und zu den Referenzierungen bereitgestellt. Weitere Informationen finden Sie unter Generierte CloudFormation Ressourcen.	8. April 2020

AWS Anmeldeinformationen einrichten	Es wurden Anweisungen zum Einrichten von AWS Anmeldeinformationen hinzugefügt, falls Sie sie noch nicht für die Verwendung mit anderen AWS Tools wie einem der AWS SDKs oder dem AWS CLI eingerichtet haben. Weitere Informationen finden Sie unter AWS Anmeldeinformationen einrichten.	17. Januar 2020
AWS SAM Spezifikation und AWS SAMCLI Aktualisierungen	Die AWS SAM Spezifikation von GitHub wurde migriert. Weitere Informationen finden Sie in der AWS SAM Spezifikation. Außerdem wurde der Bereitstellungs-Workflow mit Änderungen am `sam deploy`Befehl aktualisiert.	25. November 2019
Neue Optionen zur Steuerung des Zugriffs auf API Gateway Gateway-APIs und Aktualisierungen von Richtlinienvorlagen	Neue Optionen zur Steuerung des Zugriffs auf API Gateway Gateway-APIs hinzugefügt: IAM-Berechtigungen, API-Schlüssel und Ressourcenrichtlinien. Weitere Informationen finden Sie unter Steuern des Zugriffs auf API-Gateway-APIs. Außerdem wurden zwei Richtlinienvorlagen aktualisiert: RekognitionFacesPolicy und ElasticsearchHttpPostPolicy. Weitere Informationen finden Sie unter AWS SAM Richtlinienvorlagen.	29. August 2019

Updates für den Einstieg	Das Kapitel „Erste Schritte“ wurde mit verbesserten Installationsanweisungen für das AWS SAMCLI und das Hello World-Tutorial aktualisiert. Weitere Informationen finden Sie unter Erste Schritte mit AWS SAM.	25. Juli 2019
Steuern des Zugriffs auf API Gateway Gateway-APIs	Unterstützung für die Steuerung des Zugriffs auf API Gateway Gateway-APIs hinzugefügt. Weitere Informationen finden Sie unter Steuern des Zugriffs auf API-Gateway-APIs.	21. März 2019
Hinzugefügt `sam publish` zum AWS SAMCLI	Der neue `sam publish`Befehl in der AWS SAMCLI vereinfacht das Veröffentlichen von serverlosen Anwendungen in der AWS Serverless Application Repository. Weitere Informationen finden Sie unter Veröffentlichen serverloser Anwendungen mit dem AWS SAMCLI.	21. Dezember 2018
Unterstützung verschachtelter Anwendungen und Ebenen	Unterstützung für verschachtelte Anwendungen und Ebenen hinzugefügt. Weitere Informationen finden Sie unter Verwenden verschachtelter Anwendungen und Arbeiten mit Ebenen.	29. November 2018

Hinzugefügt `sam build` zum AWS SAMCLI	Der neue `sam build`Befehl im AWS SAMCLI vereinfacht das Kompilieren serverloser Anwendungen mit Abhängigk eiten, sodass Sie diese Anwendungen lokal testen und bereitstellen können. Weitere Informationen finden Sie unter Anwendungen erstellen.	19. November 2018
Es wurden neue Installat ionsoptionen für hinzugefügt AWS SAMCLI	Es wurden die Installat ionsoptionen Linuxbrew (Linux), MSI (Windows) und Homebrew (macOS) für hinzugefügt. AWS SAMCLI Weitere Informationen finden Sie unter Installation von. AWS SAMCLI	7. November 2018
Neues Handbuch	Dies ist die erste Version des AWS Serverless Application Model -Entwicklerhandbuchs.	17. Oktober 2018

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