Guide du développeur

AWS Serverless Application Model

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

AWS Serverless Application Model: Guide du développeur

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

Les marques et la présentation commerciale d’Amazon ne peuvent être utilisées en relation avec un produit ou un service qui n’est pas d’Amazon, d’une manière susceptible de créer une confusion parmi les clients, ou d’une manière qui dénigre ou discrédite Amazon. Toutes les autres marques commerciales qui ne sont pas la propriété d’Amazon appartiennent à leurs propriétaires respectifs, qui peuvent ou non être affiliés ou connectés à Amazon, ou sponsorisés par Amazon.

Table of Contents

Qu'est-ce que c'est AWS SAM ? ... 1
Quand utiliser AWS SAM ... 1
Fonctions principales ... 2
Informations connexes ... 3
Comment ça marche ... 3
Quelle est la spécification du AWS SAM modèle ? ... 4
Quels sont le AWS SAM projet et le AWS SAM modèle ? ... 4
Qu'est-ce que c'est AWS SAMCLI ? ... 11
En savoir plus ... 19
Étapes suivantes ... 19
Concepts sans serveur ... 19
Concepts sans serveur ... 20
Infrastructure en tant que code ... 21
Outils IaC pour les AWS applications sans serveur ... 21
Comparaison des outils IaC pour les applications sans serveur ... 22
En savoir plus ... 23
Prise en main ... 24
Conditions préalables ... 24
Étape 1 : Ouvrez un AWS compte ... 25
Étape 2 : créer un compte utilisateur IAM ... 25
Étape 3 : créer un ID de clé d'accès et clé d'accès secrète ... 26
Étape 4 : installez le AWS CLI ... 29
Étape 5 : utilisez le AWS CLI pour configurer les AWS informations d'identification ... 30
Étape 6 : (Facultatif) Installer le AWS Toolkit for VS Code ... 31
Étapes suivantes ... 31
Installer la CLI AWS SAM ... 32
Installation de la CLI AWS SAM ... 33
Résolution des erreurs d'installation ... 43
Étapes suivantes ... 45
Facultatif : vérifiez le AWS SAMCLI programme d'installation ... 46
Didacticiel Hello World ... 59
Conditions préalables ... 60
Étape 1 : initialisation de l'exemple d'application Hello World ... 61
Étape 2 : créer votre application ... 65

Étape 3 : Déployez votre application sur AWS Cloud ... 67
Étape 4 : exécuter votre application ... 72
Étape 5 : Interagissez avec votre fonction dans AWS Cloud ... 73
Étape 6 : Modifiez et synchronisez votre application avec AWS Cloud ... 74
Étape 7 : (Facultatif) testez votre application localement ... 77
Étape 8 : Supprimez votre application du AWS Cloud ... 80
Résolution des problèmes ... 80
En savoir plus ... 80
Convertir depuis Lambda ... 81
En savoir plus ... 81
Comment utiliser AWS SAM ... 83
AWS SAM CLI ... 84
Comment les commandes de la CLI AWS SAM sont documentées ... 84
Configuration de la CLI AWS SAM ... 86
Comment créer votre fichier de configuration (le `samconfig` fichier) ... 87
Configuration des paramètres de projet ... 87
Configuration des informations d'identification et des paramètres de base ... 92
Commandes principales ... 92
Tests locaux avec AWS SAMCLI ... 94
Testez AWS SAM les applications localement ... 94
Testez CloudFormation les modèles localement ... 94
Testez les applications AWS CDK localement ... 94
Testez les applications Terraform localement ... 95
AWS SAM modèle ... 96
Anatomie du modèle ... 97
YAML ... 98
Sections de modèle ... 98
Étapes suivantes ... 101
Globals ... 101
Ressources et propriétés ... 107
AWS::Serverless::Api ... 107
AWS::Serverless::Application ... 167
AWS::Serverless::CapacityProvider ... 172
AWS::Serverless::Connector ... 182
AWS::Serverless::Function ... 196
AWS::Serverless::GraphQLApi ... 356

AWS::Serverless::HttpApi ... 392
AWS::Serverless::LayerVersion ... 425
AWS::Serverless::SimpleTable ... 431
AWS::Serverless::StateMachine ... 437
Ressources générées ... 479
Référencement des ressources générées CloudFormation ... 479
Scénarios CloudFormation de ressources générés ... 480
AWS::Serverless::Api ... 483
AWS::Serverless::Application ... 485
AWS::Serverless::CapacityProvider ... 485
AWS::Serverless::Connector ... 486
AWS::Serverless::Function ... 487
AWS::Serverless::GraphQLApi ... 493
AWS::Serverless::HttpApi ... 495
AWS::Serverless::LayerVersion ... 497
AWS::Serverless::SimpleTable ... 497
AWS::Serverless::StateMachine ... 497
Attributs de ressources supportés ... 499
Exceptions ... 500
extensions API Gateway ... 501
Fonctions intrinsèques ... 502
Exemples d'applications ... 504
Traiter les événements DynamoDB ... 504
Avant de commencer ... 504
Étape 1 : initialiser l'application ... 504
Étape 2 : tester l'application localement ... 505
Étape 3 : créer le package de l'application ... 505
Étape 4 : déployer l'application ... 506
Étapes suivantes ... 507
Traiter les événements Amazon S3 ... 507
Avant de commencer ... 508
Étape 1 : initialiser l'application ... 508
Étape 2 : empaqueter l'application ... 508
Étape 3 : déployer l'application ... 509
Étape 4 : tester l'application localement ... 510
Étapes suivantes ... 510

Développez votre application ... 512
Créez votre application ... 512
Initialiser une nouvelle application sans serveur ... 513
Options pour sam init ... 519
Résolution des problèmes ... 520
Exemples ... 520
En savoir plus ... 520
Étapes suivantes ... 521
Définissez votre infrastructure ... 521
Définition des ressources de l'application ... 521
Configuration de l’accès ... 523
Contrôlez l'accès à l'API ... 603
Améliorez l'efficacité grâce aux couches ... 617
Code de réutilisation ... 620
Gérez les événements temporels ... 624
Orchestration d'applications ... 627
Configurer la signature de code ... 629
Valider les fichiers AWS SAM modèles ... 632
Créez votre application ... 632
Présentation de sam build ... 633
Version par défaut ... 648
Personnalisez votre build ... 656
Testez votre application ... 687
Présentation de sam local ... 687
Utilisation de la commande sam local ... 688
Présentation de sam local generate-event ... 689
Présentation de sam local invoke ... 695
Présentation de sam local start-api ... 701
Présentation de sam local start-lambda ... 708
Invoquer des fonctions localement ... 711
Fichier de variable d'environnement ... 711
Layers ... 713
En savoir plus ... 713
API Gateway exécuté localement ... 713
Fichier de variable d'environnement ... 715
Layers ... 716

Testez avec sam remote test-event ... 717
Configurer l'CLI AWS SAM pour utiliser sam remote test-event ... 718
Utilisation de la commande sam remote test-event ... 718
Utilisation des événements de test partageables ... 721
Gestion des événements de test partageables ... 721
Testez avec sam remote invoke ... 723
Utilisation de la même commande d'invocation à distance ... 724
Utilisation des options de commande sam remote invoke ... 728
Configurez le fichier de configuration de votre projet ... 734
Exemples ... 734
Liens connexes ... 750
Automatisez les tests d'intégration ... 750
Générez des exemples de charges utiles ... 752
Testez et déboguez des fonctions durables ... 753
Flux de travail de test local ... 753
Scénarios de test courants ... 754
Résolution des problèmes ... 754
Conseils de débogage ... 755
En savoir plus ... 755
Déboguez votre application ... 757
Fonctions de débogage locales ... 757
Utilisation de AWS boîtes à outils ... 758
Exécution AWS SAM locale en mode debug ... 760
Transmettre plusieurs arguments d'exécution ... 761
Valider avec cfn-lint ... 761
Exemples ... 762
En savoir plus ... 762
Déployez votre application et vos ressources ... 763
Présentation de sam deploy ... 763
Conditions préalables ... 764
Déploiement d'applications avec sam deploy ... 764
Bonnes pratiques ... 774
Options pour sam deploy ... 775
Résolution des problèmes ... 775
Exemples ... 775
En savoir plus ... 784

Options de déploiement ... 784
Comment utiliser le AWS SAMCLI pour déployer manuellement ... 784
Déployez avec CI/CD des systèmes et des pipelines ... 785
Déploiements graduels ... 785
Résolution de problèmes de déploiement à l'aide de la CLI AWS SAM ... 785
En savoir plus ... 713
Déployez avec CI/CD des systèmes et des pipelines ... 786
Qu'est-ce qu'un pipeline ? ... 787
Comment AWS SAM télécharge les fichiers locaux ... 788
Générer un pipeline de démarrage ... 796
Personnalisez les pipelines de démarrage ... 802
Automatisez vos déploiements ... 804
Utiliser l'authentification OIDC ... 809
Présentation de sam sync ... 812
Détectez et synchronisez automatiquement les modifications locales apportées au AWS Cloud ... 813
Personnalisez les modifications locales qui sont synchronisées avec AWS Cloud ... 814
Préparer votre application dans le cloud pour les tests et la validation ... 815
Options pour la commande sam sync ... 815
Résolution des problèmes ... 817
Exemples ... 818
En savoir plus ... 824
Surveillez votre application ... 825
Application Insights ... 825
Configuration d' CloudWatch Application Insights avec AWS SAM ... 825
Étapes suivantes ... 829
Utilisation des journaux ... 829
Récupération des journaux par pile CloudFormation ... 830
Extraction des journaux par nom de fonction Lambda ... 830
Journaux détaillés ... 830
Affichage des journaux pour une plage de temps spécifique ... 830
Filtrage des journaux ... 830
Mise en surbrillance des erreurs ... 831
Impression JSON « pretty printing » ... 831
AWS SAM référence ... 832
AWS SAM spécification et AWS SAM modèle ... 832

Référence des commandes CLI AWS SAM 832
Modèles de politique AWS SAM 833
Rubriques 833
AWS SAMCLIcommandes 834
sam build 835
sam delete 840
sam deploy 842
sam init 849
sam list 852
sam local callback 861
sam local execution 867
sam local generate-event 873
sam local invoke 874
sam local start-api 881
sam local start-lambda 886
sam logs 891
sam package 895
sam pipeline bootstrap 899
sam pipeline init 904
sam publish 905
sam remote callback 907
sam remote execution 913
sam remote invoke 920
sam remote test-event 925
sam sync 933
sam traces 940
sam validate 942
AWS SAMCLIgestion 943
Fichier de configuration CLI AWS SAM 944
Gestion des versions CLI AWS SAM 951
Configuration des AWS informations d'identification 961
Télémétrie CLI AWS SAM 962
Résolution des problèmes 965
Référence de connecteur 971
Types de ressources pris en charge 971
Politiques IAM créées par les connecteurs 982

Installation de Docker 1005
Installation deDocker 1005
Étapes suivantes 1009
Installation deFinch 1009
Installation deFinch 1009
Configuration à l'Finchaide de la AWS SAM CLI 1011
Vérification de l'installation 1013
Arrêt d'Finch 1014
Résolution des problèmes 1014
Étapes suivantes 1015
Référentiels d'images 1015
Référentiel d'images URIs 1016
Exemples 1018
Déploiement progressif 1018
Déploiement progressif d'une fonction Lambda pour la première fois 1021
En savoir plus 1023
Remarques importantes 1023
2023 1023
Prise en charge dans l'Terraform 1024
Prise en main 1024
Conditions préalables 1024
Utilisation des commandes de la CLI AWS SAM avec Terraform 1025
Configuration pour les projets Terraform 1026
Configuration pour Terraform Cloud 1031
Utilisation de la CLI AWS SAM avec Terraform 1033
Tests locaux avec sam local invoke 1033
Tests locaux avec sam local start-api 1034
Tests locaux avec sam local start-lambda 1035
Restrictions liées à Terraform 1036
Utilisation de la CLI AWS SAM avec Serverless.tf 1036
Référence Terraform 1037
AWS SAM référence des fonctionnalités prises en charge 1037
Référence spécifique à Terraform 1037
métadonnées sam 1037
Prise en charge de Terraform par la CLI AWS SAM 1041
Qu'est-ce que c'est AWS SAMCLI ? 1041

Comment utiliser la CLI AWS SAM avec Terraform ? 1042
Étapes suivantes 1042
Publier pour que d'autres puissent l'utiliser 1043
Conditions préalables 1043
Publication d'une nouvelle application 1045
Étape 1 : ajouter une `Metadata` section au AWS SAM modèle 1045
Étape 2 : empaqueter l'application 1045
Étape 3 : publier l'application 1046
Étape 4 : partager l'application (facultatif) 1046
Publication d'une nouvelle version d'une application existante 1047
Rubriques supplémentaires 1047
Propriétés de la section Métadonnées 1047
Propriétés 1047
Cas d’utilisation 1050
Exemple 1051
Historique de la documentation 1053
..... mlxxxi

Qu'est-ce que le AWS Serverless Application Model (AWS SAM) ?

AWS Serverless Application Model (AWS SAM) est un framework open source permettant de créer des applications sans serveur en utilisant l'infrastructure en tant que code (IAc). Avec AWS SAM sa syntaxe abrégée, les développeurs déclarent les CloudFormationressources et les ressources sans serveur spécialisées qui sont transformées en infrastructure lors du déploiement. Lorsque vous travaillerez avec AWS SAM, vous interagirez avec :

1. AWS SAM CLI- Un outil de ligne de commande qui vous permet de développer, de tester localement et de déployer vos applications sans serveur.
2. AWS SAM Modèle : extension de CloudFormation ce modèle qui fournit une syntaxe simplifiée pour définir les ressources sans serveur.

Lorsque vous utilisez la sam init commande, elle crée un répertoire de projet, que nous appellerons le AWS SAM projet, qui inclut généralement votre AWS SAM modèle, le code de votre application et d'autres fichiers de configuration.

Quand utiliser AWS SAM

AWS SAM est une solution IaC idéale pour les scénarios dans lesquels vous souhaitez un développement sans serveur simplifié avec toute la puissance de CloudFormation. Par exemple, vous pouvez utiliser SAM pour :

- Applications sans serveur : vous pouvez utiliser SAM pour définir rapidement des AWS Lambda fonctions, des fonctions Lambda durables, Amazon API APIs Gateway, des tables Amazon DynamoDB et d'autres ressources sans serveur avec un minimum de code.
- CloudFormation amélioration : vous pouvez combiner SAM avec des CloudFormation modèles existants pour ajouter des composants sans serveur à l'infrastructure traditionnelle. Les ressources SAM fonctionnent aux côtés CloudFormation des ressources standard dans le même modèle.
- Développement et tests locaux : vous pouvez utiliser le SAM CLI pour tester les fonctions Lambda localement, simuler les points de terminaison API Gateway et déboguer des applications sans serveur sur votre machine de développement avant de les déployer sur. AWS

- CI/CD pour le mode sans serveur : vous pouvez créer des pipelines de déploiement à l'aide de modèles SAM qui génèrent automatiquement l' CloudFormation infrastructure nécessaire aux environnements de préparation et de production.
- Migration à partir de ressources créées par la console : vous pouvez convertir les fonctions Lambda et les ressources API Gateway créées dans la console de AWS gestion en infrastructure sous forme de code à l'aide de modèles SAM.

Comparaison AWS SAM avec d'autres outils IaC

- Utilisez SAM plutôt que CloudFormation pour simplifier les définitions des ressources sans serveur tout en préservant la compatibilité des modèles.
- Utilisez SAM plutôt que AWS CDK si vous préférez une approche déclarative pour décrire votre infrastructure plutôt qu'une approche programmatique.
- Combinez SAM avec AWS CDK les fonctionnalités de test locales de SAM CLI pour améliorer vos applications CDK.

Fonctions principales

AWS SAM offre de nombreux avantages qui améliorent l'expérience des développeurs en vous permettant de :

Définir rapidement le code de votre infrastructure d'applications en utilisant moins de code

Créez des AWS SAM modèles pour définir le code de votre infrastructure d'applications sans serveur. Déployez vos modèles directement CloudFormation pour provisionner vos ressources.

Gérer vos applications sans serveur tout au long de leur cycle de développement

Utilisez la CLI AWS SAM pour gérer votre application sans serveur tout au long des phases de création, de construction, de déploiement, de test et de surveillance de votre cycle de vie de développement. Pour de plus amples informations, veuillez consulter AWS SAM CLI.

Attribuez rapidement des autorisations entre les ressources à l'aide de AWS SAM connecteurs

Utilisez des AWS SAM connecteurs dans vos AWS SAM modèles pour définir les autorisations entre vos AWS ressources. AWS SAM transforme votre code en autorisations IAM requises pour faciliter votre intention. Pour de plus amples informations, veuillez consulter Gestion des autorisations relatives aux ressources à l'aide de AWS SAM connecteurs.

Synchroniser en continu les modifications locales dans le cloud au fur et à mesure de votre développement

Utilisez la AWS SAMCLI sam sync commande pour synchroniser automatiquement les modifications locales dans le cloud, accélérant ainsi vos flux de travail de développement et de test dans le cloud. Pour de plus amples informations, veuillez consulter Présentation de l'utilisation sam sync de la synchronisation avec AWS Cloud.

Gérer vos applications Terraform sans serveur

Utilisez la CLI AWS SAM pour effectuer un débogage et des tests locaux de vos fonctions et couches Lambda. Pour de plus amples informations, veuillez consulter Prise en charge de Terraform par la CLI AWS SAM.

Informations connexes

- Pour plus d'informations sur le AWS SAM fonctionnement, voirComment AWS SAM fonctionne.
- Pour commencer à utiliser AWS SAM, voirCommencer avec AWS SAM.
- Pour une vue d'ensemble de la manière dont vous pouvez AWS SAM créer une application sans serveur, consultezComment utiliser AWS SAM.

Comment AWS SAM fonctionne

Lorsque vous créez votre application sans serveur, vous allez interagir avec les composants suivants : AWS SAM

1. AWS SAM modèle— Un fichier important qui définit vos AWS ressources. Ce modèle inclut la spécification du AWS SAM modèle, le framework open source fourni avec une syntaxe abrégée simplifiée que vous utilisez pour définir les fonctions, les événements APIs, les configurations et les autorisations de votre application sans serveur. Ce fichier se trouve dans le AWS SAM projet, qui est le dossier de l'application créé lorsque vous exécutez la sam init commande.
2. AWS SAM CLI— Un outil de ligne de commande que vous pouvez utiliser avec votre AWS SAM projet et les intégrations tierces prises en charge pour créer et exécuter vos applications sans serveur. AWS SAMCLIl s'agit de l'outil que vous utilisez pour exécuter des commandes sur votre AWS SAM projet et éventuellement le transformer en application sans serveur.

Pour exprimer les ressources, les mappages de sources d'événements et les autres propriétés qui définissent votre application sans serveur, vous définissez les ressources et développez votre application dans le AWS SAM modèle et dans d'autres fichiers de votre AWS SAM projet. Vous utilisez le AWS SAMCLI pour exécuter des commandes sur votre AWS SAM projet, c'est-à-dire pour initialiser, créer, tester et déployer votre application sans serveur.

> Vous débutez dans le monde du sans serveur ?
>
> Nous vous recommandons de passer en revueConcepts sans serveur pour AWS Serverless Application Model.

Quelle est la spécification du AWS SAM modèle ?

La spécification du AWS SAM modèle est un framework open source que vous pouvez utiliser pour définir et gérer le code de votre infrastructure d'applications sans serveur. La spécification du AWS SAM modèle est la suivante :

- Construit sur AWS CloudFormation : vous utilisez la CloudFormation syntaxe directement dans votre AWS SAM modèle, en tirant parti de sa prise en charge étendue des configurations de ressources et de propriétés. Si vous le connaissez déjà CloudFormation, vous n'avez pas besoin de vous familiariser avec un nouveau service pour gérer le code de votre infrastructure d'applications.
- Une extension de CloudFormation — AWS SAM propose sa propre syntaxe unique qui vise spécifiquement à accélérer le développement sans serveur. Vous pouvez utiliser à la fois la AWS SAM syntaxe CloudFormation et dans le même modèle.
- Une syntaxe abstraite et abrégée : en utilisant la syntaxe AWS SAM , vous pouvez définir votre infrastructure rapidement, en moins de lignes de code et avec moins de risques d'erreurs. Sa syntaxe est spécialement conçue pour faire abstraction de la complexité de la définition de votre infrastructure d'application sans serveur.
- Transformationnel : AWS SAM effectue le travail complexe de transformation de votre modèle en code nécessaire au provisionnement de votre infrastructure. CloudFormation

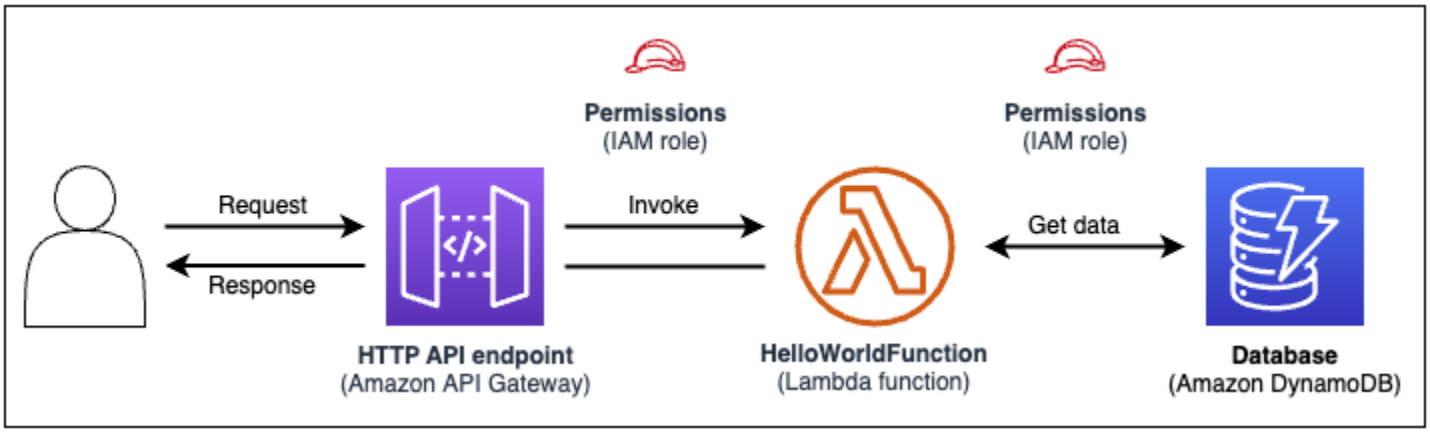
Quels sont le AWS SAM projet et le AWS SAM modèle ?

Le AWS SAM projet inclut le AWS SAM modèle qui contient la spécification du AWS SAM modèle. Cette spécification est le framework open source que vous utilisez pour définir votre infrastructure

d'applications sans serveur AWS, avec certains composants supplémentaires qui facilitent leur utilisation. En ce sens, les AWS SAM modèles sont une extension des CloudFormation modèles.

Voici un exemple d'application de base sans serveur. Cette application traite les demandes d'obtention de tous les éléments d'une base de données via une requête HTTP. Elle se compose des éléments suivants :

1. Une fonction qui contient la logique de traitement de la demande.
2. Une API HTTP pour servir de communication entre le client (demandeur) et l'application.
3. Une base de données pour stocker les articles.
4. Des autorisations permettant à l'application de s'exécuter en toute sécurité.



Le code d'infrastructure de cette application peut être défini dans le modèle AWS SAM suivant :

```
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
```

En 23 lignes de code, l'infrastructure suivante est définie :

- Fonction utilisant le AWS Lambda service.
- Une API HTTP utilisant le service Amazon API Gateway.
- Une base de données utilisant le service Amazon DynamoDB.
- Les autorisations Gestion des identités et des accès AWS (IAM) nécessaires pour que ces services interagissent les uns avec les autres.

Pour provisionner cette infrastructure, le modèle est déployé sur CloudFormation. Lors du déploiement, AWS SAM transforme les 23 lignes de code en la CloudFormation syntaxe requise pour générer ces ressources dans AWS. Le CloudFormation modèle transformé contient plus de 200 lignes de code !

CloudFormation Modèle transformé

```
{
  "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"
            }
          ]
        }
      }
    }
  }
```

En utilisant AWS SAM, vous définissez 23 lignes de code d'infrastructure. AWS SAM transforme votre code en plus de 200 lignes de CloudFormation code nécessaires au provisionnement de votre application.

Qu'est-ce que c'est AWS SAMCLI ?

AWS SAMCLIIl s'agit d'un outil de ligne de commande que vous pouvez utiliser avec des AWS SAM modèles et des intégrations tierces prises en charge pour créer et exécuter vos applications sans serveur. Utilisez la CLI AWS SAM pour :

- Initialiser rapidement un nouveau projet d'application.
- Créer votre application pour le déploiement.
- Effectuer le débogage et les tests au niveau local.
- Déployez votre application.
- Configurez les pipelines de CI/CD déploiement.
- Surveiller et dépanner votre application dans le cloud.
- Synchroniser les modifications locales dans le cloud au fur et à mesure de votre développement.
- Et bien plus encore !

AWS SAMCLIII est préférable de l'utiliser avec AWS SAM des CloudFormation modèles. Elle fonctionne également avec des produits tiers tels que Terraform.

Initialiser un nouveau projet

Sélectionnez des modèles de départ ou choisissez un emplacement de modèle personnalisé pour commencer un nouveau projet.

Ici, nous utilisons la commande sam init pour initialiser un nouveau projet d'application. Nous sélectionnons le projet Hello World Example pour commencer. La CLI AWS SAM télécharge un modèle de démarrage et crée la structure de répertoires de dossiers de notre projet.

```
→  what-is 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
        2 - Multi-step workflow
        3 - Serverless API
        4 - Scheduled task
        5 - Standalone function
        6 - Data processing
        7 - Infrastructure event management
        8 - Serverless Connector Hello World Example
        9 - Multi-step workflow with Connectors
        10 - Lambda EFS example
        11 - Machine Learning
Template: 1

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

Pour en savoir plus, consultez Créez votre application dans AWS SAM.

Créer votre application pour le déploiement

Regroupez les dépendances de vos fonctions et organisez le code de votre projet et la structure de dossiers pour préparer le déploiement.

Ici, nous utilisons la commande sam build pour préparer notre application en vue du déploiement. La CLI AWS SAM crée un répertoire `.aws-sam` et organise les dépendances et les fichiers de nos applications pour le déploiement.

```
➜  sam-app sam build
Building codeuri: /Users/evzz/Demo/what-is/sam-app/hello_world runtime: python3.9 metadata: {} archit
ecture: x86_64 functions: HelloWorldFunction
Running PythonPipBuilder:ResolveDependencies
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
➜  sam-app cd .aws-sam
➜  .aws-sam ls
build      build.toml
➜  .aws-sam
```

Pour en savoir plus, consultez Créez votre application.

Effectuer le débogage et les tests au niveau local

Sur votre machine locale, simulez des événements, testez APIs, invoquez des fonctions, etc. pour déboguer et tester votre application.

Ici, nous utilisons la commande sam local invoke pour appeler notre `HelloWorldFunction` localement. Pour ce faire, la CLI AWS SAM crée un conteneur local, crée notre fonction, l'invoque et affiche les résultats. Vous pouvez utiliser une application telle que Docker pour exécuter des conteneurs sur votre machine.

```
➜  sam-app sam local invoke HelloWorldFunction
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/evzz/Demo/what-is/sam-app/.aws-sam/build/HelloWorldFunction as /var/task:ro,delegated
 inside runtime container
START RequestId: 6f8347ce-6b04-4246-a0de-6dc37f0eef51 Version: $LATEST
END RequestId: 6f8347ce-6b04-4246-a0de-6dc37f0eef51
REPORT RequestId: 6f8347ce-6b04-4246-a0de-6dc37f0eef51  Init Duration: 1.23 ms  Duration: 639.26 ms B
illed Duration: 640 ms  Memory Size: 128 MB     Max Memory Used: 128 MB
{"statusCode": 200, "body": "{\"message\": \"hello world\"}"}
```

Pour de plus amples informations, veuillez consulter Testez votre application et Déboguez votre application.

Déploiement de votre application

Configurez les paramètres de déploiement de votre application et déployez-la AWS dans le cloud pour provisionner vos ressources.

Ici, nous utilisons la commande sam deploy --guided pour déployer notre application via un flux interactif. Il nous AWS SAMCLI guide dans la configuration des paramètres de déploiement de notre application, transforme notre modèle en CloudFormation et le déploie CloudFormation pour créer nos ressources.

```
➜  sam-app sam deploy --guided

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

        Looking for config file [samconfig.toml] :  Not found

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

        Looking for resources needed for deployment:
         Managed S3 bucket: aws-sam-cli-managed-default-samclisourcebucket-1a4x26zbcdkqr
         A different default S3 bucket can be set in samconfig.toml
```

Pour en savoir plus, consultez Déployez votre application et vos ressources.

Configuration des pipelines CI/CD de déploiement

Créez des pipelines sécurisés d'intégration et de livraison continues (CI/CD) à l'aide d'un système compatible CI/CD .

Ici, nous utilisons la commande sam pipeline init --bootstrap pour configurer un pipeline de déploiement CI/CD pour notre application. Il nous AWS SAMCLI guide à travers nos options et génère les AWS ressources et le fichier de configuration à utiliser avec notre système CI/CD.

```
[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]:
```

Pour en savoir plus, consultez Déployez avec CI/CD des systèmes et des pipelines.

Surveiller et dépanner votre application dans le cloud

Consultez les informations importantes sur vos ressources déployées, collectez les journaux et utilisez les outils de surveillance intégrés tels que AWS X-Ray.

Ici, nous utilisons la commande sam list pour visualiser nos ressources déployées. Nous obtenons le point de terminaison de notre API et l'appelons, ce qui déclenche notre fonction. Ensuite, nous l'utilisons sam logs pour consulter les journaux de nos fonctions.

```
➜  sam-app sam logs --stack-name sam-app
2023/03/13/[$LATEST]0a433e844dd445bd82d0d78cd55e0cdc 2023-03-13T21:06:42.075000 INIT_START Runtime Ve
rsion: python:3.9.v16   Runtime Version ARN: arn:aws:lambda:us-west-2::runtime:07a48df201798d627f2b95
0f03bb227aab4a655a1d019c3296406f95937e2525
2023/03/13/[$LATEST]0a433e844dd445bd82d0d78cd55e0cdc 2023-03-13T21:06:42.180000 START RequestId: 778e
4226-0a80-435f-929b-5b19292ed9a7 Version: $LATEST
2023/03/13/[$LATEST]0a433e844dd445bd82d0d78cd55e0cdc 2023-03-13T21:06:42.181000 END RequestId: 778e42
26-0a80-435f-929b-5b19292ed9a7
2023/03/13/[$LATEST]0a433e844dd445bd82d0d78cd55e0cdc 2023-03-13T21:06:42.182000 REPORT RequestId: 778
e4226-0a80-435f-929b-5b19292ed9a7       Duration: 1.69 ms       Billed Duration: 2 ms   Memory Size:
128 MB  Max Memory Used: 36 MB  Init Duration: 104.13 ms
```

Pour en savoir plus, consultez Surveillez votre application.

Synchroniser les modifications locales dans le cloud au fur et à mesure de votre développement

Au fur et à mesure que vous développez sur votre ordinateur local, synchronisez automatiquement les modifications dans le cloud. Visualisez rapidement vos modifications et effectuez des tests et des validations dans le cloud.

Ici, nous utilisons la commande sam sync --watch pour que la CLI AWS SAM surveille les modifications locales. Nous modifions notre code `HelloWorldFunction` et la CLI AWS SAM détecte automatiquement la modification et déploie nos mises à jour dans le cloud.

```
-----------------------------------------------------------------------------------------------------
Key                 HelloWorldFunctionIamRole
Description         Implicit IAM Role created for Hello World function
Value               arn:aws:iam::513423067560: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-2:513423067560: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.
Syncing Lambda Function HelloWorldFunction...
Manifest is not changed for (HelloWorldFunction), running incremental build
Building codeuri: /Users/evzz/Demo/what-is/sam-app/hello_world runtime: python3.9 metadata: {} archit
ecture: x86_64 functions: HelloWorldFunction
Running PythonPipBuilder:CopySource
Finished syncing Lambda Function HelloWorldFunction.
```

Testez les ressources prises en charge dans le cloud

Invoquez et transmettez des événements aux ressources prises en charge dans le cloud.

Ici, nous utilisons la commande sam remote invoke pour tester une fonction Lambda déployée dans le cloud. Nous appelons notre fonction Lambda et recevons ses journaux et sa réponse. Notre fonction Lambda étant configurée pour diffuser les réponses, la CLI AWS SAM diffuse sa réponse en temps réel.

```
→ lambda-streaming-app sam remote invoke StreamingFunction --stack-name lambda-streaming-app
```

En savoir plus

Pour en savoir plus AWS SAM, consultez les ressources suivantes :

- L' AWS SAM atelier complet — Un atelier conçu pour vous enseigner les nombreuses fonctionnalités principales qu'il AWS SAM fournit.
- Sessions avec SAM — Série de vidéos créée par notre équipe AWS Serverless Developer Advocate sur l'utilisation AWS SAM.
- Serverless Land : site qui rassemble les dernières informations, blogs, vidéos, codes et ressources d'apprentissage pour le AWS sans-serveur.

Étapes suivantes

Si c'est la première fois que vous l'utilisez AWS SAM, consultezCommencer avec AWS SAM.

Concepts sans serveur pour AWS Serverless Application Model

Découvrez les concepts de base du mode sans serveur avant d'utiliser le AWS Serverless Application Model (AWS SAM).

Concepts sans serveur

Architecture basée sur les événements

Une application sans serveur comprend des AWS services individuels, tels que AWS Lambda pour le calcul et Amazon DynamoDB pour la gestion des bases de données, qui jouent chacun un rôle spécialisé. Ces services sont ensuite librement intégrés les uns aux autres par le biais d'une architecture basée sur les événements. Pour en savoir plus sur l'architecture basée sur les événements, consultez Qu'est-ce qu'une architecture basée sur les événements ?.

Infrastructure en tant que code (IaC)

L'infrastructure en tant que code (IaC) permet de traiter l'infrastructure de la même manière que les développeurs traitent le code, en appliquant la même rigueur que le développement du code d'application à l'approvisionnement de l'infrastructure. Vous définissez votre infrastructure dans un fichier modèle, vous la déployez et vous AWS créez les ressources pour vous. AWS Avec IaC, vous définissez dans le code ce que vous AWS souhaitez provisionner. Pour une comparaison des outils IaC disponibles pour AWS, voirInfrastructure en tant que code (IaC).

Technologies sans serveur

Grâce aux technologies AWS sans serveur, vous pouvez créer et exécuter des applications sans avoir à gérer vos propres serveurs. L'ensemble de la gestion des serveurs est assuré par AWS ce qui offre de nombreux avantages tels que le dimensionnement automatique et la haute disponibilité intégrée, ce qui vous permet de mettre rapidement votre idée en production. Grâce aux technologies sans serveur, vous pouvez vous concentrer sur l'essentiel de votre produit sans avoir à vous soucier de la gestion et de l'exploitation des serveurs. Pour en savoir plus sur le sans serveur, consultez les informations suivantes :

- Sans serveur activé AWS
- Guide du développeur sans serveur : fournit une présentation conceptuelle du développement sans serveur dans le Cloud AWS .

Pour une introduction de base aux principaux services AWS sans serveur, voir Serverless 101 : Understanding the serverless services at Serverless Land.

Application sans serveur

Lorsque vous utilisez AWS SAM, vous gérez les ressources associées dans une application, qui comprend votre AWS SAM projet et votre modèle. Toutes les ressources de votre application sont définies ou référencées dans votre AWS SAM modèle. Lorsque vous AWS SAM traitez

votre modèle, il crée CloudFormation des ressources. Dans CloudFormation, les ressources sont gérées dans une seule unité appelée pile, et toutes les ressources d'une pile sont définies par le CloudFormation modèle de la pile.

Infrastructure en tant que code (IaC)

Avec l'infrastructure en tant que code (IaC), vous pouvez automatiser le déploiement et la gestion de vos AWS ressources, y compris les applications sans serveur. IaC vous permet de définir votre infrastructure à l'aide de code, ce qui facilite la création de versions, le partage et la réplication de vos déploiements. Cette approche vous permet de :

- Accélérez votre cycle de développement
- Simplifier la gestion des configurations
- Améliorez la fiabilité et la cohérence de vos déploiements

Outils IaC pour les AWS applications sans serveur

AWS propose plusieurs outils IaC pour vous aider à créer, déployer et gérer vos ressources cloud. Cette section explique comment AWS SAM s'intègre à cet écosystème et fonctionne avec d'autres outils AWS IaC.

AWS CloudFormation

À l'aide de modèles YAML ou JSON CloudFormation, vous pouvez modéliser et approvisionner l'ensemble de votre AWS infrastructure. CloudFormation gère automatiquement la création, la mise à jour et la suppression des ressources. Lorsque vous déployez AWS SAM des applications, CloudFormation traitez les modèles transformés pour créer et gérer vos ressources.

AWS Serverless Application Model (AWS SAM)

AWS SAM vous aide à créer des applications sans serveur avec une syntaxe simplifiée pour définir les ressources sans serveur. Vous pouvez utiliser des AWS SAM modèles pour configurer des fonctions Lambda APIs, des bases de données et des sources d'événements à l'aide d'une syntaxe YAML concise. AWS SAM transforme ces modèles en CloudFormation modèles lors du déploiement.

> **Note**
>
> Bien que vous soyez AWS SAM spécialisé dans les applications sans serveur, vous pouvez utiliser tous les types de CloudFormation ressources dans vos AWS SAM modèles. Cela vous donne la flexibilité d'inclure des ressources non sans serveur en cas de besoin.

AWS Kit de développement Cloud (AWS CDK)

Avec AWS CDK, vous pouvez définir votre infrastructure sans serveur à l'aide de langages de programmation courants tels que Python TypeScript, Java, C#/.Net ou Go. Vous pouvez utiliser des structures de programmation telles que des boucles et des conditions pour définir votre infrastructure et AWS CDK générer des CloudFormation modèles pour le déploiement. Vous pouvez utiliser le AWS SAMCLI pour tester et déboguer localement les applications créées avec AWS CDK. Pour en savoir plus, consultez Tester les applications CDK localement.

Comparaison des outils IaC pour les applications sans serveur

Lorsque vous choisissez un outil IaC pour vos applications sans serveur, tenez compte des préférences de votre équipe, des exigences du projet et des flux de travail existants. Le tableau suivant compare les principales caractéristiques des outils AWS IaC pour le développement sans serveur :

Outil	Utilisation principale	Idéal pour	Fonctionne avec AWS SAM	Quand choisir
CloudFormation	Gestion d'une AWS infrastructure complexe	Applications nécessitant un contrôle détaillé des AWS ressources	AWS SAM les modèles se transforment en CloudFormation modèles lors du déploiement	Pour un contrôle précis des ressources non sans serveur
AWS SAM	Développement d'applications sans serveur	Équipes développant des applications sans	Fonctionnalité native	Lorsque vous vous concentrez principal

Outil	Utilisation principale	Idéal pour	Fonctionne avec AWS SAM	Quand choisir
		serveur à l'aide de Lambda		ement sur des architectures sans serveur dotées de fonctions Lambda, d'API APIs Gateway et d'autres ressources sans serveur
AWS CDK	Définition d'infrastructures à l'aide de langages de programmation	Équipes privilégiant les langages dactylographiés et l'approche axée sur le code	Générer des AWS SAM modèles et les utiliser AWS SAMCLI pour les tests	Lorsque vous avez besoin d'une définition d'infrastructure programmatique ou d'une logique de configuration de ressources complexe

> **Note**
>
> Bien que ce guide se concentre sur les outils iAc AWS natifs, Terraform est une autre solution IaC populaire qui peut être utilisée pour définir des applications sans serveur. Il AWS SAMCLI prend en charge les tests locaux des fonctions Lambda définies dans Terraform. Pour de plus amples informations, veuillez consulter Prise en charge de Terraform par la CLI AWS SAM.

En savoir plus

- Pour en savoir plus sur AWS les DevOps pratiques en DevOps matière de AWS
- Pour plus d'informations sur l'utilisation de Lambda avec différents outils IaC, voir Utilisation de Lambda avec une infrastructure en tant que code (iAc)

Commencer avec AWS SAM

Commencez AWS SAM en passant en revue et en complétant les rubriques de cette section. AWS SAM prérequisfournit des instructions détaillées sur la configuration d'un AWS compte, la création d'utilisateurs IAM, la création d'un accès par clé, ainsi que sur l'installation et la configuration du AWS SAMCLI. Après avoir rempli les prérequis, vous pouvez choisir votre approche de développement :

- Installer la CLI AWS SAM, ce que vous pouvez faire sur les systèmes d'exploitation Linux, Windows et macOS.
- Démarrez avec AWS SAM à l'aide de VS Code Toolkit pour une expérience de développement visuelle avec des fonctionnalités de débogage et de test intégrées.

Une fois l'installation terminée, vous pouvez éventuellement suivre le didacticiel AWS SAM Hello World. Ce didacticiel vous expliquera le processus de création d'une application sans serveur de base avec AWS SAM. Après avoir terminé le didacticiel, vous serez prêt à passer en revue les concepts détaillés dansComment utiliser AWS Serverless Application Model (AWS SAM).

Rubriques

- AWS SAM prérequis
- Installer la CLI AWS SAM
- Tutoriel : Déployer une application Hello World avec AWS SAM
- Conversion de fonctions Lambda en applications AWS SAM

AWS SAM prérequis

Remplissez les conditions préalables suivantes avant d'installer et d'utiliser l'interface de ligne de AWS Serverless Application Model commande (AWS SAMCLI).

Pour utiliser le AWS SAMCLI, vous avez besoin des éléments suivants :

- Un AWS compte, des informations d'identification Gestion des identités et des accès AWS (IAM) et une paire de clés d'accès IAM.
- Le AWS Command Line Interface (AWS CLI) pour configurer les AWS informations d'identification.

Rubriques

- Étape 1 : Ouvrez un AWS compte
- Étape 2 : créer un compte utilisateur IAM
- Étape 3 : créer un ID de clé d'accès et clé d'accès secrète
- Étape 4 : installez le AWS CLI
- Étape 5 : utilisez le AWS CLI pour configurer les AWS informations d'identification
- Étape 6 : (Facultatif) Installer le AWS Toolkit for VS Code
- Étapes suivantes

Étape 1 : Ouvrez un AWS compte

Si vous n'en avez pas Compte AWS, procédez comme suit pour en créer un.

Pour vous inscrire à un Compte AWS

1. Ouvrez l'https://portal.aws.amazon.com/billing/inscription.
2. Suivez les instructions en ligne.

   Dans le cadre de la procédure d'inscription, vous recevrez un appel téléphonique ou un SMS et vous saisirez un code de vérification en utilisant le clavier numérique du téléphone.

   Lorsque vous vous inscrivez à un Compte AWS, un Utilisateur racine d'un compte AWSest créé. Par défaut, seul l'utilisateur racine a accès à l'ensemble des Services AWS et des ressources de ce compte. La meilleure pratique de sécurité consiste à attribuer un accès administratif à un utilisateur, et à utiliser uniquement l'utilisateur racine pour effectuer les tâches nécessitant un accès utilisateur racine.

Étape 2 : créer un compte utilisateur IAM

Afin de créer un utilisateur administrateur, choisissez l'une des options suivantes :

Choisissez un moyen de gérer votre administrateur	À	En	Vous pouvez également
Dans IAM Identity Center (Recommandé)	Utiliser des informations d'identification à court terme pour accéder à AWS. C'est conforme aux bonnes pratiques en matière de sécurité. Pour plus d'informations sur les bonnes pratiques, consultez Bonnes pratiques de sécurité dans IAM dans le Guide de l'utilisateur IAM.	Suivant les instructions fournies dans Mise en route dans le Guide de l'utilisateur AWS IAM Identity Center .	Configurez l'accès par programmation en configurant le AWS CLI à utiliser AWS IAM Identity Center dans le guide de l'AWS Command Line Interface utilisateur.
Dans IAM (Non recommandé)	Utiliser les informations d'identification à long terme pour accéder à AWS.	Suivant les instructions fournies dans Création d'un utilisateur IAM pour l'accès d'urgence dans le Guide de l'utilisateur IAM.	Configurer l'accès par programmation en suivant les instructions fournies dans Gestion des clés d'accès pour les utilisateurs IAM dans le Guide de l'utilisateur IAM.

Étape 3 : créer un ID de clé d'accès et clé d'accès secrète

Pour accéder à la CLI, vous avez besoin d'un ID de clé d'accès et d'une clé d'accès secrète. Utilisation des informations d'identification temporaires au lieu des clés d'accès à long terme si possible. Les informations d'identification temporaires incluent un ID de clé d'accès, une clé d'accès

secrète et un jeton de sécurité qui indique la date d'expiration des informations d'identification. Pour plus d'informations, consultez la section Utilisation d'informations d'identification temporaires avec AWS des ressources dans le Guide de l'utilisateur IAM.

Les utilisateurs ont besoin d'un accès programmatique s'ils souhaitent interagir avec AWS l'extérieur du AWS Management Console. La manière d'accorder un accès programmatique dépend du type d'utilisateur qui y accède AWS.

Pour accorder aux utilisateurs un accès programmatique, choisissez l'une des options suivantes.

Quel utilisateur a besoin d'un accès programmatique ?	À	Méthode
IAM	(Recommandé) Utilisez les informations d'identification de la console comme informations d'identification temporaires pour signer les demandes programmatiques adressées au AWS CLI AWS SDKs, ou AWS APIs.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour le AWS CLI, voir Connexion pour le développement AWS local dans le guide de AWS Command Line Interface l'utilisateur. • Pour AWS SDKs, voir Connexion pour le développement AWS local dans le guide de référence AWS SDKs and Tools.
Identité de la main-d'œuvre (Utilisateurs gérés dans IAM Identity Center)	Utilisez des informations d'identification temporaires pour signer les demandes programmatiques adressées au AWS CLI AWS SDKs, ou AWS APIs.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour le AWS CLI, voir Configuration du AWS CLI à utiliser AWS IAM Identity Center dans le

Quel utilisateur a besoin d'un accès programmatique ?	À	Méthode
		guide de AWS Command Line Interface l'utilisateur. • Pour AWS SDKs, outils, et AWS APIs, voir Authentification IAM Identity Center dans le guide de référence AWS SDKs et Tools.
IAM	Utilisez des informations d'identification temporaires pour signer les demandes programmatiques adressées au AWS CLI AWS SDKs, ou AWS APIs.	Suivez les instructions de la section Utilisation d'informations d'identification temporaires avec AWS les ressources du Guide de l'utilisateur IAM.

Quel utilisateur a besoin d'un accès programmatique ?	À	Méthode
IAM	(Non recommandé) Utilisez des informations d'identification à long terme pour signer des demandes programmatiques adressées au AWS CLI AWS SDKs, ou AWS APIs.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour le AWS CLI, voir Authentification à l'aide des informations d'identification utilisateur IAM dans le Guide de l'AWS Command Line Interface utilisateur. • Pour les outils AWS SDKs et, voir Authentifier à l'aide d'informations d'identif ication à long terme dans le guide de référence des outils AWS SDKs et. • Pour AWS APIs, voir Gestion des clés d'accès pour les utilisateurs IAM dans le Guide de l'utilisateur IAM.

Étape 4 : installez le AWS CLI

AWS CLI Il s'agit d'un outil open source qui vous permet d'interagir à Services AWS l'aide de commandes dans votre shell de ligne de commande. AWS SAMCLICela nécessite AWS CLI des activités telles que la configuration des informations d'identification. Pour en savoir plus sur le AWS CLI, voir Qu'est-ce que le AWS Command Line Interface ? dans le guide de AWS Command Line Interface l'utilisateur.

Pour l'installer AWS CLI, reportez-vous à la section Installation ou mise à jour de la AWS CLI dernière version du Guide de AWS Command Line Interface l'utilisateur.

Étape 5 : utilisez le AWS CLI pour configurer les AWS informations d'identification

Pour configurer les informations d'identification avec IAM Identity Center

- Pour configurer les informations d'identification avec IAM Identity Center, voir Configurer votre profil à l'aide de l'assistant de AWS configuration SSO.

Pour configurer les informations d'identification à l'aide du AWS CLI

1. Exécutez la commande `aws configure` depuis la ligne de commande.
2. Configurez ce qui suit. Cliquez sur chaque lien pour en savoir plus :

   a. ID de clé d'accès

   b. Clé d'accès secrète

   c. Région AWS

   d. Format de sortie

   L'exemple suivant montre des exemples de valeurs.

```
$ 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
```

AWS CLI Stocke ces informations dans un profil (un ensemble de paramètres) nommé `default` dans les `config` fichiers `credentials` et. Ces fichiers se trouvent dans `.aws` de votre répertoire de base. Par défaut, les informations de ce profil sont utilisées lorsque vous exécutez une AWS CLI commande qui ne spécifie pas explicitement le profil à utiliser. Pour plus d'informations sur le fichier `credentials`, consultez Paramètres des fichiers de configuration et d'informations d'identification dans le Guide de l'utilisateur AWS Command Line Interface .

Pour plus d'informations sur la configuration des informations d'identification, comme l'utilisation d'un fichier de configuration et d'informations d'identification existant, consultez Configuration rapide dans le Guide de l'utilisateur AWS Command Line Interface .

Étape 6 : (Facultatif) Installer le AWS Toolkit for VS Code

Pour les développeurs qui préfèrent un environnement de développement intégré, le AWS Toolkit for VS Code fournit de puissantes fonctionnalités pour le développement sans serveur, notamment le débogage visuel, CodeLens l'intégration et des flux de travail de déploiement rationalisés.

Prérequis pour le développement de VS Code

- Visual Studio Code (version 1.73.0 ou version ultérieure) installé sur votre système
- Extension de support du langage YAML pour VS Code

Pour installer le AWS Toolkit for VS Code

1. Ouvrez Visual Studio Code
2. Ouvrez la vue Extensions (Ctrl+Shift+X ou Cmd+Shift+X)
3. Recherchez « AWS Toolkit »
4. Installez l'extension « AWS Toolkit » d'Amazon Web Services
5. Installez l'extension « YAML » de Red Hat (requise pour les CodeLens fonctionnalités du modèle SAM)

Avantages de l'utilisation de VS Code avec AWS SAM

- Débogage visuel : définissez des points d'arrêt et parcourez vos fonctions Lambda localement
- CodeLens intégration : créez, déployez et invoquez des fonctions directement à partir de votre modèle SAM
- Terminal intégré : accédez aux AWS CLI commandes AWS SAM sans quitter votre éditeur
- Validation des modèles : validation en temps réel et IntelliSense pour les modèles SAM

Pour plus d'informations sur la configuration de vos AWS informations d'identification dans VS Code, consultez la section Configuration des informations d'identification dans le guide de l'utilisateur du AWS Toolkit for VS Code.

Étapes suivantes

- Vous êtes maintenant prêt à l'installer AWS SAMCLI et à commencer à l'utiliser AWS SAM. Pour installer le AWS SAMCLI, voirInstaller la CLI AWS SAM.

- Pour configurer Visual Studio Code pour le développement sans serveur, consultez Configuration du AWS Toolkit for VS Code.

Installer la CLI AWS SAM

Installez la dernière version de l'interface de ligne de AWS Serverless Application Model commande (AWS SAMCLI) sur les systèmes d'exploitation pris en charge en suivant les instructions figurant dansÉtape 4 : installez le AWS CLI.

Pour plus d'informations sur la gestion d'une version actuellement installée du AWS SAMCLI, notamment sur la mise à niveau, la désinstallation ou la gestion des versions nocturnes, consultezGestion des versions CLI AWS SAM.

> Est-ce la première fois que vous installez la AWS SAM CLI ?
>
> Remplissez l'ensemble des conditions préalables de la section précédente avant de poursuivre. Cela inclut notamment les éléments suivants :
>
> 1. Création d'un AWS compte.
> 2. Configuration d'un accès sécurisé à AWS.
> 3. Création d'un ID de clé d'accès et d'une clé d'accès secrète.
> 4. Installation du AWS CLI.
> 5. Configuration des AWS informations d'identification.

Rubriques

- Installation de la CLI AWS SAM
- Résolution des erreurs d'installation
- Étapes suivantes
- Facultatif : vérifier l'intégrité du AWS SAMCLI programme d'installation

Installation de la CLI AWS SAM

> **Note**
>
> À compter de septembre 2023, le programme d'Homebrewinstallation AWS géré pour le AWS SAMCLI (`aws/tap/aws-sam-cli`) ne AWS sera plus maintenu. Si vous avez l'Homebrewhabitude d'installer et de gérer le AWS SAMCLI, consultez les options suivantes :
>
> - Pour continuer à utiliser Homebrew, vous pouvez utiliser le programme d'installation géré par la communauté. Pour de plus amples informations, veuillez consulter Gestion de la CLI AWS SAM avec Homebrew.
> - Nous vous recommandons d'utiliser l'une des méthodes d'installation de premier niveau décrites sur cette page. Avant d'utiliser l'une de ces méthodes, consultez Passer de Homebrew.
> - Pour plus de détails, reportez-vous à la version de publication : 1.121.0.

Pour l'installer AWS SAMCLI, suivez les instructions correspondant à votre système d'exploitation.

Linux

x86_64 - command line installer

1. Téléchargez le fichier .zip de la CLI AWS SAM dans un répertoire de votre choix.
2. (Facultatif) Vous pouvez vérifier l'intégrité du programme d'installation avant l'installation. Pour obtenir des instructions, veuillez consulter Facultatif : vérifier l'intégrité du AWS SAMCLI programme d'installation.
3. Décompressez les fichiers d'installation dans le répertoire de votre choix. Voici un exemple avec utilisation du sous-répertoire `sam-installation`.

   > **Note**
   >
   > Si votre système d'exploitation ne dispose pas de la commande unzip intégrée, utilisez un équivalent.

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

4. Installez la CLI AWS SAM en lançant l'exécutable `install`. L'exécutable se trouve dans le répertoire utilisé à l'étape précédente. Voici un exemple avec utilisation du sous-répertoire `sam-installation` :

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

5. Vérifiez l'installation.

```
$ sam --version
```

Pour confirmer la réussite de l'installation, vous devriez voir une sortie qui remplace le texte entre crochets suivant par la dernière version disponible :

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

arm64 - command line installer

1. Téléchargez le fichier .zip de la CLI AWS SAM dans un répertoire de votre choix.
2. (Facultatif) Vous pouvez vérifier l'intégrité du programme d'installation avant l'installation. Pour obtenir des instructions, veuillez consulter Facultatif : vérifier l'intégrité du AWS SAMCLI programme d'installation.
3. Décompressez les fichiers d'installation dans le répertoire de votre choix. Voici un exemple avec utilisation du sous-répertoire `sam-installation`.

> **Note**
> Si votre système d'exploitation ne dispose pas de la commande unzip intégrée, utilisez un équivalent.

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

4. Installez la CLI AWS SAM en lançant l'exécutable `install`. L'exécutable se trouve dans le répertoire utilisé à l'étape précédente. Voici un exemple avec utilisation du sous-répertoire `sam-installation` :

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

5. Vérifiez l'installation.

```
$ sam --version
```

Pour confirmer la réussite de l'installation, vous devriez voir un résultat comme celui-ci, mais qui remplace le texte entre crochets par la dernière version de la CLI SAM :

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

macOS

Étapes d'installation

Utilisez le programme d'installation du package pour installer le AWS SAMCLI. En outre, le programme d'installation du package propose deux méthodes d'installation parmi lesquelles vous pouvez choisir : interface graphique et ligne de commande. Vous pouvez l'installer pour tous les utilisateurs ou uniquement pour votre utilisateur actuel. Pour effectuer l'installation pour tous les utilisateurs, l'autorisation de superutilisateur est requise.

> **Note**
>
> AWS SAM La CLI ne prend pas en charge les versions de macOS antérieures à macOS 13.x. Effectuez une mise à niveau vers une version compatible de macOS ou installez la AWS SAM CLI avecHomebrew.

GUI - All users

Pour télécharger le programme d'installation du package et installer le AWS SAMCLI

> **Note**
>
> Si vous avez déjà installé la CLI AWS SAM via Homebrew ou pip, vous devez d'abord la désinstaller. Pour obtenir des instructions, veuillez consulter Désinstallation de la CLI AWS SAM.

1. Téléchargez le macOS `pkg` dans le répertoire de votre choix :

   - Pour les Mac équipés de processeurs Intel, choisissez x86_64 — -x86_64.pkg aws-sam-cli-macos
   - Pour les Mac équipés d'Apple Silicon, choisissez arm64 — -arm64.pkg aws-sam-cli-macos

   > **Note**
   >
   > Vous avez la possibilité de vérifier l'intégrité du programme d'installation avant l'installation. Pour obtenir des instructions, veuillez consulter Facultatif : vérifier l'intégrité du AWS SAMCLI programme d'installation.

2. Exécutez le fichier téléchargé et suivez les instructions qui s'affichent à l'écran pour suivre les étapes Introduction, Read Me (Lisez-moi) et License (Licence).
3. Pour Destination Select (Sélection de destination), sélectionnez Install for all users of this computer (Installer pour tous les utilisateurs de cet ordinateur).
4. Dans Type d'installation, choisissez l'emplacement d'installation de la CLI AWS SAM et appuyez sur Installer. L'emplacement par défaut recommandé est `/usr/local/aws-sam-cli`.

   > **Note**
   >
   > Pour invoquer la CLI AWS SAM avec la commande sam, le programme d'installation crée automatiquement un lien symbolique entre `/usr/local/bin/sam` et `/usr/local/aws-sam-cli/sam` ou le dossier d'installation que vous avez choisi.

5. La CLI AWS SAM va s'installer et le message L'installation s'est déroulée avec succès s'affichera. Appuyez sur Close (Fermer).

Pour vérifier la réussite de l'installation

- Vérifiez que la CLI AWS SAM est correctement installée et que votre lien symbolique est configuré en exécutant :

```
$ which sam
/usr/local/bin/sam
$ sam --version
```

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

GUI - Current user

Pour télécharger et installer AWS SAMCLI

> **Note**
>
> Si vous avez déjà installé la CLI AWS SAM via Homebrew ou pip, vous devez d'abord la désinstaller. Pour obtenir des instructions, veuillez consulter Désinstallation de la CLI AWS SAM.

1. Téléchargez le macOS `pkg` dans le répertoire de votre choix :
   - Pour les Mac équipés de processeurs Intel, choisissez x86_64 — -x86_64.pkg aws-sam-cli-macos
   - Pour les Mac équipés d'Apple Silicon, choisissez arm64 — -arm64.pkg aws-sam-cli-macos

   > **Note**
   >
   > Vous avez la possibilité de vérifier l'intégrité du programme d'installation avant l'installation. Pour obtenir des instructions, veuillez consulter Facultatif : vérifier l'intégrité du AWS SAMCLI programme d'installation.

2. Exécutez le fichier téléchargé et suivez les instructions qui s'affichent à l'écran pour suivre les étapes Introduction, Read Me (Lisez-moi) et License (Licence).
3. Pour Destination Select (Sélection de destination), sélectionnez Install for me only (Installer pour moi uniquement). Si vous ne voyez pas cette option, passez à l'étape suivante.
4. Pour Installation Type (Type d'installation), procédez comme suit :
   1. Choisissez l'emplacement d'installation de la CLI AWS SAM. L'emplacement par défaut est `/usr/local/aws-sam-cli`. Sélectionnez un emplacement pour lequel vous disposez d'autorisations d'écriture. Pour modifier l'emplacement d'installation, sélectionnez local et choisissez votre emplacement. Appuyez sur Continue (Continuer) lorsque vous avez terminé.

2. Si vous n'avez pas eu la possibilité de choisir Install for me only (Installer pour moi uniquement) à l'étape précédente, sélectionnez Change Install Location (Modifier l'emplacement d'installation) > Install for me only et appuyez sur Continue (Continuer).
3. Appuyez sur Install (Installer).

5. La CLI AWS SAM va s'installer et le message L'installation s'est déroulée avec succès s'affichera. Appuyez sur Close (Fermer).

Pour créer un lien symbolique

- Pour invoquer la CLI AWS SAMà l'aide de la commande sam, vous devez créer manuellement un lien symbolique entre le programme CLI AWS SAM et votre `$PATH`. Créez votre lien symbolique en modifiant et en exécutant la commande suivante :

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

  - *`sudo`*— Si votre utilisateur dispose d'autorisations d'écriture pour`$PATH`, ce n'sudoest pas obligatoire. Dans le cas contraire, la valeur sudo est obligatoire.
  - *`path-to`*— Chemin d'accès à l'endroit où vous avez installé le AWS SAMCLI programme. Par exemple, `/Users/myUser/Desktop`.
  - *`path-to-symlink-directory`*— Votre variable d'`$PATH`environnement. L'emplacement par défaut est `/usr/local/bin`.

Pour vérifier la réussite de l'installation

- Vérifiez que la CLI AWS SAM est correctement installée et que votre lien symbolique est configuré en exécutant :

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

Command line - All users

Pour télécharger et installer AWS SAMCLI

> **Note**
>
> Si vous avez déjà installé la CLI AWS SAM via Homebrew ou pip, vous devez d'abord la désinstaller. Pour obtenir des instructions, veuillez consulter Désinstallation de la CLI AWS SAM.

1. Téléchargez le macOS `pkg` dans le répertoire de votre choix :

   - Pour les Mac équipés de processeurs Intel, choisissez x86_64 — -x86_64.pkg aws-sam-cli-macos
   - Pour les Mac équipés d'Apple Silicon, choisissez arm64 — -arm64.pkg aws-sam-cli-macos

   > **Note**
   >
   > Vous avez la possibilité de vérifier l'intégrité du programme d'installation avant l'installation. Pour obtenir des instructions, veuillez consulter Facultatif : vérifier l'intégrité du AWS SAMCLI programme d'installation.

2. Modifiez et exécutez le script d'installation :

```
$ 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**
   >
   > Pour invoquer la CLI AWS SAM avec la commande sam, le programme d'installation crée automatiquement un lien symbolique entre `/usr/local/bin/sam` et `/usr/local/aws-sam-cli/sam`.

Pour vérifier la réussite de l'installation

- Vérifiez que la CLI AWS SAM est correctement installée et que votre lien symbolique est configuré en exécutant :

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

Command line - Current user

Pour télécharger et installer AWS SAMCLI

> **Note**
>
> Si vous avez déjà installé la CLI AWS SAM via Homebrew ou pip, vous devez d'abord la désinstaller. Pour obtenir des instructions, veuillez consulter Désinstallation de la CLI AWS SAM.

1. Téléchargez le macOS `pkg` dans le répertoire de votre choix :

   - Pour les Mac équipés de processeurs Intel, choisissez x86_64 — -x86_64.pkg aws-sam-cli-macos
   - Pour les Mac équipés d'Apple Silicon, choisissez arm64 — -arm64.pkg aws-sam-cli-macos

   > **Note**
   >
   > Vous avez la possibilité de vérifier l'intégrité du programme d'installation avant l'installation. Pour obtenir des instructions, veuillez consulter Facultatif : vérifier l'intégrité du AWS SAMCLI programme d'installation.

2. Déterminez le répertoire d'installation pour lequel vous disposez d'autorisations d'écriture. Créez ensuite un fichier `xml` à l'aide du modèle et modifiez-le pour qu'il reflète votre répertoire d'installation. Le répertoire doit déjà exister.

Par exemple, si vous remplacez *path-to-my-directory* par`/Users/myUser/Desktop`, le dossier `aws-sam-cli` du programme y sera installé.

```
<?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. Enregistrez le fichier `xml` et vérifiez sa validité en procédant ainsi :

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

La réponse générée doit afficher les préférences qui seront appliquées au programme CLI AWS SAM.

4. Exécutez ce qui suit pour installer 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.
```

Pour créer un lien symbolique

- Pour invoquer la CLI AWS SAMà l'aide de la commande sam, vous devez créer manuellement un lien symbolique entre le programme CLI AWS SAM et votre `$PATH`. Créez votre lien symbolique en modifiant et en exécutant la commande suivante :

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

  - *sudo*— Si votre utilisateur dispose d'autorisations d'écriture pour`$PATH`, ce n'sudoest pas obligatoire. Dans le cas contraire, la valeur sudo est obligatoire.
  - *path-to*— Chemin d'accès à l'endroit où vous avez installé le AWS SAMCLI programme. Par exemple, `/Users/myUser/Desktop`.
  - *path-to-symlink-directory*— Votre variable d'`$PATH`environnement. L'emplacement par défaut est `/usr/local/bin`.

Pour vérifier la réussite de l'installation

- Vérifiez que la CLI AWS SAM est correctement installée et que votre lien symbolique est configuré en exécutant :

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

Windows

Les fichiers Windows Installer (MSI) sont les fichiers du programme d'installation du package pour le système d'exploitation Windows.

Procédez comme suit pour installer la CLI AWS SAMà l'aide du fichier MSI.

1. Téléchargez la CLI AWS SAM 64 bits.
2. (Facultatif) Vous pouvez vérifier l'intégrité du programme d'installation avant l'installation. Pour obtenir des instructions, veuillez consulter Facultatif : vérifier l'intégrité du AWS SAMCLI programme d'installation.
3. Vérifiez l'installation.

Une fois l'installation terminée, vérifiez-la en ouvrant une nouvelle invite de commande ou une nouvelle PowerShell invite. Vous devriez être en mesure d'appeler `sam` à partir de la ligne de commande.

```
sam --version
```

Une fois l'installation réussie du AWS SAMCLI, vous devriez voir le résultat suivant :

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

4. Activez les chemins d'accès longs (Windows 10 et versions ultérieures uniquement).

> **Important**
>
> Ils AWS SAMCLI peuvent interagir avec des chemins de fichiers qui dépassent la limite de chemin maximale de Windows. Cela peut provoquer des erreurs lors de l'exécution `sam init` en raison des MAX_PATH limitations de Windows 10. Pour résoudre ce problème, vous devez configurer le comportement des nouveaux chemins d'accès longs.

Pour activer les chemins d'accès longs, veuillez consulter la rubrique Activer les chemins longs dans Windows 10, version 1607 et ultérieures dans la documentation de développement d'applications Microsoft Windows.

5. Installez Git.

Pour télécharger des exemples d'applications à l'aide de la commande `sam init`, vous devez également installer Git. Pour obtenir des instructions, veuillez consulter Installation de Git.

Résolution des erreurs d'installation

Linux

Erreur Docker : « Impossible de se connecter au Démon Docker. Le démon docker fonctionne-t-il sur cet hôte ? »

Dans certains cas, pour autoriser `ec2-user` à accéder au démon Docker, vous devrez peut-être redémarrer votre instance. Si vous recevez cette erreur, essayez de redémarrer votre instance.

Erreur shell : « commande introuvable »

Si vous recevez cette erreur, votre shell ne peut pas localiser l'exécutable CLI AWS SAM dans le chemin d'accès. Vérifiez l'emplacement du répertoire dans lequel vous avez installé l'exécutable CLI AWS SAM, puis vérifiez que le répertoire se trouve sur votre chemin d'accès.

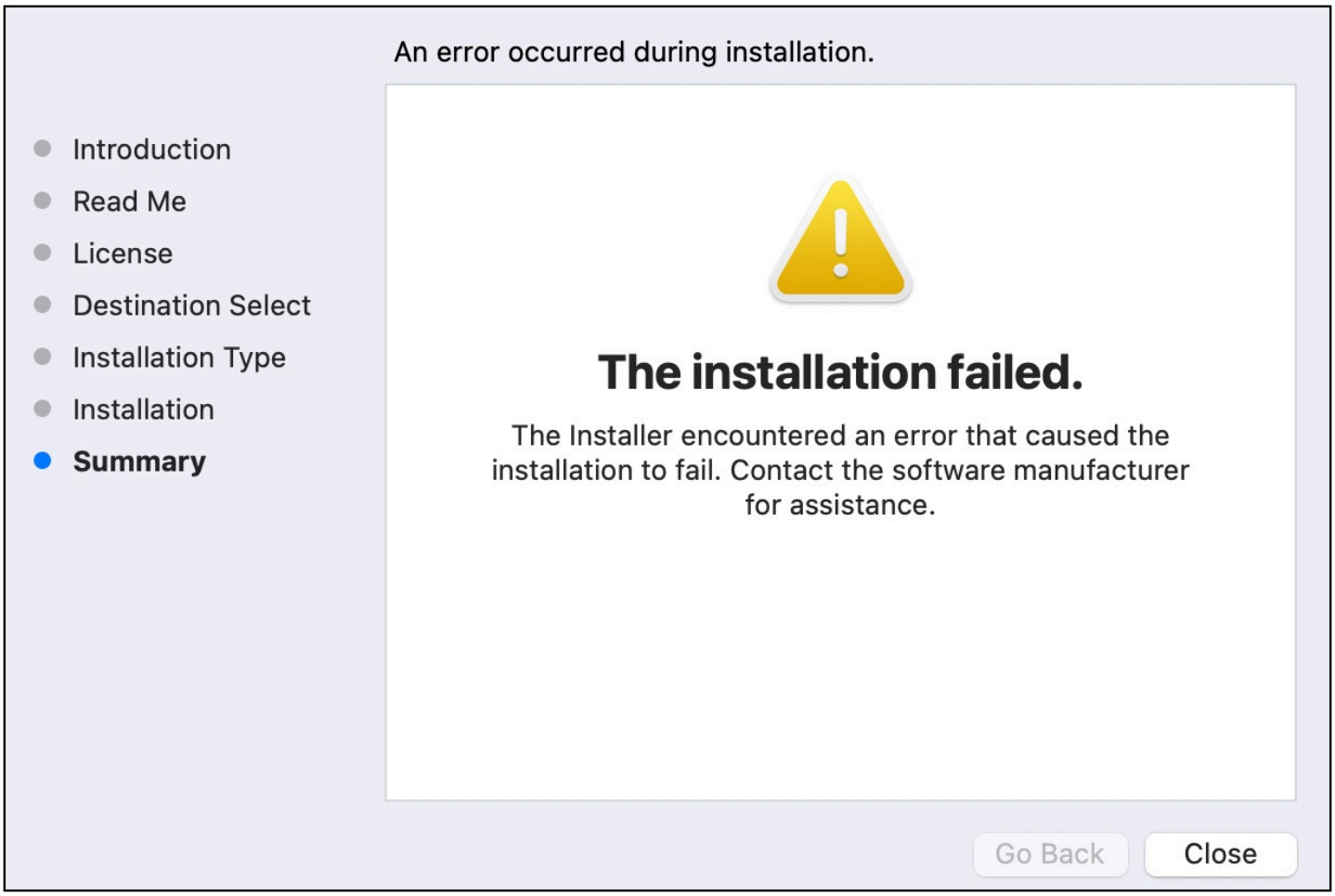
AWS SAMCLIerreur : « /lib64/libc.so.6 : version `GLIBC_2.14' introuvable (requise par/.so.1) » usr/local/aws-sam-cli/dist/libz

Si vous recevez cette erreur, vous utilisez une version non prise en charge de Linux et la version glibc intégrée est obsolète. Essayez l'une des actions suivantes :

- Mettez à niveau votre hôte Linux vers la version 64 bits d'une distribution récente de CentOS, Fedora, Ubuntu ou Amazon Linux 2.
- Suivez les instructions ci-dessous pour Installer la CLI AWS SAM.

macOS

Échec de l'installation



Si vous installez la CLI AWS SAM pour votre utilisateur et que vous avez sélectionné un répertoire d'installation pour lequel vous ne disposez pas d'autorisations d'écriture, cette erreur peut survenir. Essayez l'une des actions suivantes :

1. Sélectionnez un autre répertoire d'installation pour lequel vous disposez d'autorisations d'écriture.
2. Supprimez le programme d'installation. Ensuite, téléchargez-le et exécutez-le à nouveau.

Étapes suivantes

Pour en savoir plus sur la CLI AWS SAM et pour commencer à créer vos propres applications sans serveur, veuillez consulter les ressources suivantes :

- Tutoriel : Déployer une application Hello World avec AWS SAM— Step-by-step instructions pour télécharger, créer et déployer une application sans serveur de base.
- L' AWS SAM atelier complet — Un atelier conçu pour vous enseigner les nombreuses fonctionnalités principales qu'il AWS SAM fournit.
- AWS SAM exemples d'applications et de modèles : exemples d'applications et de modèles créés par des auteurs de la communauté que vous pouvez tester plus avant.

Facultatif : vérifier l'intégrité du AWS SAMCLI programme d'installation

Lorsque vous installez l'interface de ligne de AWS Serverless Application Model commande (AWS SAMCLI) à l'aide d'un programme d'installation de package, vous pouvez vérifier son intégrité avant l'installation. Il s'agit d'une étape facultative, mais vivement recommandée.

Les deux options de vérification disponibles sont les suivantes :

- Vérifier le fichier de signature du programme d'installation de packages.
- Vérifier la valeur de hachage du programme d'installation de packages.

Nous vous recommandons l'option de vérification du fichier de signature lorsqu'elle est disponible pour votre plateforme. Cette option offre un niveau de sécurité supplémentaire puisque les valeurs clés sont publiées ici et gérées séparément de notre référentiel GitHub.

Rubriques

- Vérification du fichier de signature du programme d'installation
- Vérification de la valeur de hachage

Vérification du fichier de signature du programme d'installation

Linux

arm64 : installateur en ligne de commande

AWS SAM utilise GnuPG pour signer AWS SAMCLI le programme d'installation .zip. La vérification s'effectue selon les étapes suivantes :

1. Utilisation de la clé publique principale pour vérifier la clé publique du signataire.

2. Utilisation de la clé publique du signataire pour vérifier le programme d'installation de packages de la CLI AWS SAM.

Pour vérifier l'intégrité de la clé publique du signataire

1. Copiez la clé publique principale et enregistrez-la sur votre machine locale sous la forme d'un fichier `.txt`. Par exemple, *`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. Importez la clé publique principale dans votre porte-clés.

```
$ 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. Copiez la clé publique du signataire et enregistrez-la sur votre machine locale sous la forme d'un fichier `.txt`. Par exemple, *`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. Importez la clé publique du signataire dans votre porte-clés.

```
$ 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
```

Prenez note de la valeur de la clé à partir de la sortie. Par exemple, *4094ABB1BEDFDAB4*.

5. Utilisez la valeur de la clé pour obtenir et vérifier l'empreinte digitale de la clé publique du signataire.

```
$ 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>
```

L'empreinte digitale doit correspondre à ce qui suit :

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

Si la chaîne de l'empreinte digitale ne correspond pas, n'utilisez pas le programme d'installation de la CLI AWS SAM. Soumettez le problème à l' AWS SAM équipe en créant un problème dans le aws-sam-cli GitHub référentiel.

6. Vérifiez les signatures de la clé publique du signataire :

```
$ 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>
```

Si vous voyez `1 signature not checked due to a missing key`, répétez les étapes précédentes pour importer la clé publique principale et la clé publique du signataire dans votre porte-clés.

Les valeurs des clés de la clé publique principale et de la clé publique du signataire devraient être répertoriées.

Maintenant que vous avez vérifié l'intégrité de la clé publique du signataire, vous pouvez l'utiliser pour vérifier le programme d'installation de packages de la CLI AWS SAM.

Pour vérifier l'intégrité du programme d'installation de packages de la CLI AWS SAM

1. Obtenez le fichier de signature du package de la CLI AWS SAM : téléchargez le fichier de signature du programme d'installation de packages de la CLI AWS SAMà l'aide de la commande suivante :

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

2. Vérifiez le fichier de signature : passez les fichiers `.sig` et `.zip` téléchargés en tant que paramètres de la commande gpg. Voici un exemple :

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

La sortie doit ressembler à ce qui suit :

```
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
```

- Le message `WARNING: This key is not certified with a trusted signature!` peut être ignoré. Cet avertissement est dû au fait qu'il n'y a pas de chaîne de confiance entre votre clé PGP personnelle (si vous en avez une) et la clé PGP de la CLI AWS SAM . Pour de plus amples informations, consultez Web of trust.
- Si le résultat contient l'expression `BAD signature`, vérifiez que vous avez effectué la procédure correctement. Si vous continuez à recevoir cette réponse, signalez-le à l' AWS SAM équipe en créant un problème dans le aws-sam-cli GitHub référentiel et en évitant d'utiliser le fichier téléchargé.

Le message `Good signature from "AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>"` signifie que la signature est vérifiée et que vous pouvez poursuivre l'installation.

x86_64 – programme d'installation en ligne de commande

AWS SAM utilise GnuPG pour signer AWS SAMCLI le programme d'installation .zip. La vérification s'effectue selon les étapes suivantes :

1. Utilisation de la clé publique principale pour vérifier la clé publique du signataire.
2. Utilisation de la clé publique du signataire pour vérifier le programme d'installation de packages de la CLI AWS SAM.

Pour vérifier l'intégrité de la clé publique du signataire

1. Copiez la clé publique principale et enregistrez-la sur votre machine locale sous la forme d'un fichier `.txt`. Par exemple, *`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. Importez la clé publique principale dans votre porte-clés.

```
$ 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. Copiez la clé publique du signataire et enregistrez-la sur votre machine locale sous la forme d'un fichier `.txt`. Par exemple, *`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. Importez la clé publique du signataire dans votre porte-clés.

```
$ 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
```

Prenez note de la valeur de la clé à partir de la sortie. Par exemple, *4094ABB1BEDFDAB4*.

5. Utilisez la valeur de la clé pour obtenir et vérifier l'empreinte digitale de la clé publique du signataire.

```
$ 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>
```

L'empreinte digitale doit correspondre à ce qui suit :

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

Si la chaîne de l'empreinte digitale ne correspond pas, n'utilisez pas le programme d'installation de la CLI AWS SAM. Soumettez le problème à l' AWS SAM équipe en créant un problème dans le aws-sam-cli GitHub référentiel.

6. Vérifiez les signatures de la clé publique du signataire :

```
$ 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>
```

Si vous voyez `1 signature not checked due to a missing key`, répétez les étapes précédentes pour importer la clé publique principale et la clé publique du signataire dans votre porte-clés.

Les valeurs des clés de la clé publique principale et de la clé publique du signataire devraient être répertoriées.

Maintenant que vous avez vérifié l'intégrité de la clé publique du signataire, vous pouvez l'utiliser pour vérifier le programme d'installation de packages de la CLI AWS SAM.

Pour vérifier l'intégrité du programme d'installation de packages de la CLI AWS SAM

1. Obtenez le fichier de signature du package de la CLI AWS SAM : téléchargez le fichier de signature du programme d'installation de packages de la CLI AWS SAMà l'aide de la commande suivante :

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

2. Vérifiez le fichier de signature : passez les fichiers `.sig` et `.zip` téléchargés en tant que paramètres de la commande gpg. Voici un exemple :

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

La sortie doit ressembler à ce qui suit :

```
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
```

- Le message `WARNING: This key is not certified with a trusted signature!` peut être ignoré. Cet avertissement est dû au fait qu'il n'y a pas de chaîne de confiance entre votre clé PGP personnelle (si vous en avez une) et la clé PGP de la CLI AWS SAM . Pour de plus amples informations, consultez Web of trust.
- Si le résultat contient l'expression `BAD signature`, vérifiez que vous avez effectué la procédure correctement. Si vous continuez à recevoir cette réponse, signalez-le à l' AWS SAM équipe en créant un problème dans le aws-sam-cli GitHub référentiel et en évitant d'utiliser le fichier téléchargé.

Le message `Good signature from "AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>"` signifie que la signature est vérifiée et que vous pouvez poursuivre l'installation.

macOS

Programme d'installation à interface graphique et en ligne de commande

Vous pouvez vérifier l'intégrité du fichier de signature du programme d'installation de packages de la CLI AWS SAMà l'aide de l'outil `pkgutil` ou manuellement.

Pour vérifier à l'aide de l'outil pkgutil

1. Exécutez la commande suivante, en indiquant le chemin d'accès au programme d'installation téléchargé sur votre ordinateur local :

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

   Voici un exemple :

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

2. Dans le résultat, localisez SHA256 fingerprint pour Developer ID Installer: AMZN Mobile LLC. Voici un exemple :

   ```
   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. Developer ID Installer: AMZN Mobile LLC SHA256 fingerprint doit correspondre à la valeur suivante :

```
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
```

   Si la chaîne de l'empreinte digitale ne correspond pas, n'utilisez pas le programme d'installation de la CLI AWS SAM. Soumettez le problème à l' AWS SAM équipe en créant un problème dans le aws-sam-cli GitHub référentiel. Si la chaîne d'empreintes digitales correspond, vous pouvez continuer à utiliser le programme d'installation de packages.

Pour vérifier manuellement le programme d'installation de packages

- Consultez Vérification de l'authenticité des mises à jour logicielles Apple téléchargées manuellement (français non garanti) sur le site web d'assistance Apple.

Windows

Le AWS SAMCLI programme d'installation est fourni sous forme de MSI fichiers pour le système Windows d'exploitation.

Pour vérifier l'intégrité du programme d'installation

1. Cliquez avec le bouton droit de la souris sur le programme d'installation et ouvrez la fenêtre Propriétés.
2. Choisissez l'onglet Signatures numériques.
3. Dans Liste des signatures, choisissez Amazon Web Services, Inc., puis Détails.
4. Choisissez l'onglet General (Général), s'il n'est pas déjà sélectionné, puis View Certificate (Afficher le certificat).
5. Sélectionnez l'onglet Détails, puis sélectionnez All (Tous) dans la liste déroulante Show (Afficher), si cette option n'est pas déjà sélectionnée.
6. Faites défiler l'écran vers le bas jusqu'au champ Thumbprint (Empreinte), puis choisissez Thumbprint (Empreinte). Cela affichera la valeur complète de l'empreinte dans la fenêtre inférieure.
7. Faites correspondre la valeur de l'empreinte à la valeur suivante. Si la valeur correspond, poursuivez l'installation. Si ce n'est pas le cas, signalez-le à l' AWS SAM équipe en créant un problème dans le aws-sam-cli GitHub référentiel.

```
cd62479397f09d72a04c7399a254b0a91da53d6c
```

Vérification de la valeur de hachage

Linux

x86_64 – programme d'installation en ligne de commande

Vérifiez l'intégrité et l'authenticité des fichiers d'installation téléchargés en générant une valeur de hachage à l'aide de la commande suivante :

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

Le résultat doit ressembler à l'exemple qui suit :

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

Comparez la valeur de hachage SHA-256 de 64 caractères avec celle de la version de CLI AWS SAM de votre choix dans les Notes de mise à jour de la CLI AWS SAM sur GitHub.

macOS

Programme d'installation à interface graphique et en ligne de commande

Vérifiez l'intégrité et l'authenticité du programme d'installation téléchargé en générant une valeur de hachage à l'aide de la commande suivante :

```
$ 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
```

Comparez votre valeur de hachage SHA-256 à 64 caractères avec la valeur correspondante dans le référentiel GitHub des notes de mise à jour de la CLI AWS SAM.

Tutoriel : Déployer une application Hello World avec AWS SAM

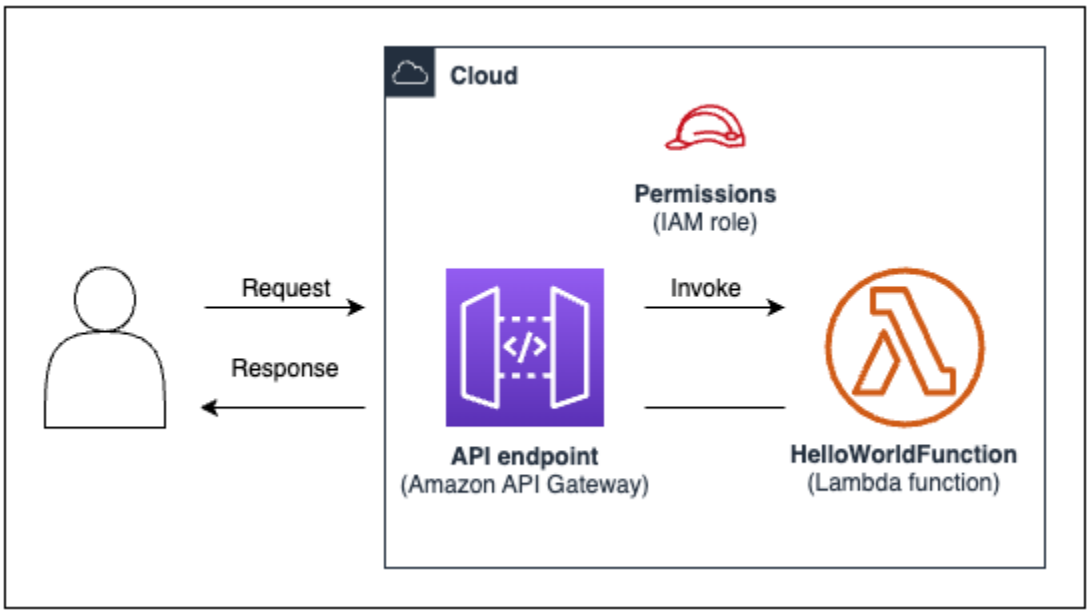
Dans ce didacticiel, vous allez utiliser l'interface de ligne de AWS Serverless Application Model commande (AWS SAMCLI) pour effectuer les opérations suivantes :

- Initialiser, créer et déployer un exemple d'application Hello World.
- Apportez des modifications locales et synchronisez avec AWS CloudFormation.
- Effectuez des tests locaux sur votre hébergeur de développement.
- Supprimer l'exemple d'application du AWS Cloud.

L'exemple d'application Hello World implémente une API backend de base. Elle regroupe les ressources suivantes :

- Amazon API Gateway : point de terminaison d'API que vous utilisez pour appeler votre fonction.
- AWS Lambda : fonction qui traite la requête HTTP API GET et renvoie un message `hello world`.
- Gestion des identités et des accès AWS Rôle (IAM) : fournit des autorisations permettant aux services d'interagir en toute sécurité.

Le diagramme suivant montre les composants de cette application :



Rubriques

- Conditions préalables
- Étape 1 : initialisation de l'exemple d'application Hello World
- Étape 2 : créer votre application
- Étape 3 : Déployez votre application sur AWS Cloud
- Étape 4 : exécuter votre application
- Étape 5 : Interagissez avec votre fonction dans AWS Cloud
- Étape 6 : Modifiez et synchronisez votre application avec AWS Cloud
- Étape 7 : (Facultatif) testez votre application localement
- Étape 8 : Supprimez votre application du AWS Cloud
- Résolution des problèmes
- En savoir plus

Conditions préalables

Vérifiez que vous avez effectué les opérations suivantes :

- AWS SAM prérequis
- Installer la CLI AWS SAM

> **Note**
>
> Si vous préférez utiliser Visual Studio Code pour le développement sans serveur, consultez la section Création d'applications sans serveur dans le guide de l'utilisateur du AWS Toolkit for VS Code pour obtenir des step-by-step instructions sur l'utilisation de l'interface VS Code.

Étape 1 : initialisation de l'exemple d'application Hello World

Au cours de cette étape, vous allez utiliser la CLI AWS SAM pour créer un exemple de projet d'application Hello World sur votre ordinateur local.

Pour initialiser l'exemple d'application Hello World

1. Dans votre ligne de commande, exécutez la commande suivante à partir du répertoire de départ de votre choix :

```
$ sam init
```

> **Note**
>
> Cette commande initialise votre application sans serveur et crée un répertoire de projet. Le répertoire contient plusieurs fichiers et dossiers, dont le `template.yaml` fichier, qui est votre AWS SAM modèle.
> La version de Python installée doit correspondre à la `Runtime` propriété spécifiée dans`template.yaml`. Pour les charges de travail de production, nous recommandons d'utiliser Python 3.12 ou des versions antérieures entièrement prises en charge dans AWS les environnements. Si les versions de Python ne correspondent pas, vous rencontrerez des erreurs de compilation.
> Pour résoudre les problèmes de compatibilité des versions :
>
> - Spécifiez un environnement d'exécution compatible lors de l'initialisation : `sam init --runtime python3.9`
> - Modifier la `Runtime` propriété `template.yaml` après l'initialisation

2. La CLI AWS SAM vous guidera tout au long de l'initialisation d'une nouvelle application. Configurez ce qui suit :

1. Sélectionnez Modèles de démarrage rapide AWS pour choisir un modèle de départ.
2. Choisissez le modèle Hello World Example et téléchargez-le.
3. Utilisez l'exécution Python et le type de package `zip`.
4. Dans le cadre de ce didacticiel, désactivez le AWS X-Ray traçage. Pour en savoir plus, consultez Qu'est-ce que c'est AWS X-Ray ? dans le Guide AWS X-Ray du développeur.
5. Pour ce didacticiel, désactivez la surveillance avec Amazon CloudWatch Application Insights. Pour en savoir plus, consultez Amazon CloudWatch Application Insights dans le guide de CloudWatch l'utilisateur Amazon.
6. Pour ce didacticiel, désactivez la configuration de la journalisation structurée au format JSON sur vos fonctions Lambda.
7. Nommez votre application sam-app.

Pour utiliser le flux interactif de la CLI AWS SAM :

- Les crochets (`[ ]`) indiquent les valeurs par défaut. Laissez la réponse vide pour sélectionner la valeur par défaut.
- Saisissez **y** pour oui et **n** pour non.

Vous trouverez ci-dessous un exemple du flux interactif `sam init` :

```
$ 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. AWS SAMCLITélécharge ensuite votre modèle de départ et crée la structure du répertoire du projet d'application sur votre machine locale. Voici un exemple de réponse générée par la CLI AWS SAM :

```
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. À partir de votre ligne de commande, accédez au répertoire `sam-app` que vous venez de créer. Voici un exemple de ce que la CLI AWS SAM a créé :

> **Note**
>
> Si `tree` la commande n'est pas installée automatiquement, exécutez cette commande : `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
```

Quelques fichiers importants à signaler :

- `hello_world/app.py` : contient le code de votre fonction Lambda.
- `hello_world/requirements.txt` : contient toutes les dépendances Python requises par votre fonction Lambda.
- `samconfig.toml`— Fichier de configuration de votre application qui stocke les paramètres par défaut utilisés par le AWS SAMCLI.
- `template.yaml`— Le AWS SAM modèle qui contient le code de l'infrastructure de votre application.

Vous disposez désormais d'une application sans serveur entièrement créée sur votre ordinateur local !

Étape 2 : créer votre application

Au cours de cette étape, vous utilisez la CLI AWS SAM pour créer votre application et préparer le déploiement. Lorsque vous créez, la CLI AWS SAM crée un répertoire `.aws-sam` et organise vos dépendances de fonctions, votre code de projet et vos fichiers de projet.

Pour créer votre application

- Dans votre ligne de commande, à partir du répertoire du projet `sam-app`, exécutez ce qui suit :

```
$ sam build
```

> **Note**
>
> Si vous n'avez pas Python sur votre ordinateur local, utilisez la commande sam build --use-container à la place. La CLI AWS SAM créera un conteneur Docker qui inclut l'exécution et les dépendances de votre fonction. Cette commande nécessite Docker sur votre ordinateur local. Pour installer Docker, consultez Installation de Docker.

Voici un exemple de réponse générée par la CLI AWS SAM :

```
$ 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
```

Voici un exemple abrégé du répertoire `.aws-sam` créé par la CLI AWS SAM :

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

Quelques fichiers importants à signaler :

- `build/HelloWorldFunction` : contient le code de votre fonction Lambda et ses dépendances. La CLI AWS SAM crée un répertoire pour chaque fonction de votre application.
- `build/template.yaml`— Contient une copie de votre AWS SAM modèle référencée par CloudFormation lors du déploiement.

- `build.toml` : le fichier de configuration qui stocke les valeurs de paramètres par défaut référencées par la CLI AWS SAM lors de la création et du déploiement de votre application.

Vous êtes maintenant prêt à déployer votre application sur le AWS Cloud.

Étape 3 : Déployez votre application sur AWS Cloud

> **Note**
>
> Cette étape nécessite la configuration AWS des informations d'identification. Pour plus d'informations, consultez Étape 5 : utilisez le AWS CLI pour configurer les AWS informations d'identification dans AWS SAM prérequis.

Au cours de cette étape, vous utilisez la CLI AWS SAM pour déployer votre application sur le AWS Cloud. Ils AWS SAMCLI effectueront les opérations suivantes :

- Vous guide dans la configuration des paramètres de votre application pour le déploiement.
- Charge les fichiers de votre application sur Amazon Simple Storage Service (Amazon S3).
- Transformez votre AWS SAM modèle en CloudFormation modèle. Il télécharge ensuite votre modèle sur le CloudFormation service pour fournir vos AWS ressources.

Pour déployer votre application

1. Dans votre ligne de commande, à partir du répertoire du projet `sam-app`, exécutez ce qui suit :

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

2. Suivez le flux interactif de la CLI AWS SAM pour configurer les paramètres de votre application. Configurez ce qui suit :

   1. Le nom de la CloudFormation pile — Une pile est un ensemble de AWS ressources que vous pouvez gérer comme une seule unité. Pour en savoir plus, consultez la section Utilisation des piles dans le Guide de l'utilisateur AWS CloudFormation .
   2. Le Région AWSsur lequel déployer votre CloudFormation stack. Pour de plus amples informations, consultez la section Points de terminaison CloudFormation dans le Guide de l'utilisateur AWS CloudFormation .

3. Pour ce didacticiel, désactivez la confirmation des modifications avant le déploiement.
4. Autoriser la création de rôles IAM : cela permet de AWS SAM créer le rôle IAM nécessaire à l'interaction entre votre ressource API Gateway et votre ressource de fonction Lambda.
5. Pour ce didacticiel, désactivez l'option de désactivation de la restauration.
6. Autoriser HelloWorldFunction sans autorisation définie : ce message s'affiche car votre point de terminaison API Gateway est configuré pour être accessible au public, sans autorisation. Comme il s'agit de la configuration prévue pour votre application Hello World, autorisez la CLI AWS SAMà continuer. Pour plus d'informations sur la configuration de l'autorisation, consultez Contrôlez l'accès aux API avec votre AWS SAM modèle.
7. Enregistrer les arguments dans le fichier de configuration : cela mettra à jour le fichier `samconfig.toml` de votre application avec vos préférences de déploiement.
8. Sélectionnez le nom du fichier de configuration par défaut.
9. Sélectionnez l'environnement de configuration par défaut.

Vous trouverez ci-dessous un exemple du résultat du flux interactif `sam deploy --guided` :

```
$ 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. AWS SAMCLIDéploie ensuite votre application en procédant comme suit :

   - La CLI AWS SAM crée un compartiment Amazon S3 et charge votre répertoire `.aws-sam`.
   - AWS SAMCLITransforme votre AWS SAM modèle CloudFormation et le télécharge sur le CloudFormation service.
   - CloudFormation approvisionne vos ressources.

   Pendant le déploiement, la CLI AWS SAM affiche votre progression. Voici un exemple de résultat :

```
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
```

Votre application est désormais déployée et s'exécute dans le AWS Cloud !

Étape 4 : exécuter votre application

Au cours de cette étape, vous allez envoyer une requête GET à votre point de terminaison d'API et consulter le résultat de votre fonction Lambda.

Pour obtenir la valeur du point de terminaison de votre API

1. À partir des informations affichées par la CLI AWS SAMà l'étape précédente, localisez la section `Outputs`. Dans cette section, recherchez votre ressource `HelloWorldApi` pour trouver la valeur de votre point de terminaison HTTP. Voici un exemple de résultat :

```
-----------------------------------------------------------------------------------------------------
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. Vous pouvez également utiliser la commande sam list endpoints --output json pour obtenir ces informations. Voici un exemple de résultat :

```
$ 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']"
    ]
  }
]
```

Pour appeler votre fonction

- À l'aide de votre navigateur ou de la ligne de commande, envoyez une requête GET au point de terminaison de votre API. Voici un exemple d'utilisation de la commande curl :

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

Étape 5 : Interagissez avec votre fonction dans AWS Cloud

Au cours de cette étape, vous utilisez la CLI AWS SAM pour invoquer votre fonction Lambda sur le AWS Cloud.

Pour invoquer votre fonction Lambda dans le cloud

1. Prenez note des fonctions `LogicalResourceId` que vous avez utilisées à l'étape précédente. Ça devrait être `HelloWorldFunction`.
2. Dans votre ligne de commande, à partir du répertoire du projet `sam-app`, exécutez ce qui suit :

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

3. AWS SAMCLIInvoque votre fonction dans le cloud et renvoie une réponse. Voici un exemple de résultat :

```
$ 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\"}"}%
```

Étape 6 : Modifiez et synchronisez votre application avec AWS Cloud

Au cours de cette étape, vous utilisez la AWS SAMCLI sam sync --watch commande pour synchroniser les modifications locales apportées au AWS Cloud.

Pour utiliser sam sync

1. Dans votre ligne de commande, à partir du répertoire du projet `sam-app`, exécutez ce qui suit :

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

2. La CLI AWS SAM vous invite à confirmer que vous synchronisez une pile de développement. Étant donné que la sam sync --watch commande déploie automatiquement les modifications locales AWS Cloud en temps réel, nous la recommandons uniquement pour les environnements de développement.

   La CLI AWS SAM effectue un déploiement initial avant de commencer à surveiller les modifications locales. Voici un exemple de résultat :

   ```
   $ 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.
```

Ensuite, vous allez modifier le code de votre fonction Lambda. Le AWS SAMCLI détectera automatiquement ce changement et synchronisera votre application avec le AWS Cloud.

Pour modifier et synchroniser votre application

1. Dans l'IDE de votre choix, ouvrez le fichier `sam-app/hello_world/app.py`.
2. Remplacez le `message` et enregistrez le fichier. Voici un exemple :

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

3. AWS SAMCLIDétecte votre modification et synchronise votre application avec le AWS Cloud. Voici un exemple de résultat :

```
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. Pour vérifier votre modification, envoyez à nouveau une requête GET au point de terminaison de votre API.

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

Étape 7 : (Facultatif) testez votre application localement

> **Note**
>
> Cette étape est facultative.

> **Important**
>
> Cette étape nécessite que vous Docker utilisiez votre machine locale. Vous devez avoir Docker installé et configuré pour utiliser le AWS SAMCLI pour les tests locaux. Pour de plus amples informations, veuillez consulter Installation de Docker.

Au cours de cette étape, vous devez utiliser la AWS SAMCLI sam local commande pour tester votre application localement. Pour ce faire, la CLI AWS SAM crée un environnement local à l'aide de Docker. Cet environnement local émule l'environnement d'exécution basé sur le cloud de votre fonction Lambda.

Vous effectuez les actions suivantes :

1. Créez un environnement local pour votre fonction Lambda et appelez-la.
2. Hébergez le point de terminaison de votre API HTTP en local et utilisez-le pour appeler votre fonction Lambda.

Pour appeler votre fonction Lambda localement

1. Dans votre ligne de commande, à partir du répertoire du projet `sam-app`, exécutez ce qui suit :

```
$ sam local invoke
```

2. La CLI AWS SAM crée un conteneur Docker local et invoque votre fonction. Voici un exemple de résultat :

```
$ 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\"}"}
```

Pour héberger votre API localement

1. Dans votre ligne de commande, à partir du répertoire du projet `sam-app`, exécutez ce qui suit :

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

2. La CLI AWS SAM crée un conteneur Docker local pour votre fonction Lambda et crée un serveur HTTP local pour simuler le point de terminaison de votre API. Voici un exemple de résultat :

```
$ 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. À l'aide de votre navigateur ou de la ligne de commande, envoyez une requête GET au point de terminaison de votre API local. Voici un exemple d'utilisation de la commande curl :

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

Étape 8 : Supprimez votre application du AWS Cloud

Au cours de cette étape, vous utilisez la AWS SAMCLI sam delete commande pour supprimer votre application du AWS Cloud.

Pour supprimer votre candidature du AWS Cloud

1. Dans votre ligne de commande, à partir du répertoire du projet `sam-app`, exécutez ce qui suit :

```
$ sam delete
```

2. La CLI AWS SAM vous demandera de confirmer. Ensuite, il supprimera le bucket et la CloudFormation pile Amazon S3 de votre application. Voici un exemple de résultat :

```
$ 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
```

Résolution des problèmes

Pour résoudre le problème AWS SAMCLI, voirRésolution des problèmes de la CLI AWS SAM.

En savoir plus

Pour en savoir plus AWS SAM, consultez les ressources suivantes :

- The Complete AWS SAM Workshop : atelier destiné à vous enseigner un bon nombre des principales fonctionnalités d' AWS SAM .

- Sessions avec SAM — Série de vidéos créée par notre équipe AWS Serverless Developer Advocate sur l'utilisation AWS SAM.
- Serverless Land : site qui rassemble les dernières informations, blogs, vidéos, code et ressources d'apprentissage pour AWS sans serveur.

Conversion de fonctions Lambda en applications AWS SAM

Si vous avez une fonction dans la console Lambda que vous souhaitez gérer dans VS Code en tant qu'infrastructure en tant que code (IaC), vous pouvez la transférer vers VS Code, puis la convertir en modèle. AWS SAM Une fois votre fonction convertie en AWS SAM modèle, vous pouvez contrôler le versionnement de votre infrastructure, automatiser les déploiements, déboguer les fonctions à distance et maintenir des environnements cohérents tout au long de votre cycle de développement. Grâce aux fonctionnalités de débogage locales et à distance, vous pouvez également résoudre les problèmes plus efficacement en parcourant votre code, en inspectant les variables et en définissant des points d'arrêt à la fois localement et dans le cloud. AWS

Pour obtenir des instructions sur le déplacement de vos fonctions depuis la console et la configuration du développement local, consultez la section Développement de fonctions Lambda localement avec VS Code dans le guide du AWS Lambda développeur. Suivez ces instructions pour déplacer une fonction Lambda de la console vers VS Code, puis la convertir en AWS SAM modèle. Après cela, vous pouvez :

- Déboguez votre fonction à distance dans le AWS cloud. Pour plus de détails, voir Déboguer à distance les fonctions Lambda avec VS Code dans AWS Lambda le Guide du développeur.
- Gérez les ressources et autorisations associées en tant qu'infrastructure en tant que code. Pour plus de détails, consultez la section Utilisation de Lambda avec une infrastructure en tant que code (IaC) dans le guide du développeur AWS Lambda
- Déployez et mettez à jour votre application à l'aide du AWS SAMCLI. Pour en savoir plus, consultez AWS SAM CLI.
- Concevez visuellement votre architecture sans serveur à l'aide d'Infrastructure Composer. Pour plus de détails, reportez-vous au guide du développeur d'Infrastructure Composer.

En savoir plus

Pour en savoir plus AWS SAM, consultez les ressources suivantes :

- The Complete AWS SAM Workshop : atelier destiné à vous enseigner un bon nombre des principales fonctionnalités d' AWS SAM .
- Sessions avec SAM — Série de vidéos créée par notre équipe AWS Serverless Developer Advocate sur l'utilisation AWS SAM.
- Serverless Land : site qui rassemble les dernières informations, blogs, vidéos, code et ressources d'apprentissage pour AWS sans serveur.

Comment utiliser AWS Serverless Application Model (AWS SAM)

Les principaux outils que vous utilisez pour développer votre application sont le AWS SAM modèle AWS SAMCLIet le AWS SAM projet (qui est le répertoire de votre projet d'application). Vous utilisez ces outils pour :

1. Développez votre application(cela inclut l'initialisation de votre application, la définition de vos ressources et la création de votre application).
2. Testez votre application.
3. Déboguez votre application.
4. Déployez votre application et vos ressources.
5. Surveillez votre application.

AWS SAM crée votre AWS SAM projet une fois que vous avez exécuté la sam init commande et terminé le flux de travail suivant. Vous définissez votre application sans serveur en ajoutant du code à votre AWS SAM projet. Bien que votre AWS SAM projet se compose d'un ensemble de fichiers et de dossiers, le fichier le plus important qu'il contient est votre AWS SAM modèle (nommé`template.yaml`). Dans ce modèle, vous écrivez votre code pour exprimer les ressources, les mappages de sources d'événements et les autres propriétés qui définissent votre application sans serveur.

AWS SAMCLIIl contient un référentiel de commandes que vous utilisez dans votre AWS SAM projet. Plus précisément, AWS SAMCLI c'est ce que vous utilisez pour créer, transformer, déployer, déboguer, empaqueter, initialiser et synchroniser votre AWS SAM projet. En d'autres termes, c'est ce que vous utilisez pour transformer votre AWS SAM projet en application sans serveur.

Pour plus de détails sur ces outils, consultez les chapitres suivants :

AWS SAMCLI- Découvrez comment l' AWS SAMCLIutiliser pour créer et exécuter vos applications sans serveur.

AWS SAM projet et AWS SAM modèle - Découvrez la structure AWS SAM du projet et comment utiliser le AWS SAM modèle pour définir votre application sans serveur.

AWS SAM CLI

AWS Serverless Application Model L'interface de ligne de commande (AWS SAMCLI) est un outil de ligne de commande pour le développement local et le test d'applications sans serveur. Vous AWS SAMCLI permet de créer, de transformer, de déployer, de déboguer, d'empaqueter, d'initialiser et de synchroniser vos applications sans serveur localement avant de les déployer dans le cloud.

AWS SAMCLIfonctionne avec des applications sans serveur définies à l'aide de différents frameworks et outils d'infrastructure en tant que code (IaC), avec différents niveaux de support :

- AWS SAM modèles : fournit un support natif avec l'ensemble complet des fonctionnalités, y compris les fonctionnalités de test local, de débogage, d'empaquetage et de déploiement.
- AWS Applications CDK : prend en charge le test local des fonctions Lambda une fois que vous avez synthétisé l'application CDK en modèles CloudFormation à AWS l'aide de la commande cdk synth.
- CloudFormation modèles : offre une compatibilité directe car elle AWS SAM s'étend CloudFormation et prend en charge les ressources sans serveur définies dans les CloudFormation modèles standard.
- Applications Terraform — Fournit un support limité pour la création et les tests locaux de fonctions Lambda. Vous oblige à générer des artefacts de AWS SAM modèle qui représentent vos fonctions Lambda définies par Terraform.

Pour bénéficier d'une prise en charge complète des fonctionnalités et d'une expérience de développement rationalisée, nous vous recommandons d'utiliser des AWS SAM modèles natifs.

Rubriques

- Comment les commandes de la CLI AWS SAM sont documentées
- Configuration de la CLI AWS SAM
- AWS SAMCLIcommandes de base
- Tests locaux avec AWS SAMCLI

Comment les commandes de la CLI AWS SAM sont documentées

Les commandes de la CLI AWS SAM sont documentées en utilisant le format suivant :

- Invite : l'invite Linux est documentée par défaut et s'affiche sous la forme (`$ `). Pour les commandes spécifiques à Windows, (`> `) est utilisé comme invite. N'incluez pas le symbole d'invite lorsque vous saisissez des commandes.
- Répertoire : lorsque des commandes doivent être saisies depuis un répertoire spécifique, le nom de répertoire s'affiche sous le symbole d'invite.
- Entrée utilisateur : le texte de commande que vous saisissez en ligne de commande est indiqué sous la forme **`user input`**.
- Texte remplaçable : le texte variable, tel que les noms de fichiers et les paramètres, est formaté comme suit. *`replaceable text`* Dans les commandes sur plusieurs lignes ou les commandes dans lesquelles une saisie clavier spécifique est nécessaire, les commandes clavier peuvent également être indiquées sous la forme de texte remplaçable. Par exemple, *`ENTER`*.
- Sortie : la sortie renvoyée en réponse à la commande est indiquée sous la forme `computer output`.

Voici un exemple de sortie de la commande `sam deploy` et de la sortie :

```
$ 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` est la commande que vous saisissez dans la ligne de commande.
2. **`sam deploy --guided --template`** doit être fourni tel quel.
3. *`template.yaml`* peut être remplacé par votre nom de fichier spécifique.
4. La sortie commence à `Configuring SAM deploy`.
5. Dans la sortie, *`ENTER`* et *`y`* indiquez les valeurs remplaçables que vous fournissez.

Configuration de la CLI AWS SAM

L'un des avantages AWS SAM est qu'il optimise le temps du développeur en supprimant les tâches répétitives. AWS SAMCLIinclut un fichier de configuration nommé `samconfig` à cet effet. Par défaut, aucune configuration n' AWS SAMCLIest nécessaire, mais vous pouvez mettre à jour votre fichier de configuration pour vous permettre d'exécuter des commandes avec moins de paramètres en autorisant plutôt AWS SAM à référencer vos paramètres personnalisés dans votre fichier de configuration. Les exemples présentés dans le tableau suivant montrent comment optimiser vos commandes :

Original	Optimisé avec **`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

AWS SAMCLIFournit un ensemble de commandes pour aider les développeurs à créer, développer et déployer des applications sans serveur. Chacune de ces commandes est configurable avec des indicateurs facultatifs en fonction des préférences de l'application et du développeur. Pour plus d'informations, consultez le AWS SAMCLIcontenu dans GitHub

Les rubriques de cette section vous montrent comment créer Fichier de configuration CLI AWS SAM et personnaliser ses paramètres par défaut afin d'optimiser le temps de développement de votre application sans serveur.

Rubriques

- Comment créer votre fichier de configuration (le samconfig fichier)
- Configuration des paramètres de projet
- Configuration des informations d'identification et des paramètres de base

Comment créer votre fichier de configuration (le **`samconfig`** fichier)

Le fichier AWS SAMCLI de configuration (nom de fichier`samconfig`) est un fichier texte qui utilise généralement la structure TOML, mais qui peut également être au format YAML. Lorsque vous utilisez un modèle de démarrage AWS rapide, ce fichier est créé lorsque vous exécutez la sam init commande. Vous pouvez mettre à jour ce fichier lorsque vous déployez une application à l'aide de la sam deploy -\-guided commande.

Une fois le déploiement terminé, le `samconfig` fichier contient un profil nommé `default` si vous avez utilisé les valeurs par défaut. Lorsque vous réexécutez la deploy commande, AWS SAM applique les paramètres de configuration enregistrés à partir de ce profil.

L'avantage du `samconfig` fichier est qu'il AWS SAM stocke les paramètres de configuration pour toutes les autres commandes disponibles en plus de la commande de déploiement. Au-delà de ces valeurs créées lors d'un nouveau déploiement, vous pouvez définir un certain nombre d'attributs dans le `samconfig` fichier afin de simplifier d'autres aspects du flux de travail des développeurs AWS SAMCLI.

Configuration des paramètres de projet

Vous pouvez spécifier des paramètres spécifiques au projet, tels que les valeurs des paramètres de AWS SAMCLI commande, dans un fichier de configuration à utiliser avec le. AWS SAMCLI Pour en savoir plus sur ce fichier de configuration, veuillez consulter la section Fichier de configuration CLI AWS SAM.

Utilisation des fichiers de configuration

Les fichiers de configuration sont structurés par environnement, commande et valeur de paramètre. Pour de plus amples informations, veuillez consulter Principes de base relatifs au fichier de configuration.

Pour configurer un nouvel environnement

1. Spécifiez votre nouvel environnement dans votre fichier de configuration.

Voici un exemple de spécification d'un nouvel environnement `prod` :

TOML

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

YAML

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

2. Spécifiez les valeurs des paramètres sous forme de paires clé-valeur dans la section des paramètres du fichier de configuration.

   Voici un exemple de spécification du nom de la pile de votre application pour l'environnement `prod`.

   TOML

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

   YAML

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

3. Utilisez l'option `--config-env` pour spécifier l'environnement à utiliser.

   Voici un exemple :

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

Pour configurer et utiliser des valeurs de paramètres

1. Spécifiez la commande CLI AWS SAM pour laquelle vous souhaitez configurer les valeurs des paramètres. Pour configurer les valeurs des paramètres pour toutes les commandes de la CLI AWS SAM, utilisez l'identificateur `global`.

   Voici un exemple de spécification des valeurs de paramètres pour la commande `sam deploy` de l'environnement `default` :

   TOML

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

   YAML

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

   Voici un exemple de spécification des valeurs de paramètres pour toutes les commandes CLI AWS SAM de l'environnement `default` :

   TOML

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

   YAML

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

2. Vous pouvez également spécifier des valeurs de paramètres et modifier votre fichier de configuration via le flux interactif de la CLI AWS SAM.

   Vous trouverez ci-dessous un exemple du flux interactif `sam deploy --guided` :

```
$ 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
```

Pour de plus amples informations, veuillez consulter Création et modification de fichiers de configuration.

Exemples

Exemple de base TOML

Voici un exemple de fichier de configuration `samconfig.toml` :

```
...
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
```

Exemple de base YAML

Voici un exemple de fichier de configuration `samconfig.yaml` :

```
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
```

Configuration des informations d'identification et des paramètres de base

Utilisez le AWS Command Line Interface (AWS CLI) pour configurer les paramètres de base tels que les AWS informations d'identification, le nom de région par défaut et le format de sortie par défaut. Une fois configurés, vous pouvez utiliser ces paramètres avec la CLI AWS SAM. Pour en savoir plus, consultez les informations suivantes dans le Guide de l'utilisateur AWS Command Line Interface :

- Principes de base de la configuration
- Paramètres des fichiers de configuration et d'informations d'identification
- Profils nommés pour AWS CLI
- Utilisation d'un profil nommé activé par IAM Identity Center

Pour des instructions de configuration rapides, consultez Étape 5 : utilisez le AWS CLI pour configurer les AWS informations d'identification.

AWS SAMCLIcommandes de base

AWS SAMCLIcomporte certaines commandes de base que vous pouvez utiliser pour créer, créer, tester, déployer et synchroniser votre application sans serveur. Le tableau ci-dessous répertorie ces commandes et fournit des liens contenant des informations supplémentaires pour chacune d'entre elles.

Pour obtenir la liste complète des AWS SAMCLI commandes, voirRéférence des commandes CLI AWS SAM.

Commande	Ce qu'il fait	Rubriques en relation
sam build	Prépare une application pour les étapes suivantes du flux de travail du développeur,	• Initiation à la construction avec AWS SAM • sam build

Commande	Ce qu'il fait	Rubriques en relation
	telles que les tests locaux ou le déploiement AWS dans le cloud.	
sam deploy	Déploie une application dans le AWS cloud à l'aide CloudFormation de.	• Présentation du déploiement avec AWS SAM • sam deploy
sam init	Fournit des options pour initialiser et créer une nouvelle application sans serveur.	• Créez votre application dans AWS SAM • sam init
sam local	Fournit des sous-commandes pour tester vos applications sans serveur localement.	• Présentation des tests à l'aide de la sam local commande • sam local generate-event • sam local invoke • sam local start-api • sam local start-lambda
sam remote invoke	Permet d'interagir avec les AWS ressources prises en charge dans le AWS cloud.	• Présentation des tests dans le cloud avec sam remote invoke • sam remote invoke
sam remote test-event	Permet d'accéder à des événements de test partageables pour vos fonctions AWS Lambda et de les gérer.	• Présentation des tests dans le cloud avec sam remote test-event • sam remote test-event
sam sync	Fournit des options permettant de synchroniser rapidement les modifications apportées aux applications locales dans le AWS cloud.	• Présentation de l'utilisation sam sync de la synchronisation avec AWS Cloud • sam sync

Tests locaux avec AWS SAMCLI

AWS SAMCLICela vous permet de tester des applications sans serveur localement sur différents outils d'infrastructure en tant que code (IaC). Ce guide explique comment utiliser le AWS SAMCLI pour les tests locaux avec différents frameworks IaC.

Les principaux avantages de l'utilisation AWS SAMCLI pour les tests locaux sont les suivants :

- Développement rapide : testez les modifications du code sans les déployer sur AWS
- Rentabilité - Développez et testez sans frais AWS
- Fonctionnalité hors ligne : travaillez sur vos applications sans connexion Internet
- Débogage simplifié : parcourez le code de fonction Lambda localement à l'aide d'un débogueur
- Tests réalistes : testez vos applications à l'aide de l'émulation locale de services AWS

Testez AWS SAM les applications localement

Pour plus d'informations sur le test des applications définies à l'aide de AWS SAM modèles, voir Tester et déboguer des applications sans serveur dans ce guide.

Testez CloudFormation les modèles localement

Pour utiliser les fonctionnalités AWS SAMCLI de test locales de l' CloudFormation outil, ajoutez la AWS SAM transformation à votre CloudFormation modèle. Pour plus d'informations, consultez l'anatomie du AWS SAM modèle pour :

- Ajouter une AWS SAM transformation aux CloudFormation modèles
- Comprendre la compatibilité des modèles
- Exploration de la syntaxe des ressources sans serveur

Testez les applications AWS CDK localement

Vous pouvez utiliser le AWS SAMCLI pour tester les applications AWS CDK localement après avoir synthétisé l'application AWS CDK en CloudFormation modèles à l'aide de la commande cdk synth.

Pour plus d'informations, consultez la section Tester et créer localement des applications AWS CDK AWS SAMCLI à l'aide du guide du développeur du AWS CDK. Cela inclut notamment les éléments suivants :

- Commencer à tester localement
- Tester les applications AWS CDK en local avec AWS SAM

Testez les applications Terraform localement

Il AWS SAMCLI prend en charge les projets Terraform et Terraform Cloud. Vous pouvez l'utiliser pour effectuer le débogage et le test locaux des fonctions et des couches Lambda, ainsi que des protocoles HTTP et REST d'Amazon API Gateway. APIs

Pour configurer votre environnement et en savoir plus sur toutes les fonctionnalités disponibles, consultez le support Terraform dans ce guide. Cela inclut notamment les éléments suivants :

- Commencer à utiliser le support Terraform pour AWS SAMCLI
- Utilisation du AWS SAMCLI avec Terraform pour le débogage et les tests locaux

AWS SAM modèle

Après avoir exécuté la sam init commande et terminé le flux de travail suivant, AWS SAM crée le répertoire de votre projet d'application, qui est votre AWS SAM projet. Vous définissez votre application sans serveur en ajoutant du code à votre AWS SAM projet. Bien que votre AWS SAM projet se compose d'un ensemble de fichiers et de dossiers, le fichier sur lequel vous travaillez principalement est votre AWS SAM modèle (nommé`template.yaml`). Dans ce modèle, vous écrivez le code pour exprimer les ressources, les mappages de sources d'événements et les autres propriétés qui définissent votre application sans serveur.

> **Note**
>
> Un élément clé du AWS SAM modèle est la spécification du AWS SAM modèle. Cette spécification fournit une syntaxe abrégée qui, par rapport à CloudFormation, vous permet d'utiliser moins de lignes de code pour définir les ressources, les mappages de sources d'événements, les autorisations et les autres propriétés de votre application sans serveur. APIs

Cette section explique comment utiliser les sections du AWS SAM modèle pour définir les types de ressources, les propriétés des ressources, les types de données, les attributs des ressources, les fonctions intrinsèques et les extensions API Gateway.

AWS SAM les modèles sont une extension des CloudFormation modèles, avec des types de syntaxe uniques qui utilisent une syntaxe abrégée avec moins de lignes de code que. CloudFormation Cela accélère votre développement lorsque vous créez une application sans serveur. Pour plus d'informations, consultez AWS SAM ressources et propriétés. Pour la référence complète des CloudFormation modèles, voir Référence des CloudFormation modèles dans le guide de AWS CloudFormation l'utilisateur.

Lors du développement, vous trouverez souvent utile de diviser le code de votre application en fichiers distincts afin de mieux l'organiser et la gérer. Un exemple simple en est l'utilisation d'un fichier distinct pour votre code de AWS Lambda fonction plutôt que d'avoir ce code dans votre AWS SAM modèle. Pour ce faire, organisez le code de votre fonction Lambda dans un sous-répertoire de votre projet et référencez son chemin local dans votre AWS Serverless Application Model modèle ().AWS SAM

Rubriques

- AWS SAM anatomie du modèle
- AWS SAM ressources et propriétés
- CloudFormation Ressources générées pour AWS SAM
- Attributs de ressources pris en charge par AWS SAM
- Extensions API Gateway pour AWS SAM
- Fonctions intrinsèques pour AWS SAM

AWS SAM anatomie du modèle

Un fichier AWS SAM modèle suit de près le format d'un fichier CloudFormation modèle, qui est décrit dans la section Anatomie du modèle dans le guide de AWS CloudFormation l'utilisateur. Les principales différences entre les fichiers AWS SAM modèles et les fichiers CloudFormation modèles sont les suivantes :

- Déclaration de transformation. La déclaration `Transform: AWS::Serverless-2016-10-31` est requise pour les fichiers de modèle AWS SAM . Cette déclaration identifie un fichier CloudFormation modèle en tant que fichier AWS SAM modèle. Pour plus d'informations sur les transformations, consultez Transformation dans ce Guide de l'utilisateur AWS CloudFormation .
- Section Globales. La `Globals` section est unique à AWS SAM. Il définit les propriétés communes à toutes vos fonctions sans serveur et APIs. Toutes les `AWS::Serverless::StateMachine` ressources `AWS::Serverless::Function` `AWS::Serverless::ApiAWS::Serverless::CapacityProvider`,`AWS::Serverless::HttpApi`,A et héritent des propriétés définies dans la `Globals` section. Pour plus d'informations sur cette section, consultez Section « Globales » du modèle AWS SAM.
- Section Ressources. Dans les AWS SAM modèles, la `Resources` section peut contenir une combinaison de CloudFormation ressources et de AWS SAM ressources. Pour plus d'informations sur les CloudFormation ressources, consultez la référence aux types de AWS ressources et de propriétés dans le guide de AWS CloudFormation l'utilisateur. Pour plus d'informations sur AWS SAM les ressources, consultezAWS SAM ressources et propriétés.

Toutes les autres sections d'un fichier AWS SAM modèle correspondent à la section du fichier CloudFormation modèle portant le même nom.

YAML

L'exemple suivant illustre un fragment de modèle au format YAML.

```
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
```

Sections de modèle

AWS SAM les modèles peuvent inclure plusieurs sections principales. Seules les sections `Transform` et `Resources` sont requises.

Vous pouvez inclure les sections d'un modèle dans n'importe quel ordre. Toutefois, si vous utilisez des extensions de langue, vous devez les ajouter `AWS::LanguageExtensions` avant la transformation sans serveur (c'est-à-dire avant`AWS::Serverless-2016-10-31`) comme indiqué dans l'exemple suivant :

```
Transform:
```

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

Lorsque vous créez votre modèle, il peut être utile d'utiliser l'ordre logique indiqué dans la liste suivante. Cela est dû au fait que les valeurs d'une section peuvent faire référence à des valeurs d'une section précédente.

Transformation (obligatoire)

Pour les AWS SAM modèles, vous devez inclure cette section avec une valeur de`AWS::Serverless-2016-10-31`.

Les transformations supplémentaires sont facultatives. Pour plus d'informations sur les transformations, consultez Transformation dans ce Guide de l'utilisateur AWS CloudFormation .

Globales (facultatif)

Propriétés communes à toutes vos fonctions sans serveur et tables simples. APIs Toutes les `AWS::Serverless::StateMachine` ressources `AWS::Serverless::Function` `AWS::Serverless::Api``AWS::Serverless::CapacityProvider`,`AWS::Serverless::HttpApi` et héritent des propriétés définies dans la `Globals` section.

Cette section est propre à AWS SAM. Il n'y a pas de section correspondante dans les modèles CloudFormation .

Description (facultative)

Chaîne de texte qui décrit le modèle.

Cette section correspond directement à la `Description` section des CloudFormation modèles.

Métadonnées (facultatives)

Objets qui fournissent des informations supplémentaires sur le modèle.

Cette section correspond directement à la `Metadata` section des CloudFormation modèles.

Paramètres (facultatifs)

Les valeurs à transmettre au modèle lors de l'exécution (lorsque vous créez ou mettez à jour une pile). Vous pouvez faire référence aux paramètres dans les sections `Resources` et `Outputs` du modèle. Les objets déclarés dans la section `Parameters` provoquent la commande sam deploy --guided pour présenter des invites supplémentaires à l'utilisateur.

Les valeurs transmises à l'aide du `--parameter-overrides` paramètre de la `sam deploy` commande, ainsi que les entrées du fichier de configuration, ont priorité sur les entrées du fichier modèle. AWS SAM Pour plus d'informations sur la commande `sam deploy`, consultez sam deploy dans la AWS SAM référence des commandes de la CLI . Pour plus d'informations sur le fichier de configuration, consultez Fichier de configuration CLI AWS SAM.

Mappages (facultatifs)

Mappage de clés et de valeurs associées que vous pouvez utiliser pour spécifier des valeurs de paramètres conditionnelles, comme pour une table de recherche. Pour associer une clé à une valeur correspondante, utilisez la fonction intrinsèque `Fn::FindInMap` dans les sections `Resources` et `Outputs`.

Cette section correspond directement à la `Mappings` section des CloudFormation modèles.

Conditions (facultatives)

Conditions déterminant si certaines ressources sont créées ou si une valeur est attribuée à certaines propriétés de ressources pendant la création ou la mise à jour de la pile. Par exemple, vous pouvez créer une ressource de manière conditionnelle, laquelle varie selon que la pile est destinée à un environnement de production ou de test.

Cette section correspond directement à la `Conditions` section des CloudFormation modèles.

Ressources (obligatoires)

Les ressources de la pile et leurs propriétés, telles qu'une instance Amazon Elastic Compute Cloud (Amazon EC2) ou un bucket Amazon Simple Storage Service (Amazon S3). Vous pouvez faire référence à des ressources dans les sections `Resources` et `Outputs` du modèle.

Cette section est semblable à la section `Resources` des modèles CloudFormation . Dans les AWS SAM modèles, cette section peut contenir des AWS SAM ressources en plus des CloudFormation ressources.

Sorties (facultatives)

Les valeurs qui sont renvoyées chaque fois que vous affichez les propriétés de votre pile. Par exemple, vous pouvez déclarer une sortie pour le nom d'un compartiment S3, puis appeler la commande `aws cloudformation describe-stacks` AWS Command Line Interface (AWS CLI) pour afficher le nom.

Cette section correspond directement à la section `Outputs` des modèles CloudFormation .

Étapes suivantes

Pour télécharger et déployer un exemple d'application sans serveur contenant un fichier AWS SAM modèle, consultez Commencer avec AWS SAM et suivez les instructions figurant dansTutoriel : Déployer une application Hello World avec AWS SAM.

Section « Globales » du modèle AWS SAM

Les ressources que vous déclarez dans un AWS SAM modèle ont parfois des configurations communes. Par exemple, vous pouvez avoir une application avec plusieurs ressources `AWS::Serverless::Function` ayant des configurations `Runtime`, `Memory`, `VPCConfig`, `Environment`, et `Cors` identiques. Au lieu de dupliquer ces informations dans chaque ressource, vous pouvez les déclarer une fois dans la section `Globals` et laisser vos ressources en hériter.

La `Globals` section prend en charge les types de AWS SAM ressources suivants :

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

Exemple :

```
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
```

Dans cet exemple, `HelloWorldFunction` et `ThumbnailFunction` utilisent toutes deux « nodejs12.x » pour `Runtime`, « 180 » secondes pour `Timeout`, et " index.handler " pour `Handler`. `HelloWorldFunction` ajoute la variable d'environnement MESSAGE, en plus du TABLE_NAME hérité.`ThumbnailFunction`hérite de toutes les propriétés `Globals` et ajoute une source d'événement API.

Ressources et propriétés prises en charge

AWS SAM prend en charge les ressources et propriétés suivantes.

```
Globals:
  Api:
    AccessLogSetting:
    Auth:
    BinaryMediaTypes:
    CacheClusterEnabled:
    CacheClusterSize:
    CanarySetting:
    Cors:
    DefinitionUri:
    Domain:
    EndpointConfiguration:
    GatewayResponses:
    MethodSettings:
    MinimumCompressionSize:
    Name:
    OpenApiVersion:
    PropagateTags:
    TracingEnabled:
    Variables:
```

```
  CapacityProvider:
    InstanceRequirements:
    KmsKeyArn:
    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:
```

> **Note**
>
> Toutes les ressources et propriétés qui ne sont pas incluses dans la liste précédente ne sont pas prises en charge. Voici quelques raisons pour lesquelles elles ne sont pas prises en charge : 1) elles ouvrent des problèmes de sécurité potentiels, ou 2) elles rendent le modèle difficile à comprendre.

Incident APIs

AWS SAM crée une valeur implicite APIs lorsque vous déclarez une API dans la `Events` section. Vous pouvez l'utiliser `Globals` pour remplacer toutes les propriétés APIs implicites.

Propriétés substituables

Les ressources peuvent remplacer les propriétés que vous déclarez dans la section `Globals`. Par exemple, vous pouvez ajouter de nouvelles variables à une carte de variables d'environnement ou remplacer les variables déclarées globalement. Mais la ressource ne peut pas supprimer une propriété spécifiée dans la section `Globals`.

De manière plus générale, la section `Globals` déclare les propriétés que toutes vos ressources partagent. Certaines ressources peuvent fournir de nouvelles valeurs pour les propriétés déclarées globalement, mais elles ne peuvent pas les supprimer. Si certaines ressources utilisent une propriété mais que d'autres ne le font pas, alors vous ne devez pas les déclarer dans la section `Globals`.

Les sections suivantes décrivent comment fonctionne le remplacement pour différents types de données.

Les types de données primitives sont remplacés

Les types de données primitives peuvent être des chaînes, des nombres, des valeurs booléennes, etc.

La valeur spécifiée dans la section `Resources` remplace la valeur dans la section`Globals`.

Exemple :

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

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

L'`Runtime` pour `MyFunction` est définie sur `python3.9`.

Les cartes sont fusionnées

Les cartes sont également désignées sous le nom de dictionnaires ou de collections de paires valeur-clé.

Les entrées de carte dans la section `Resources` sont fusionnées avec des entrées de carte globales. En cas de doublons, la section `Resource` remplace l'entrée de la section `Globals`.

Exemple :

```
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
```

Les variables d'environnement de `MyFunction` sont définis comme suivant :

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

Les listes sont additives

Les listes sont également désignées sous le nom de tableaux.

Les entrées de liste dans la section `Globals` sont ajoutées à la liste dans la section `Resources`.

Exemple :

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

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

Les `SecurityGroupIds` pour les `VpcConfig` de `MyFunction` sont définis comme suivant :

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

AWS SAM ressources et propriétés

Cette section décrit les types de ressources et de propriétés spécifiques à. AWS SAM Vous définissez ces ressources et propriétés à l'aide de la AWS SAM syntaxe abrégée. AWS SAM prend également en charge les types de CloudFormation ressources et de propriétés. Pour obtenir des informations de référence sur tous les types de AWS ressources et de propriétés CloudFormation ainsi que sur le AWS SAM support, voir la référence aux types de AWS ressources et de propriétés dans le guide de AWS CloudFormation l'utilisateur.

Rubriques

- AWS::Serverless::Api
- AWS::Serverless::Application
- AWS::Serverless::CapacityProvider
- AWS::Serverless::Connector
- AWS::Serverless::Function
- AWS::Serverless::GraphQLApi
- AWS::Serverless::HttpApi
- AWS::Serverless::LayerVersion
- AWS::Serverless::SimpleTable
- AWS::Serverless::StateMachine

AWS::Serverless::Api

Crée un ensemble de ressources et de méthodes Amazon API Gateway qui peuvent être appelées via les points de terminaison HTTPS.

Il n'est pas nécessaire d'ajouter explicitement une AWS::Serverless::Api ressource à un modèle de définition d'application AWS sans serveur. Une ressource de ce type est implicitement créée à partir de l'union des événements Api définis su les ressources AWS::Serverless::Function définies dans le modèle qui ne font pas référence à une ressource AWS::Serverless::Api.

Une AWS::Serverless::Api ressource doit être utilisée pour définir et documenter l'API utilisée OpenApi, ce qui permet de mieux configurer les ressources Amazon API Gateway sous-jacentes.

Nous vous recommandons d'utiliser CloudFormation des hooks ou des politiques IAM pour vérifier que les ressources API Gateway sont associées à des autorisateurs afin de contrôler l'accès à celles-ci.

Pour plus d'informations sur l'utilisation CloudFormation des hooks, consultez la section Enregistrement des hooks dans le guide de l'utilisateur de la CloudFormation CLI et dans le apigw-enforce-authorizer GitHub référentiel.

Pour plus d'informations sur l'utilisation de politiques IAM, veuillez consulter Exiger que les routes d'API disposent d'une autorisation dans le Guide du développeur API Gateway.

> **Note**
>
> Lorsque vous déployez vers AWS CloudFormation, vos AWS SAM ressources sont AWS SAM transformées en CloudFormation ressources. Pour de plus amples informations, veuillez consulter CloudFormation Ressources générées pour AWS SAM.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

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
  EndpointConfiguration: EndpointConfiguration
  FailOnWarnings: Boolean
  GatewayResponses: Map
  MergeDefinitions: Boolean
  MethodSettings: MethodSettings
  MinimumCompressionSize: Integer
  Mode: String
  Models: Map
  Name: String
  OpenApiVersion: String
  PropagateTags: Boolean
  Policy: JSON
  StageName: String
  Tags: Map
  TracingEnabled: Boolean
  Variables: Map
```

Propriétés

`AccessLogSetting`

Configure les réglages d'accès au journal pour une étape.

Type : AccessLogSetting

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `AccessLogSetting` propriété d'une `AWS::ApiGateway::Stage` ressource.

`AlwaysDeploy`

Déploie toujours l'API, même si aucune modification n'a été détectée.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`ApiKeySourceType`

Source de la clé API pour les demandes de relevé en fonction d'un plan d'utilisation. Les valeurs valides sont `HEADER` et `AUTHORIZER`.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `ApiKeySourceType` propriété d'une `AWS::ApiGateway::RestApi` ressource.

`Auth`

Configurez le mécanisme d'autorisation utilisé pour contrôler l'accès à votre API API Gateway.

Pour plus d'informations sur la configuration de l'accès à l' AWS SAM aide deContrôlez l'accès aux API avec votre AWS SAM modèle. Pour un exemple montrant comment remplacer un autorisateur global, voir. the section called “Remplacer un mécanisme d'autorisation global pour votre API REST Amazon API Gateway”

Type : ApiAuth

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`BinaryMediaTypes`

Liste des types MIME que votre API pourrait renvoyer. Utilisez-le pour activer le support binaire pourAPIs.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est similaire à celle `BinaryMediaTypes` d'une `AWS::ApiGateway::RestApi` ressource. La liste des BinaryMediaTypes est ajoutée à la fois à la CloudFormation ressource et au document OpenAPI.

`CacheClusterEnabled`

Indique si la mise en cache est activée pour l'étape. Pour mettre en cache les réponses, vous devez également définir `CachingEnabled` sur `true` sous `MethodSettings`.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `CacheClusterEnabled` propriété d'une `AWS::ApiGateway::Stage` ressource.

`CacheClusterSize`

Taille du cluster de cache de l'environnement.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `CacheClusterSize` propriété d'une `AWS::ApiGateway::Stage` ressource.

`CanarySetting`

Configurez un paramètre Canary à une étape d'un déploiement régulier.

Type : CanarySetting

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `CanarySetting` propriété d'une `AWS::ApiGateway::Stage` ressource.

`Cors`

Gérez le partage de ressources entre origines (CORS) pour l'ensemble de votre API Gateway. APIs Spécifiez le domaine à autoriser en tant que chaîne ou spécifiez un dictionnaire avec une configuration Cors supplémentaire.

> **Note**
>
> CORS nécessite AWS SAM de modifier votre définition OpenAPI. Créez une définition OpenAPI intégrée dans `DefinitionBody` le pour activer CORS.

Pour plus d'informations, consultez Activation de CORS pour une ressource API REST dans API Gateway dans le Guide du développeur API Gateway.

Type : Chaîne | CorsConfiguration

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`DefinitionBody`

Spécification OpenAPI qui décrit votre API. Si ni `DefinitionUri`, ni `DefinitionBody` sont spécifiés, SAM générera un `DefinitionBody` pour vous en fonction de la configuration de votre modèle.

Pour faire référence à un fichier OpenAPI local qui définit votre API, utilisez la macro `AWS::Include` transform. Pour en savoir plus, veuillez consulter la section Comment AWS SAM télécharge les fichiers locaux.

Type : JSON

Obligatoire : non

CloudFormation compatibilité : cette propriété est similaire à celle Body d'une `AWS::ApiGateway::RestApi` ressource. Si certaines propriétés sont fournies, le contenu peut être inséré ou modifié dans le DefinitionBody avant d'être transmis à CloudFormation. Les propriétés comprennent un `Auth`, `BinaryMediaTypes`, `Cors`, `GatewayResponses`, `Models`, et un `EventSource` de type Api activée pour une `AWS::Serverless::Function` correspondante.

`DefinitionUri`

Amazon S3 Uri, chemin d'accès au fichier local ou objet d'emplacement du document OpenAPI définissant l'API. L'objet Amazon S3 référencé par cette propriété doit être un fichier OpenAPI valide. Si ni `DefinitionUri`, ni `DefinitionBody` sont spécifiés, SAM générera un `DefinitionBody` pour vous en fonction de la configuration de votre modèle.

Si un chemin d'accès de fichier local est fourni, le modèle doit passer par le flux comprenant la propriété `sam deploy` ou `sam package`, afin que la définition soit correctement transformée.

Les fonctions intrinsèques ne sont pas prises en charge dans OpenApi les fichiers externes référencés par`DefinitionUri`. Utilisez plutôt la `DefinitionBody` propriété avec la transformation Include pour importer une OpenApi définition dans le modèle.

Type : Chaîne | ApiDefinition

Obligatoire : non

CloudFormation compatibilité : cette propriété est similaire à celle `BodyS3Location` d'une `AWS::ApiGateway::RestApi` ressource. Les propriétés imbriquées d'Amazon S3 sont nommées différemment.

`Description`

Une description de la ressource API.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Description` propriété d'une `AWS::ApiGateway::RestApi` ressource.

`DisableExecuteApiEndpoint`

Spécifie si les clients peuvent appeler votre API à l'aide du point de terminaison `execute-api` par défaut. Par défaut, les clients peuvent appeler votre API avec l'`https://{api_id}.execute-api.{region}.amazonaws.com` par défaut. Pour obliger les clients à utiliser un nom de domaine personnalisé pour appeler votre API, spécifiez `True`.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est similaire à celle `DisableExecuteApiEndpoint` d'une `AWS::ApiGateway::RestApi` ressource. Il est transmis directement à la propriété `disableExecuteApiEndpoint` d'une extension `x-amazon-apigateway-endpoint-configuration`, qui est ajoutée à la propriété `Body` d'une ressource `AWS::ApiGateway::RestApi`.

`Domain`

Configure un domaine personnalisé pour cette API Gateway.

Type : DomainConfiguration

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`EndpointConfiguration`

Type de point de terminaison d'une API REST.

Type : EndpointConfiguration

Obligatoire : non

CloudFormation compatibilité : cette propriété est similaire à celle `EndpointConfiguration` d'une `AWS::ApiGateway::RestApi` ressource. Les propriétés de configuration imbriquées sont nommées différemment.

`FailOnWarnings`

Indique si la création d'API doit être annulée (`true`) ou non (`false`) lorsqu'un avertissement est rencontré. La valeur par défaut est `false`.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `FailOnWarnings` propriété d'une `AWS::ApiGateway::RestApi` ressource.

`GatewayResponses`

Configure les réponses de passerelle pour une API. Les réponses de passerelle sont des réponses renvoyées par API Gateway, directement ou via l'utilisation des mécanismes d'autorisation Lambda. Pour plus d'informations, consultez la documentation de l' OpenApi extension Api Gateway pour Gateway Responses.

Type: carte (map)

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`MergeDefinitions`

AWS SAM génère une OpenAPI spécification à partir de la source d'événements de votre API. Spécifiez `true` de le AWS SAM fusionner dans la OpenAPI spécification en ligne définie dans votre `AWS::Serverless::Api` ressource. Spécifiez la valeur `false` pour ne pas fusionner.

`MergeDefinitions` nécessite la propriété `DefinitionBody` pour que `AWS::Serverless::Api` soit définie. `MergeDefinitions` n'est pas compatible avec la propriété `DefinitionUri` pour `AWS::Serverless::Api`.

Valeur par défaut : `false`

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`MethodSettings`

Configure tous les paramètres de l'étape API, y compris la journalisation, les métriques, le CacheTL, la limitation.

Type : liste de propriétés MethodSetting

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `MethodSettings` propriété d'une `AWS::ApiGateway::Stage` ressource.

`MinimumCompressionSize`

Autoriser la compression des corps de réponse en fonction de l'en-tête Accept-Encoding du client. La compression est déclenchée lorsque la taille du corps de la réponse est supérieure ou égale au seuil configuré. Le seuil de taille maximale du corps est de 10 Mo (10 485 760 octets). - Les types de compression suivants sont pris en charge : gzip, deflate et identity.

Type : entier

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `MinimumCompressionSize` propriété d'une `AWS::ApiGateway::RestApi` ressource.

`Mode`

Cette propriété s'applique uniquement lorsque vous utilisez OpenAPI pour définir votre API REST. Le `Mode` détermine comment API Gateway gère les mises à jour des ressources. Pour plus d'informations, consultez la propriété Mode du type de ressource AWS::ApiGateway::RestApi.

Valeurs valides : `overwrite` ou `merge`

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Mode` propriété d'une `AWS::ApiGateway::RestApi` ressource.

`Models`

Les schémas à utiliser par vos méthodes d'API. Ces schémas peuvent être décrits avec JSON ou YAML. Consultez la section Exemples au bas de cette page pour obtenir des exemples de modèles.

Type: carte (map)

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Name`

Nom de la RestApi ressource API Gateway

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Name` propriété d'une `AWS::ApiGateway::RestApi` ressource.

`OpenApiVersion`

Version de OpenApi à utiliser. Cela peut être soit `2.0` pour la spécification Swagger, soit pour l'une des versions OpenApi 3.0, comme. `3.0.1` Pour plus d'informations sur OpenAPI, consultez Spécification OpenAPI.

> **Note**
>
> AWS SAM crée une étape appelée `Stage` par défaut. La définition de cette propriété sur n'importe quelle valeur valide empêchera la création de l'étape `Stage`.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`PropagateTags`

Indiquez s'il faut ou non transmettre les balises de la propriété `Tags` aux ressources AWS::Serverless::Api que vous avez générées. Spécifiez `True` pour la propagation des balises dans vos ressources générées.

Type : valeur booléenne

Obligatoire : non

Par défaut : `False`

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Policy`

Document de politique qui contient les autorisations pour l'API. Pour définir l'ARN de la politique, utilisez la fonction `!Join` intrinsèque avec `""` comme délimiteur et les valeurs `"execute-api:/"` et `"*"`.

Type : JSON

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la propriété Policy d'une `AWS::ApiGateway::RestApi` ressource.

`StageName`

Le nom de l'étape qu'API Gateway utilise en tant que premier segment de chemin d'accès dans l'identificateur de ressource uniforme (URI) appelé.

Pour référencer la ressource de l'étape, utilisez `<api-logical-id>.Stage`. Pour plus d'informations sur le référencement des ressources générées lorsqu'une AWS::Serverless::Api est spécifiée, consultez CloudFormation ressources générées lorsque cela AWS::Serverless::Api est spécifié. Pour des informations générales sur les CloudFormation ressources générées, consultezCloudFormation Ressources générées pour AWS SAM.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est similaire à celle `StageName` d'une `AWS::ApiGateway::Stage` ressource. Elle est requise dans SAM, mais pas dans API Gateway

Informations complémentaires : l'API implicite a un nom d'étape « Prod ».

`Tags`

Un mappage (chaîne à chaîne) qui spécifie les balises à ajouter à cette étape API Gateway. Pour plus de détails sur les clés et les valeurs valides pour les étiquettes, voir l'étiquette Ressource dans le Guide de l'utilisateur AWS CloudFormation .

Type: carte (map)

Obligatoire : non

CloudFormation compatibilité : cette propriété est similaire à celle `Tags` d'une `AWS::ApiGateway::Stage` ressource. La propriété Tags dans SAM se compose de paires Key:Value ; CloudFormation elle consiste en une liste d'objets Tag.

`TracingEnabled`

Indique si le suivi actif avec X-Ray est activé pour cette étape. Pour plus d'informations sur X-Ray, consultez la section Suivi des demandes des utilisateurs vers REST à APIs l'aide de X-Ray dans le guide du développeur d'API Gateway.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `TracingEnabled` propriété d'une `AWS::ApiGateway::Stage` ressource.

`Variables`

Un mappage (chaîne à chaîne) qui définit les variables de l'étape, où le nom de la variable correspond à la clé et où la valeur de la variable correspond à la valeur. Les noms de variables peuvent uniquement utiliser des caractères alphanumériques. Les valeurs doivent respecter l'expression régulière suivante : `[A-Za-z0-9._~:/?#&=,-]+`.

Type: carte (map)

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Variables` propriété d'une `AWS::ApiGateway::Stage` ressource.

Valeurs renvoyées

Réf

Lorsque l'ID logique de cette ressource est fournie à la fonction intrinsèque `Ref`, elle renvoie l'ID de l'API API Gateway sous-jacente.

Pour plus d'informations sur l'utilisation de la fonction `Ref`, consultez `Ref` dans le Guide de l'utilisateur AWS CloudFormation .

Ventilateur : GetAtt

`Fn::GetAtt` renvoie une valeur pour un attribut de ce type indiqué. Voici les attributs disponibles et des exemples de valeurs de retour.

Pour plus d'informations sur l'utilisation de `Fn::GetAtt`, consultez `Fn::GetAtt` dans le Guide de l'utilisateur AWS CloudFormation .

`RootResourceId`

ID de ressource racine d'une ressource `RestApi`, comme `a0bc123d4e`.

Exemples

SimpleApiExample

Un fichier AWS SAM modèle Hello World qui contient une fonction Lambda avec un point de terminaison d'API. Il s'agit d'un fichier AWS SAM modèle complet pour une application sans serveur fonctionnelle.

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

Un extrait de AWS SAM modèle avec une API définie dans un fichier Swagger externe, ainsi que des intégrations Lambda et des configurations CORS. Il s'agit simplement d'une partie d'un fichier AWS SAM modèle présentant une AWS::Serverless::Api définition.

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

Extrait de AWS SAM modèle avec une API qui utilise Amazon Cognito pour autoriser les requêtes adressées à l'API. Il s'agit simplement d'une partie d'un fichier AWS SAM modèle présentant une AWS::Serverless::Api définition.

YAML

```
Resources:
  ApiGatewayApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      Cors: "'*'"
      Auth:
        DefaultAuthorizer: MyCognitoAuthorizer
        Authorizers:
          MyCognitoAuthorizer:
            UserPoolArn:
              Fn::GetAtt: [MyCognitoUserPool, Arn]
```

ApiModelsExample

Extrait AWS SAM de modèle avec une API qui inclut un schéma Models. Il ne s'agit que d'une partie d'un fichier AWS SAM modèle présentant une AWS::Serverless::Api définition avec deux schémas de modèle.

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
```

Exemple de mise en cache

Un fichier AWS SAM modèle Hello World qui contient une fonction Lambda avec un point de terminaison d'API. La mise en cache de l'API est activée pour une ressource et une méthode. Pour plus d'informations sur la mise en cache, veuillez consulter Activation de la mise en cache des API pour améliorer la réactivité dans le Guide du développeur API Gateway.

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}
```

Exemple de domaine personnalisé avec une API privée

Un fichier AWS SAM modèle Hello World qui contient une fonction Lambda avec un point de terminaison d'API mappé à un domaine privé. Le modèle crée une association d'accès au domaine entre un point de terminaison VPC et le domaine privé. Pour plus d'informations, consultez la section Noms de domaine personnalisés pour le domaine privé APIs dans 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

Configurez le mécanisme d'autorisation utilisé pour contrôler l'accès à votre API API Gateway.

Pour plus d'informations et des exemples de configuration de l'accès à l' AWS SAM aide deContrôlez l'accès aux API avec votre AWS SAM modèle.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

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

> **Note**
>
> La `Authorizers` propriété inclutAWS_IAM, mais aucune configuration supplémentaire n'est nécessaireAWS_IAM. Pour obtenir un exemple, consultez AWS IAM.

Propriétés

`AddApiKeyRequiredToCorsPreflight`

Si les propriétés `ApiKeyRequired` et `Cors` sont définies, alors le fait de définir `AddApiKeyRequiredToCorsPreflight` entraînera l'ajout la clé API à la propriété `Options`.

Type : valeur booléenne

Obligatoire : non

Par défaut : `True`

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`AddDefaultAuthorizerToCorsPreflight`

Si les propriétés `DefaultAuthorizer` et `Cors` sont définies, alors le fait de définir `AddDefaultAuthorizerToCorsPreflight`entraînera l'ajout du mécanisme d'autorisation par défaut à la propriété `Options` dans la section OpenAPI.

Type : valeur booléenne

Obligatoire : non

Valeur par défaut : VRAI

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`ApiKeyRequired`

Si la valeur est définie sur « vrai », une clé d'API est requise pour tous les événements d'API. Pour plus d'informations sur les clés API, consultez Création et utilisation de plans d'utilisation avec les clés API dans le Guide du développeur API Gateway.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Authorizers`

Le mécanisme d'autorisation utilisé pour contrôler l'accès à votre API API Gateway.

Pour de plus amples informations, veuillez consulter Contrôlez l'accès aux API avec votre AWS SAM modèle.

Type : CognitoAuthorizer| LambdaTokenAuthorizer| LambdaRequestAuthorizer| AWS_IAM

Obligatoire : non

Par défaut : aucun

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Remarques supplémentaires : SAM ajoute les Autorisateurs à la OpenApi définition d'une API.

`DefaultAuthorizer`

Spécifiez un autorisateur par défaut pour une API API Gateway, qui sera utilisé pour autoriser les appels d'API par défaut.

> **Note**
>
> Si l'API EventSource de la fonction associée à cette API est configurée pour utiliser les autorisations IAM, cette propriété doit être définie sur`AWS_IAM`, sinon une erreur se produira.

Type : chaîne

Obligatoire : non

Par défaut : aucun

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`InvokeRole`

Définit les informations d'identification d'intégration pour toutes les ressources et méthodes sur cette valeur.

`CALLER_CREDENTIALS` mappe avec `arn:aws:iam::`*`:<user>/`*, qui utilise les informations d'identification de l'appelant pour appeler le point de terminaison.

Valeurs valides : `CALLER_CREDENTIALS`, `NONE`, `IAMRoleArn`

Type : chaîne

Obligatoire : non

Par défaut : `CALLER_CREDENTIALS`

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`ResourcePolicy`

Configurez la stratégie de ressources pour toutes les méthodes et chemins d'accès d'une API.

Type : ResourcePolicyStatement

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Informations complémentaires : ce paramètre peut également être défini sur `AWS::Serverless::Function` à l'aide de ApiFunctionAuth. Ceci est requis pour APIs avec`EndpointConfiguration: PRIVATE`.

`UsagePlan`

Configure un plan d'utilisation associé à cette API. Pour plus d'informations sur les plans d'utilisation, consultez Création et utilisation de plans d'utilisation avec les clés API dans le Guide du développeur API Gateway.

Cette AWS SAM propriété génère trois CloudFormation ressources supplémentaires lorsqu'elle est définie : an AWS::ApiGateway::UsagePlanAWS::ApiGateway::UsagePlanKey, an et an

AWS::ApiGateway::ApiKey. Pour plus d'informations sur ce scénario, consultez UsagePlan la propriété est spécifiée. Pour des informations générales sur les CloudFormation ressources générées, consultezCloudFormation Ressources générées pour AWS SAM.

Type : ApiUsagePlan

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

CognitoAuth

Exemple de 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 IAM

AWS Exemple IAM

YAML

```
Auth:
  Authorizers: AWS_IAM
```

ApiUsagePlan

Configure un plan d'utilisation pour une API API Gateway. Pour plus d'informations sur les plans d'utilisation, consultez Création et utilisation de plans d'utilisation avec les clés API dans le Guide du développeur API Gateway.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

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

Propriétés

`CreateUsagePlan`

Détermine comment ce plan d'utilisation est configuré. Les valeurs valides sont `PER_API`, `SHARED` et `NONE`.

`PER_API` crée AWS::ApiGateway::UsagePlan,AWS::ApiGateway::ApiKey, et des ressources AWS::ApiGateway::UsagePlanKey propres à cette API. Ces ressources ont respectivement IDs

une logique de `<api-logical-id>UsagePlan` `<api-logical-id>ApiKey` `<api-logical-id>UsagePlanKey`, et.

`SHARED`les AWS::ApiGateway::UsagePlancréations et AWS::ApiGateway::ApiKeyles AWS::ApiGateway::UsagePlanKeyressources qui sont partagées entre toutes les API figurant également `CreateUsagePlan: SHARED` dans le même AWS SAM modèle. Ces ressources ont respectivement IDs une logique de `ServerlessUsagePlan` `ServerlessApiKeyServerlessUsagePlanKey`, et. Si vous utilisez cette option, nous vous recommandons d'ajouter une configuration supplémentaire pour ce plan d'utilisation sur une seule ressource API afin d'éviter des définitions conflictuelles et un état incertain.

`NONE` désactive la création ou l'association d'un plan d'utilisation avec cette API. Ceci n'est nécessaire que si `SHARED` ou `PER_API` est spécifié dans le paramètre Section « Globales » du modèle AWS SAM.

Valeurs valides: `PER_API`, `SHARED`, et `NONE`

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Description`

Description du plan d'utilisation.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Description` propriété d'une `AWS::ApiGateway::UsagePlan` ressource.

`Quota`

Configure le nombre de demandes que les utilisateurs peuvent soumettre au cours d'un intervalle donné.

Type : QuotaSettings

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Quota` propriété d'une `AWS::ApiGateway::UsagePlan` ressource.

`Tags`

Tableau de balises arbitraires (paires clé-valeur) à associer au plan d'utilisation.

Cette propriété utilise le type de CloudFormation balise.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Tags` propriété d'une `AWS::ApiGateway::UsagePlan` ressource.

`Throttle`

Configure le taux de demandes global (nombre moyen de demandes par seconde), ainsi que la capacité de transmission en mode rafale.

Type : ThrottleSettings

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Throttle` propriété d'une `AWS::ApiGateway::UsagePlan` ressource.

`UsagePlanName`

Nom pour le plan d'utilisation.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `UsagePlanName` propriété d'une `AWS::ApiGateway::UsagePlan` ressource.

Exemples

UsagePlan

Voici un exemple de plan d'utilisation.

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

Définissez un autorisation de groupe d'utilisateurs Amazon Cognito.

Pour plus d'informations et d'exemples, consultez Contrôlez l'accès aux API avec votre AWS SAM modèle.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

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

Propriétés

`AuthorizationScopes`

Liste des étendues d'autorisation pour cet autorisateur.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Identity`

Cette propriété peut être utilisée pour spécifier une `IdentitySource` dans une demande d'autorisation entrante.

Type : CognitoAuthorizationIdentity

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`UserPoolArn`

Peut faire référence à un utilisateur, pool/specify un arn de groupe d'utilisateurs auquel vous souhaitez ajouter cet autorisateur cognito.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

CognitoAuth

Exemple d'authentification Cognito

YAML

```
Auth:
  Authorizers:
    MyCognitoAuth:
      AuthorizationScopes:
        - scope1
        - scope2
      UserPoolArn:
        Fn::GetAtt:
```

```
          - MyCognitoUserPool
          - Arn
      Identity:
        Header: MyAuthorizationHeader
        ValidationExpression: myauthvalidationexpression
```

CognitoAuthorizationIdentity

Cette propriété peut être utilisée pour spécifier un IdentitySource dans une demande entrante pour un autorisateur. Pour plus d'informations, IdentitySource consultez l' OpenApi extension ApiGateway Authorizer.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

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

Propriétés

`Header`

Spécifiez le nom de l'en-tête pour Authorization dans la OpenApi définition.

Type : chaîne

Obligatoire : non

Par défaut : autorisation

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`ReauthorizeEvery`

Période time-to-live (TTL), en secondes, qui indique la durée pendant laquelle API Gateway met en cache les résultats de l'autorisateur. Si vous définissez une valeur supérieure à 0, API

Gateway met en cache les réponses du mécanisme d'autorisation. Par défaut, API Gateway définit cette propriété sur 300. La valeur maximale est 3 600 ou 1 heure.

Type : entier

Obligatoire : non

Par défaut : 300

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`ValidationExpression`

Spécifiez une expression de validation pour valider l'identité entrante.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

CognitoAuthIdentity

YAML

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

LambdaRequestAuthorizer

Configurez un mécanisme d'autorisation Lambda pour contrôler l'accès à votre API à l'aide d'une fonction Lambda.

Pour plus d’informations et d’exemples, consultez Contrôlez l'accès aux API avec votre AWS SAM modèle.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```yaml
DisableFunctionDefaultPermissions: Boolean
FunctionArn: String
FunctionInvokeRole: String
FunctionPayloadType: String
Identity: LambdaRequestAuthorizationIdentity
```

Propriétés

`DisableFunctionDefaultPermissions`

Spécifiez `true` pour AWS SAM empêcher la création automatique d'une `AWS::Lambda::Permissions` ressource afin de fournir des autorisations entre votre `AWS::Serverless::Api` ressource et la fonction Lambda de l'autorisateur.

Valeur par défaut : `false`

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`FunctionArn`

Spécifiez la fonction ARN de la fonction Lambda qui fournit l'autorisation pour l'API.

> **Note**
>
> AWS SAM créera automatiquement une `AWS::Lambda::Permissions` ressource lorsque cela `FunctionArn` est spécifié pour`AWS::Serverless::Api`. La ressource `AWS::Lambda::Permissions` fournit des autorisations entre votre API et la fonction Lambda du mécanisme d'autorisation.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`FunctionInvokeRole`

Ajoute les informations d'identification de l'autorisateur à la OpenApi définition de l'autorisateur Lambda.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`FunctionPayloadType`

Cette propriété peut être utilisée pour définir le type de mécanisme d'autorisation Lambda pour une API.

Valeurs valides : `TOKEN` ou `REQUEST`

Type : chaîne

Obligatoire : non

Par défaut : `TOKEN`

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Identity`

Cette propriété peut être utilisée pour spécifier une `IdentitySource` dans une demande d'autorisation entrante. Cette propriété n'est requise que si la propriété `FunctionPayloadType` est définie sur `REQUEST`.

Type : LambdaRequestAuthorizationIdentity

Obligatoire : Conditionnelle

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

LambdaRequestAuth

YAML

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

LambdaRequestAuthorizationIdentity

Cette propriété peut être utilisée pour spécifier un IdentitySource dans une demande entrante pour un autorisateur. Pour plus d'informations, IdentitySource consultez l' OpenApi extension ApiGateway Authorizer.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

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

Propriétés

`Context`

Convertit les chaînes de contexte données en expressions de mappage de format `context.contextString`.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Headers`

Convertit les en-têtes en chaîne séparée par des virgules d'expressions de mappage de formats `method.request.header.name`.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`QueryStrings`

Convertit les chaînes de requête données en chaîne séparée par des virgules d'expressions de mappage de format `method.request.querystring.queryString`.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`ReauthorizeEvery`

Période time-to-live (TTL), en secondes, qui indique la durée pendant laquelle API Gateway met en cache les résultats de l'autorisateur. Si vous définissez une valeur supérieure à 0, API Gateway met en cache les réponses du mécanisme d'autorisation. Par défaut, API Gateway définit cette propriété sur 300. La valeur maximale est 3 600 ou 1 heure.

Type : entier

Obligatoire : non

Par défaut : 300

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`StageVariables`

Convertit les variables d'une étape donnée en chaîne séparée par des virgules d'expressions de mappage de format `stageVariables.stageVariable`.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

LambdaRequestIdentity

YAML

```
Identity:
  QueryStrings:
    - auth
  Headers:
    - Authorization
  StageVariables:
    - VARIABLE
  Context:
    - authcontext
  ReauthorizeEvery: 100
```

LambdaTokenAuthorizer

Configurez un mécanisme d'autorisation Lambda pour contrôler l'accès à votre API à l'aide d'une fonction Lambda.

Pour plus d'informations et d'exemples, consultez Contrôlez l'accès aux API avec votre AWS SAM modèle.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
DisableFunctionDefaultPermissions: Boolean
FunctionArn: String
FunctionInvokeRole: String
FunctionPayloadType: String
Identity: LambdaTokenAuthorizationIdentity
```

Propriétés

`DisableFunctionDefaultPermissions`

Spécifiez `true` pour AWS SAM empêcher la création automatique d'une `AWS::Lambda::Permissions` ressource afin de fournir des autorisations entre votre `AWS::Serverless::Api` ressource et la fonction Lambda de l'autorisateur.

Valeur par défaut : `false`

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`FunctionArn`

Spécifiez la fonction ARN de la fonction Lambda qui fournit l'autorisation pour l'API.

> **Note**
>
> AWS SAM créera automatiquement une `AWS::Lambda::Permissions` ressource lorsque cela `FunctionArn` est spécifié pour`AWS::Serverless::Api`. La ressource

`AWS::Lambda::Permissions` fournit des autorisations entre votre API et la fonction Lambda du mécanisme d'autorisation.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`FunctionInvokeRole`

Ajoute les informations d'identification de l'autorisateur à la OpenApi définition de l'autorisateur Lambda.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`FunctionPayloadType`

Cette propriété peut être utilisée pour définir le type de mécanisme d'autorisation Lambda pour une Api.

Valeurs valides : `TOKEN` ou `REQUEST`

Type : chaîne

Obligatoire : non

Par défaut : `TOKEN`

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Identity`

Cette propriété peut être utilisée pour spécifier une `IdentitySource` dans une demande d'autorisation entrante. Cette propriété n'est requise que si la propriété `FunctionPayloadType` est définie sur `REQUEST`.

Type : LambdaTokenAuthorizationIdentity

Obligatoire : Conditionnelle

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

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

Cette propriété peut être utilisée pour spécifier un IdentitySource dans une demande entrante pour un autorisateur. Pour plus d'informations, IdentitySource consultez l' OpenApi extension ApiGateway Authorizer.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

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

Propriétés

`Header`

Spécifiez le nom de l'en-tête pour Authorization dans la OpenApi définition.

Type : chaîne

Obligatoire : non

Par défaut : autorisation

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`ReauthorizeEvery`

Période time-to-live (TTL), en secondes, qui indique la durée pendant laquelle API Gateway met en cache les résultats de l'autorisateur. Si vous définissez une valeur supérieure à 0, API Gateway met en cache les réponses du mécanisme d'autorisation. Par défaut, API Gateway définit cette propriété sur 300. La valeur maximale est 3 600 ou 1 heure.

Type : entier

Obligatoire : non

Par défaut : 300

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`ValidationExpression`

Spécifiez une expression de validation pour valider l'identité entrante.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

LambdaTokenIdentity

YAML

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

ResourcePolicyStatement

Configure une stratégie de ressource pour toutes les méthodes et chemins d'accès d'une API. Pour plus d'informations sur les stratégies de ressources, consultez Contrôle de l'accès à une API avec des stratégies de ressources API Gateway dans le Guide du développeur API.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
  AwsAccountBlacklist: List
  AwsAccountWhitelist: List
  CustomStatements: List
  IntrinsicVpcBlacklist: List
  IntrinsicVpcWhitelist: List
  IntrinsicVpceBlacklist: List
  IntrinsicVpceWhitelist: List
  IpRangeBlacklist: List
  IpRangeWhitelist: List
  SourceVpcBlacklist: List
```

```
  SourceVpcWhitelist: List
```

Propriétés

`AwsAccountBlacklist`

Les AWS comptes à bloquer.

Type : liste de chaînes

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`AwsAccountWhitelist`

Les AWS comptes à autoriser. Pour obtenir un exemple d'utilisation de cette propriété, consultez la section Exemples en bas de cette page.

Type : liste de chaînes

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`CustomStatements`

Une liste des instructions de stratégie de ressource personnalisées à appliquer à cette API. Pour obtenir un exemple d'utilisation de cette propriété, consultez la section Exemples en bas de cette page.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`IntrinsicVpcBlacklist`

La liste des clouds privés virtuels (VPCs) à bloquer, où chaque VPC est spécifié comme référence, telle qu'une référence dynamique ou une fonction `Ref` intrinsèque. Pour obtenir un exemple d'utilisation de cette propriété, consultez la section Exemples en bas de cette page.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`IntrinsicVpcWhitelist`

La liste des VPCs à autoriser, où chaque VPC est spécifié comme référence, telle qu'une référence dynamique ou une fonction `Ref` intrinsèque.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`IntrinsicVpceBlacklist`

Liste des points de terminaison VPC à bloquer, où chaque point de terminaison VPC est spécifié comme une référence telle qu'une Référence dynamique ou la fonction intrinsèque `Ref`.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`IntrinsicVpceWhitelist`

Liste des points de terminaison VPC à autoriser, où chaque point de terminaison VPC est spécifié en tant que référence telle qu'une Référence dynamique ou la fonction intrinsèque `Ref`. Pour obtenir un exemple d'utilisation de cette propriété, consultez la section Exemples en bas de cette page.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`IpRangeBlacklist`

L'adresse IP ou les plages d'adresses à bloquer. Pour obtenir un exemple d'utilisation de cette propriété, consultez la section Exemples en bas de cette page.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`IpRangeWhitelist`

L'adresse IP ou les plages d'adresses à autoriser.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`SourceVpcBlacklist`

Les points de terminaison VPC ou VPC source à bloquer. Les noms de VPC source doivent commencer par `"vpc-"` et les noms de point de terminaison VPC source doivent commencer par `"vpce-"`. Pour obtenir un exemple d'utilisation de cette propriété, consultez la section Exemples en bas de cette page.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`SourceVpcWhitelist`

Les points de terminaison VPC ou VPC source à autoriser. Les noms de VPC source doivent commencer par `"vpc-"` et les noms de point de terminaison VPC source doivent commencer par `"vpce-"`.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

Exemples de stratégie basée sur les ressources

L'exemple suivant bloque deux adresses IP et un VPC source, et autorise un AWS compte.

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

Un document OpenAPI définissant l'API.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```yaml
Bucket: String
Key: String
Version: String
```

Propriétés

`Bucket`

Nom du compartiment Amazon S3 dans lequel le fichier OpenAPI est stocké.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `Bucket` propriété du type de `AWS::ApiGateway::RestApi` `S3Location` données.

`Key`

La clé Amazon S3 du fichier OpenAPI.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `Key` propriété du type de `AWS::ApiGateway::RestApi` `S3Location` données.

`Version`

Pour les objets versionnés, la version du fichier OpenAPI.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Version` propriété du type de `AWS::ApiGateway::RestApi` `S3Location` données.

Exemples

Exemple de définition d'Uri

Exemple de définition d'API

YAML

```
DefinitionUri:
  Bucket: amzn-s3-demo-bucket-name
  Key: mykey-name
  Version: 121212
```

CorsConfiguration

Gérez le partage de ressources entre origines (CORS) pour votre API Gateway. APIs Spécifiez le domaine à autoriser en tant que chaîne ou spécifiez un dictionnaire avec une configuration Cors supplémentaire.

> **Note**
>
> CORS nécessite AWS SAM de modifier votre définition OpenAPI. Créez une définition OpenAPI intégrée dans `DefinitionBody` le pour activer CORS. Si le `CorsConfiguration` est défini dans la définition d'OpenAPI et également au niveau de la propriété, il les AWS SAM fusionne. Le niveau de propriété est prioritaire sur la définition d'OpenAPI.

Pour plus d'informations, consultez Activation de CORS pour une ressource API REST dans API Gateway dans le Guide du développeur API Gateway.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
  AllowCredentials: Boolean
  AllowHeaders: String
```

```
  AllowMethods: String
  AllowOrigin: String
  MaxAge: String
```

Propriétés

`AllowCredentials`

Booléen indiquant si la requête est autorisée à contenir des informations d'identification.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`AllowHeaders`

Chaîne d'en-têtes à autoriser.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`AllowMethods`

Chaîne contenant les méthodes HTTP à autoriser.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`AllowOrigin`

Chaîne d'origine à autoriser. Il peut s'agir d'une liste séparée par des virgules au format chaîne.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`MaxAge`

Chaîne contenant le nombre de secondes requises pour mettre en cache la demande CORS Preflight.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

CorsConfiguration

Exemple de configuration CORS. Il ne s'agit que d'une partie d'un fichier AWS SAM modèle présentant une AWS::Serverless::Api définition avec CORS configuré et unAWS::Serverless::Function. Si vous utilisez une intégration de proxy Lambda ou une intégration de proxy HTTP, votre backend doit renvoyer les en-têtes `Access-Control-Allow-OriginAccess-Control-Allow-Methods`, et. `Access-Control-Allow-Headers`

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

Configure un domaine personnalisé pour une API.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
AccessAssociation: DomainAccessAssociation
BasePath: List
CertificateArn: String
DomainName: String
EndpointConfiguration: String
MutualTlsAuthentication: MutualTlsAuthentication
NormalizeBasePath: Boolean
OwnershipVerificationCertificateArn: String
```

```
  Policy: Json
  Route53: Route53Configuration
  SecurityPolicy: String
```

Propriétés

`AccessAssociation`

Configuration requise pour générer une `AWS::ApiGateway::DomainNameAccessAssociation` ressource.

AWS SAM génère une AWS::ApiGateway::DomainNameAccessAssociationressource lorsque cette propriété est définie. Pour plus d'informations sur CloudFormation les ressources générées, consultezCloudFormation Ressources générées pour AWS SAM.

Type : DomainAccessAssociation

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`BasePath`

Une liste des chemins de base à configurer avec le nom de domaine Amazon API Gateway.

Type : liste

Obligatoire : non

Par défaut : /

CloudFormation compatibilité : cette propriété est similaire à celle `BasePath` d'une `AWS::ApiGateway::BasePathMapping` ressource. AWS SAM crée plusieurs `AWS::ApiGateway::BasePathMapping` ressources, une par ressource `BasePath` spécifiée dans cette propriété.

`CertificateArn`

Le nom de ressource Amazon (ARN) d'un certificat AWS géré est le point de terminaison de ce nom de domaine. AWS Certificate Manager est la seule source prise en charge.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est similaire à celle `CertificateArn` d'une `AWS::ApiGateway::DomainName` ressource. S'il `EndpointConfiguration` est défini sur `REGIONAL` (valeur par défaut), il est `CertificateArn` mappé sur RegionalCertificateArninAWS::ApiGateway::DomainName. Si le `EndpointConfiguration` est défini sur`EDGE`, `CertificateArn` correspond à CertificateArninAWS::ApiGateway::DomainName. Si elle `EndpointConfiguration` est définie sur`PRIVATE`, cette propriété est transmise à la ressource AWS::ApiGateway::DomainNameV2.

Remarques supplémentaires : Pour un `EDGE` point de terminaison, vous devez créer le certificat dans la `us-east-1` AWS région.

`DomainName`

Le nom de domaine personnalisé pour votre API dans Amazon API Gateway. Les majuscules ne sont pas prises en charge.

AWS SAM génère une AWS::ApiGateway::DomainNameressource lorsque cette propriété est définie. Pour plus d'informations sur ce scénario, consultez DomainName la propriété est spécifiée. Pour plus d'informations sur CloudFormation les ressources générées, consultezCloudFormation Ressources générées pour AWS SAM.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `DomainName` propriété d'une `AWS::ApiGateway::DomainName` ressource, ou à `AWS::ApiGateway::DomainNameV2`when EndpointConfiguration est définie sur`PRIVATE`.

`EndpointConfiguration`

Définit le type de point de terminaison API Gateway à mapper au domaine personnalisé. La valeur de cette propriété détermine la manière dont la `CertificateArn` propriété est mappée CloudFormation.

Valeurs valides : `EDGE`, `REGIONAL` ou `PRIVATE`

Type : chaîne

Obligatoire : non

Par défaut : `REGIONAL`

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`MutualTlsAuthentication`

La configuration d'authentification de protocole TLS (Transport Layer Security) mutuelle pour un nom de domaine personnalisé.

Type : MutualTlsAuthentication

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `MutualTlsAuthentication` propriété d'une `AWS::ApiGateway::DomainName` ressource.

`NormalizeBasePath`

Indique si les caractères non alphanumériques sont autorisés dans les chemins de base définis par la propriété `BasePath`. Lorsque ce paramètre est défini sur `True`, les caractères non alphanumériques sont supprimés des chemins de base.

Utilisez `NormalizeBasePath` avec la propriété `BasePath`.

Type : valeur booléenne

Obligatoire : non

Valeur par défaut : VRAI

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`OwnershipVerificationCertificateArn`

ARN du certificat public que vous avez généré dans pour valider la propriété de votre domaine personnalisé. Obligatoire uniquement lorsque vous configurez le protocole TLS mutuel et que vous précisez un ARN de certificat d'autorité de certification privé ou importé ACM pour le `CertificateArn`.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `OwnershipVerificationCertificateArn` propriété d'une `AWS::ApiGateway::DomainName` ressource.

`Policy`

La politique IAM à associer au nom de domaine API Gateway. Applicable uniquement lorsque `EndpointConfiguration` ce paramètre est défini sur`PRIVATE`.

Type : Json

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Policy` propriété d'une `AWS::ApiGateway::DomainNameV2` ressource lorsqu'elle `EndpointConfiguration` est définie sur`PRIVATE`. Pour des exemples de documents de politique valides, voir AWS::ApiGateway::DomainNameV2.

`Route53`

Définit une configuration Amazon Route 53.

Type : Route53Configuration

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`SecurityPolicy`

Version du protocole TLS plus suite de chiffrement pour ce nom de domaine.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `SecurityPolicy` propriété d'une `AWS::ApiGateway::DomainName` ressource, ou à `AWS::ApiGateway::DomainNameV2`when `EndpointConfiguration` est définie sur`PRIVATE`. Pour les `PRIVATE` points de terminaison, seul le TLS_1_2 est pris en charge.

Exemples

DomainName

DomainName exemple

YAML

```
Domain:
  DomainName: www.example.com
  CertificateArn: arn-example
  EndpointConfiguration: EDGE
  Route53:
    HostedZoneId: Z1PA6795UKMFR9
  BasePath:
    - foo
    - bar
```

DomainAccessAssociation

Configure une association d'accès au domaine entre une source d'association d'accès et un nom de domaine personnalisé privé. La seule source d'association d'accès prise en charge est un identifiant de point de terminaison VPC. Pour plus d'informations, consultez la section Noms de domaine personnalisés pour le domaine privé APIs dans API Gateway.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
VpcEndpointId: String
```

Propriétés

`VpcEndpointId`

ID de point de terminaison du point de terminaison de l'interface VPC associé au service VPC API Gateway.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `AccessAssociationSource` propriété d'une `AWS::ApiGateway::DomainNameAccessAssociation` ressource.

Route53Configuration

Configure les ensembles d'enregistrements Route53 pour une API.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
DistributionDomainName: String
EvaluateTargetHealth: Boolean
HostedZoneId: String
HostedZoneName: String
IpV6: Boolean
Region: String
SetIdentifier: String
VpcEndpointDomainName: String
VpcEndpointHostedZoneId: String
```

Propriétés

`DistributionDomainName`

Configure une distribution personnalisée du nom de domaine personnalisé de l'API.

Type : chaîne

Obligatoire : non

Par défaut : utilisez la distribution API Gateway.

CloudFormation compatibilité : cette propriété est transmise directement à la `DNSName` propriété d'une `AWS::Route53::RecordSetGroup AliasTarget` ressource.

Remarques supplémentaires : nom de domaine d'une CloudFrontdistribution.

`EvaluateTargetHealth`

Lorsque cela EvaluateTargetHealth est vrai, un enregistrement d'alias hérite de l'état de la AWS ressource référencée, comme un équilibreur de charge Elastic Load Balancing ou un autre enregistrement de la zone hébergée.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `EvaluateTargetHealth` propriété d'une `AWS::Route53::RecordSetGroup AliasTarget` ressource.

Remarques supplémentaires : vous ne pouvez pas EvaluateTargetHealth définir la valeur true lorsque l'alias cible est une CloudFront distribution.

`HostedZoneId`

ID de la zone hébergée dans laquelle vous souhaitez créer des enregistrements.

Spécifiez `HostedZoneName` ou `HostedZoneId`, mais pas les deux. Si plusieurs zones hébergées portent le même nom de domaine, vous devez spécifier la zone hébergée à l'aide de la valeur `HostedZoneId`.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `HostedZoneId` propriété d'une `AWS::Route53::RecordSetGroup RecordSet` ressource.

`HostedZoneName`

Nom de la zone hébergée dans laquelle vous souhaitez créer des enregistrements.

Spécifiez `HostedZoneName` ou `HostedZoneId`, mais pas les deux. Si plusieurs zones hébergées portent le même nom de domaine, vous devez spécifier la zone hébergée à l'aide de la valeur `HostedZoneId`.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `HostedZoneName` propriété d'une `AWS::Route53::RecordSetGroup RecordSet` ressource.

`IpV6`

Lorsque cette propriété est définie, elle AWS SAM crée une `AWS::Route53::RecordSet` ressource et attribue à Type `AAAA` la valeur fournie HostedZone.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Region`

Ensembles d'enregistrements de ressources basés sur la latence uniquement : EC2 région Amazon dans laquelle vous avez créé la ressource à laquelle ce jeu d'enregistrements de ressources fait référence. La ressource est généralement une AWS ressource, telle qu'une EC2 instance ou un équilibreur de charge ELB, à laquelle il est fait référence par une adresse IP ou un nom de domaine DNS, selon le type d'enregistrement.

Lorsqu'Amazon Route 53 reçoit une requête DNS pour un nom et un type de domaine pour lesquels vous avez créé des ensembles d'enregistrements de ressources de latence, Route 53 sélectionne le jeu d'enregistrements de ressources de latence présentant la latence la plus faible entre l'utilisateur final et la EC2 région Amazon associée. Route 53 renvoie ensuite la valeur associée à l'ensemble d'enregistrements de ressource sélectionné.

Notez ce qui suit :

- Vous pouvez uniquement spécifier un `ResourceRecord` par ensemble d'enregistrements de ressource de latence.
- Vous ne pouvez créer qu'un seul ensemble d'enregistrements de ressources de latence pour chaque EC2 région Amazon.
- Vous n'êtes pas obligé de créer des ensembles d'enregistrements de ressources de latence pour toutes les EC2 régions Amazon. Route 53 choisit la région dotée de la meilleure latence parmi les régions pour lesquelles vous créez des ensembles d'enregistrements de ressource de latence.

- Vous ne pouvez pas créer des ensembles d'enregistrements de ressource sans latence ayant les mêmes valeurs pour les éléments `Name` et `Type` que celles des ensembles d'enregistrements de ressource de latence.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Region` propriété d'un type de `AWS::Route53::RecordSetGroup` `RecordSet` données.

`SetIdentifier`

Ensembles d'enregistrements de ressource ayant une politique de routage autre que simple : un identificateur qui se différencie parmi plusieurs ensembles d'enregistrements de ressource ayant la même combinaison de nom et de type, comme plusieurs ensembles d'enregistrements de ressource pondérés nommés acme.example.com ayant un type A. Dans un groupe d'ensembles d'enregistrements de ressource ayant les mêmes nom et type, la valeur `SetIdentifier` doit être unique pour chaque ensemble d'enregistrements de ressource.

Pour de plus amples informations sur les politiques de routage, veuillez consulter Choix d'une politique de routage dans le Guide du développeur Amazon Route 53.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `SetIdentifier` propriété d'un type de `AWS::Route53::RecordSetGroup` `RecordSet` données.

`VpcEndpointDomainName`

Nom DNS du point de terminaison de l'interface VPC associé au service VPC API Gateway. Cette propriété est requise uniquement pour les domaines privés.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : Cette propriété est transmise directement à la `DNSName` propriété d'un `AWS::Route53::RecordSet` `AliasTarget` champ.

`VpcEndpointHostedZoneId`

ID de zone hébergée du point de terminaison de l'interface VPC associé au service VPC API Gateway. Cette propriété est requise uniquement pour les domaines privés.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : Cette propriété est transmise directement à la `HostedZoneId` propriété d'un `AWS::Route53::RecordSet` `AliasTarget` champ.

Exemples

Exemple de base

Dans cet exemple, nous configurons un domaine personnalisé et des ensembles d'enregistrements Route 53 pour notre 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

Type de point de terminaison d'une API REST.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```yaml
IpAddressType: String
Type: String
VPCEndpointIds: List
```

Propriétés

`IpAddressType`

Les types d'adresses IP qui peuvent appeler une API (RestApi).

Valeurs valides : `ipv4` ou `dualstack`

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `IpAddressType` propriété du type de `AWS::ApiGateway::RestApi` `EndpointConfiguration` données.

`Type`

Type de point de terminaison d'une API REST.

Valeurs valides : `EDGE`, `REGIONAL` ou `PRIVATE`

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Types` propriété du type de `AWS::ApiGateway::RestApi` `EndpointConfiguration` données.

`VPCEndpointIds`

Liste des points de terminaison VPC IDs d'une API REST par rapport auxquels créer des alias Route53.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `VpcEndpointIds` propriété du type de `AWS::ApiGateway::RestApi` `EndpointConfiguration` données.

Exemples

EndpointConfiguration

Exemple de configuration de point de terminaison

YAML

```
EndpointConfiguration:
  Type: PRIVATE
  VPCEndpointIds:
    - vpce-123a123a
    - vpce-321a321a
```

AWS::Serverless::Application

Intègre une application sans serveur à partir du AWS Serverless Application Repository ou à partir d'un compartiment Amazon S3 comme application imbriquée. Les applications imbriquées sont déployées en tant que ressourcesAWS::CloudFormation::Stack, qui peuvent contenir plusieurs autres ressources, y compris d'autres ressources AWS::Serverless::Application.

> **Note**
>
> Lorsque vous déployez vers AWS CloudFormation, vos AWS SAM ressources sont AWS SAM transformées en CloudFormation ressources. Pour de plus amples informations, veuillez consulter CloudFormation Ressources générées pour AWS SAM.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
Type: AWS::Serverless::Application
Properties:
```

```
  Location: String | ApplicationLocationObject
  NotificationARNs: List
  Parameters: Map
  Tags: Map
  TimeoutInMinutes: Integer
```

Propriétés

`Location`

URL du modèle, chemin d'accès au fichier ou objet d'emplacement d'une application imbriquée.

Si une URL de modèle est fournie, elle doit suivre le format spécifié dans la CloudFormation TemplateUrl documentation et contenir un modèle valide CloudFormation ou SAM. Une ApplicationLocationObject peut être utilisée pour spécifier une application qui a été publiée dans le AWS Serverless Application Repository.

Si un chemin d'accès de fichier local est fourni, le modèle doit passer par le flux comprenant la propriété `sam deploy` ou `sam package`, afin que l'application soit correctement transformée.

Type : Chaîne | ApplicationLocationObject

Obligatoire : oui

CloudFormation compatibilité : cette propriété est similaire à celle `TemplateURL` d'une `AWS::CloudFormation::Stack` ressource. La CloudFormation version ne nécessite pas de fichier ApplicationLocationObject pour récupérer une application depuis le AWS Serverless Application Repository.

`NotificationARNs`

Une liste des rubriques Amazon SNS existantes dans lesquelles les notifications relatives aux événements de la pile sont envoyées.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `NotificationARNs` propriété d'une `AWS::CloudFormation::Stack` ressource.

`Parameters`

Valeurs des paramètres de l'application.

Type: carte (map)

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Parameters` propriété d'une `AWS::CloudFormation::Stack` ressource.

`Tags`

Un mappage (chaîne à chaîne) qui spécifie les balises à ajouter à cette application. Les clés et les valeurs sont limitées aux caractères alphanumériques. Les clés peuvent comporter de 1 à 127 caractères Unicode en longueur et ne peut pas être précédée de « aws: ». Les valeurs de balise doivent comporter de 1 à 255 caractères Unicode en longueur.

Type: carte (map)

Obligatoire : non

CloudFormation compatibilité : cette propriété est similaire à celle `Tags` d'une `AWS::CloudFormation::Stack` ressource. La propriété Tags dans SAM se compose de paires Key:Value ; CloudFormation elle consiste en une liste d'objets Tag. Lorsque la pile est créée, SAM ajoute automatiquement une balise `lambda:createdBy:SAM` à cette application. De plus, si cette application provient du AWS Serverless Application Repository, SAM ajoutera automatiquement les deux balises supplémentaires `serverlessrepo:applicationId:`*`ApplicationId`* et`serverlessrepo:semanticVersion:`*`SemanticVersion`*.

`TimeoutInMinutes`

Durée, en minutes, pendant laquelle CloudFormation la pile imbriquée doit atteindre cet état. `CREATE_COMPLETE` La valeur par défaut m'inclut pas d'expiration. Lorsqu'il CloudFormation détecte que la pile imbriquée a atteint `CREATE_COMPLETE` cet état, il marque la ressource de pile imbriquée comme étant `CREATE_COMPLETE` dans la pile parent et reprend la création de la pile parent. Si le délai expire avant que la pile imbriquée n'atteigne son objectif`CREATE_COMPLETE`, CloudFormation marque la pile imbriquée comme ayant échoué et annule à la fois la pile imbriquée et la pile parent.

Type : entier

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `TimeoutInMinutes` propriété d'une `AWS::CloudFormation::Stack` ressource.

Valeurs renvoyées

Réf

Lorsque l'ID logique de cette ressource est fournie à la fonction intrinsèque `Ref`, elle renvoie le nom de la ressource `AWS::CloudFormation::Stack` sous-jacente.

Pour plus d'informations sur l'utilisation de la fonction `Ref`, consultez `Ref` dans le Guide de l'utilisateur AWS CloudFormation .

Ventilateur : GetAtt

`Fn::GetAtt` renvoie une valeur pour un attribut de ce type indiqué. Voici les attributs disponibles et des exemples de valeurs de retour.

Pour plus d'informations sur l'utilisation de `Fn::GetAtt`, consultez `Fn::GetAtt` dans le Guide de l'utilisateur AWS CloudFormation .

`Outputs.ApplicationOutputName`

La valeur de la sortie de la pile avec le nom*`ApplicationOutputName`*.

Exemples

Application SAR

Application qui utilise un modèle à partir du Serverless Application Repository

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
```

Application normale

Application à partir d'une URL S3

YAML

```
Type: AWS::Serverless::Application
Properties:
  Location: https://s3.amazonaws.com/sam-s3-demo-bucket/template.yaml
```

ApplicationLocationObject

Une application qui a été publiée dans le AWS Serverless Application Repository.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
  ApplicationId: String
  SemanticVersion: String
```

Propriétés

`ApplicationId`

Amazon Resource Name (ARN) de l'application.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`SemanticVersion`

Version sémantique de l'application.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

mon-application

Exemple d'objet d'emplacement d'application

YAML

```
Location:
  ApplicationId: 'arn:aws:serverlessrepo:us-east-1:012345678901:applications/my-application'
  SemanticVersion: 1.0.0
```

AWS::Serverless::CapacityProvider

Crée un fournisseur de capacité pour les AWS Lambda fonctions qui permettent d'exécuter des instances gérées Lambda sur des instances Amazon Elastic Compute Cloud appartenant au client. Cette ressource fait partie de la fonctionnalité Lambda Managed Instances, qui permet d'optimiser les coûts pour les charges de travail Lambda à grande échelle en utilisant les modèles de tarification Amazon. EC2

Le fournisseur de capacité gère le cycle de vie des EC2 instances Amazon et fournit l'infrastructure nécessaire pour que les fonctions Lambda s'exécutent sur les ressources informatiques appartenant au client tout en conservant le modèle de programmation sans serveur.

> **Note**
>
> Lorsque vous déployez vers AWS CloudFormation, vos AWS SAM ressources sont AWS SAM transformées en CloudFormation ressources. Pour de plus amples informations, veuillez consulter CloudFormation Ressources générées pour AWS SAM.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```yaml
Type: AWS::Serverless::CapacityProvider
Properties:
  CapacityProviderName: String
  VpcConfig: VpcConfig
  OperatorRole: String
  Tags: Map
  PropagateTags: Boolean
  InstanceRequirements: InstanceRequirements
  ScalingConfig: ScalingConfig
  KmsKeyArn: String
```

Propriétés

`CapacityProviderName`

Nom du fournisseur de capacité. Ce nom doit être unique au sein de votre AWS compte et de votre région.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `CapacityProviderName` propriété d'une `AWS::Lambda::CapacityProvider` ressource.

`VpcConfig`

Configuration VPC pour le fournisseur de capacité. Spécifie les sous-réseaux VPC et les groupes de sécurité sur lesquels les EC2 instances Amazon seront lancées.

Type : VpcConfig

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `VpcConfig` propriété d'une `AWS::Lambda::CapacityProvider` ressource.

`OperatorRole`

L'ARN du rôle d'opérateur pour Lambda avec les autorisations nécessaires pour créer et gérer les EC2 instances Amazon et les ressources associées dans le compte client. Si ce n'est pas le cas, il génère AWS SAM automatiquement un rôle avec les autorisations nécessaires.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : Cette propriété est transmise directement à la `CapacityProviderOperatorRoleArn` propriété `PermissionsConfig` d'une `AWS::Lambda::CapacityProvider` ressource.

`Tags`

Carte des paires clé-valeur à appliquer au fournisseur de capacité et à ses ressources associées.

Type: carte (map)

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Tags` propriété d'une `AWS::Lambda::CapacityProvider` ressource. La `Tags` propriété in AWS SAM se compose de paires clé-valeur (alors que dans CloudFormation cette propriété se compose d'une liste d'objets Tag). Ajoute également AWS SAM automatiquement une `lambda:createdBy:SAM` balise à cette fonction Lambda et aux rôles par défaut générés pour cette fonction.

`PropagateTags`

Indique s'il faut ou non transmettre les balises de la propriété Tags aux ressources que vous `AWS::Serverless::CapacityProvider` avez générées. Définissez ce paramètre sur `True` pour propager les balises dans les ressources que vous avez générées.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`InstanceRequirements`

Spécifications relatives aux types d'instances de calcul que le fournisseur de capacité peut utiliser. Cela inclut les exigences d'architecture `allowed` et/ou les types d'`excluded`instances.

Type : InstanceRequirements

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `InstanceRequirements` propriété d'une `AWS::Lambda::CapacityProvider` ressource.

`ScalingConfig`

Configuration de mise à l'échelle pour le fournisseur de capacité. Définit la manière dont le fournisseur de capacité adapte EC2 les instances Amazon en fonction de la demande.

Type : ScalingConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `CapacityProviderScalingConfig` propriété d'une `AWS::Lambda::CapacityProvider` ressource.

`KmsKeyArn`

L'ARN de la AWS KMS clé utilisée pour chiffrer les données au repos et en transit pour le fournisseur de capacité.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `KmsKeyArn` propriété d'une `AWS::Lambda::CapacityProvider` ressource.

Valeurs renvoyées

Réf

Lorsque l'identifiant logique de cette ressource est fourni à la fonction `Ref` intrinsèque, elle renvoie le nom du fournisseur de capacité.

Pour plus d'informations sur l'utilisation de la fonction `Ref`, consultez `Ref` dans le Guide de l'utilisateur AWS CloudFormation .

Ventilateur : GetAtt

`Fn::GetAtt` renvoie une valeur pour un attribut de ce type indiqué. Voici les attributs disponibles et des exemples de valeurs de retour.

Pour plus d'informations sur l'utilisation de `Fn::GetAtt`, consultez `Fn::GetAtt` dans le Guide de l'utilisateur AWS CloudFormation .

`Arn`

L'ARN du fournisseur de capacité.

Exemples

Fournisseur de capacité de base

L'exemple suivant crée un fournisseur de capacité de base avec une configuration VPC.

```
MyCapacityProvider:
  Type: AWS::Serverless::CapacityProvider
  Properties:
    CapacityProviderName: my-capacity-provider
    VpcConfig:
      SubnetIds:
        - subnet-12345678
        - subnet-87654321
      SecurityGroupIds:
        - sg-12345678
    Tags:
      Environment: Production
      Team: ServerlessTeam
```

Fournisseur de capacité avancé avec évolutivité

L'exemple suivant crée un fournisseur de capacité avec des exigences d'instance personnalisées et une configuration de dimensionnement.

```
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
    Tags:
      Environment: Production
      CostCenter: Engineering
```

VpcConfig

Configure les paramètres VPC d'un fournisseur de capacité, y compris les sous-réseaux et les groupes de sécurité sur EC2 lesquels les instances seront lancées.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
SubnetIds: List
SecurityGroupIds: List
```

Propriétés

`SubnetIds`

Liste des sous-réseaux IDs dans lesquels EC2 les instances seront lancées. Au moins un sous-réseau doit être spécifié.

Type : liste

Obligatoire : oui

CloudFormation compatibilité : Cette propriété est transmise directement à la `SubnetIds` propriété `VpcConfig` d'une `AWS::Lambda::CapacityProvider` ressource.

`SecurityGroupIds`

Liste des groupes de sécurité IDs à associer aux EC2 instances. S'il n'est pas spécifié, le groupe de sécurité par défaut du VPC sera utilisé.

Type : liste

Obligatoire : non

CloudFormation compatibilité : Cette propriété est transmise directement à la `SecurityGroupIds` `VpcConfig` propriété d'une `AWS::Lambda::CapacityProvider` ressource.

Exemples

Configuration VPC

L'exemple suivant montre une configuration VPC avec plusieurs sous-réseaux et groupes de sécurité.

```
VpcConfig:
  SubnetIds:
    - subnet-12345678
    - subnet-87654321
  SecurityGroupIds:
    - sg-12345678
    - sg-87654321
```

InstanceRequirements

Spécifie les exigences relatives aux EC2 instances qui seront lancées par le fournisseur de capacité, y compris les architectures et les contraintes de type d'instance.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```yaml
Architectures: List
AllowedTypes: List
ExcludedTypes: List
```

> **Note**
>
> Vous pouvez choisir de spécifier l'une `AllowedTypes` ou l'autre des exigences d'instance pour votre fournisseur de capacité ou `ExcludedTypes` lors de la définition des exigences en matière d'instance, mais pas les deux.

Propriétés

`Architectures`

Les architectures du jeu d'instructions pour les instances du fournisseur de capacité.

Valeurs valides : `x86_64` ou `arm64`

Type : liste

Obligatoire : non

Par défaut : `x86_64`

CloudFormation compatibilité : Cette propriété est transmise directement à la `Architectures` propriété `InstanceRequirements` d'une `AWS::Lambda::CapacityProvider` ressource.

`AllowedTypes`

Liste des types d' EC2 instances autorisés pour l'instance du fournisseur de capacité.

Type : liste

Obligatoire : non

CloudFormation compatibilité : Cette propriété est transmise directement à la `AllowedInstanceTypes` propriété `InstanceRequirements` d'une `AWS::Lambda::CapacityProvider` ressource.

`ExcludedTypes`

Liste des types d' EC2 instances à exclure du fournisseur de capacité.

Type : liste

Obligatoire : non

CloudFormation compatibilité : Cette propriété est transmise directement à la `ExcludedInstanceTypes` propriété `InstanceRequirements` d'une `AWS::Lambda::CapacityProvider` ressource.

Exemples

Configuration des exigences de l'instance

L'exemple suivant montre les exigences d'une instance avec des contraintes d'architecture et de type d'instance spécifiques.

```
InstanceRequirements:
  Architectures:
    - x86_64
  ExcludedTypes:
    - t2.micro
```

ScalingConfig

Configure la manière dont le fournisseur de capacité fait évoluer les EC2 instances en fonction de la demande, y compris les limites maximales d'instances et les politiques de dimensionnement.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
MaxVCpuCount: Integer
AverageCPUUtilization: Double
```

Propriétés

`MaxVCpuCount`

Le nombre maximum de v CPUs que le fournisseur de capacité peut fournir sur toutes les instances de calcul.

Type : entier

Obligatoire : non

CloudFormation compatibilité : Cette propriété est transmise directement à la `MaxVCpuCount` propriété `CapacityProviderScalingConfig` d'une `AWS::Lambda::CapacityProvider` ressource.

`AverageCPUUtilization`

Pourcentage moyen d'utilisation du processeur cible (0 à 100) pour les décisions de dimensionnement. Lorsque l'utilisation moyenne du processeur dépasse ce seuil, le fournisseur de capacité augmente le volume EC2 des instances Amazon. Lorsque cela est spécifié, AWS SAM construit `CapacityProviderScalingConfig` une `AWS::Lambda::CapacityProvider` ressource avec les valeurs `ScalingMode` set to `'Manual'` et `ScalingPolicies` set to`[{PredefinedMetricType: 'LambdaCapacityProviderAverageCPUUtilization', TargetValue: <this value>}]`.

Type : double

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

Configuration de la mise à l'échelle

L'exemple suivant montre une configuration de dimensionnement avec un VCpu nombre maximal et une utilisation moyenne du processeur.

```
ScalingConfig:
```

```
  MaxVCpuCount: 10
  AverageCPUUtilization: 70.0
```

AWS::Serverless::Connector

Configure les autorisations entre deux ressources. Pour obtenir une présentation des connecteurs, veuillez consulter Gestion des autorisations relatives aux ressources à l'aide de AWS SAM connecteurs.

Pour plus d'informations sur les AWS CloudFormation ressources générées, consultezCloudFormation ressources générées lorsque vous spécifiez AWS::Serverless::Connector.

Pour fournir des commentaires sur les connecteurs, soumettez un nouveau problème dans le serverless-application-model AWS GitHub référentiel.

> **Note**
>
> Lorsque vous déployez vers AWS CloudFormation, vos AWS SAM ressources sont AWS SAM transformées en CloudFormation ressources. Pour de plus amples informations, veuillez consulter CloudFormation Ressources générées pour AWS SAM.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez l'une des syntaxes suivantes.

> **Note**
>
> Nous vous recommandons d'utiliser la syntaxe des connecteurs intégrés dans la plupart des cas d'utilisation. Le fait d'être intégré à la ressource source facilite sa lecture et sa maintenance au fil du temps. Lorsque vous devez référencer une ressource source qui ne figure pas dans le même AWS SAM modèle, telle qu'une ressource dans une pile imbriquée ou une ressource partagée, utilisez la `AWS::Serverless::Connector` syntaxe.

Connecteurs intégrés

```
<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
```

Propriétés

`Destination`

Ressource de destination.

Type : ResourceReference| Liste des ResourceReference

Obligatoire : oui

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Permissions`

Type d'autorisation que la ressource source est autorisée à exécuter sur la ressource de destination.

`Read`inclut des actions Gestion des identités et des accès AWS (IAM) qui permettent de lire les données de la ressource.

`Write` inclut des actions IAM qui permettent de lancer et d'écrire des données dans une ressource.

Valeurs valides : `Read` ou `Write`

Type : liste

Obligatoire : oui

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Source`

Ressource source. Obligatoire lors de l'utilisation de la syntaxe `AWS::Serverless::Connector`.

Type : ResourceReference

Obligatoire : Conditionnelle

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`SourceReference`

Ressource source.

> **Note**
> À utiliser avec la syntaxe des connecteurs intégrés lors de la définition de propriétés supplémentaires pour la ressource source.

Type : SourceReference

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

Connecteurs intégrés

L'exemple suivant utilise des connecteurs intégrés pour définir une connexion de données `Write` entre une fonction AWS Lambda et un tableau Amazon DynamoDB :

```
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
    ...
```

L'exemple suivant utilise des connecteurs intégrés pour définir les autorisations `Read` et `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
    ...
```

L'exemple suivant utilise des connecteurs intégrés pour définir une ressource source avec une propriété autre que la propriété `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

L'exemple suivant utilise la AWS::Serverless::Connector ressource pour lire et écrire une AWS Lambda fonction dans une table Amazon DynamoDB :

```
MyConnector:
  Type: AWS::Serverless::Connector
  Properties:
    Source:
      Id: MyFunction
    Destination:
      Id: MyTable
    Permissions:
      - Read
      - Write
```

L'exemple suivant utilise la ressource AWS::Serverless::Connector pour qu'une fonction Lambda écrive dans une rubrique Amazon SNS, avec les deux ressources dans le même modèle :

```
MyConnector:
  Type: AWS::Serverless::Connector
  Properties:
    Source:
      Id: MyLambda
    Destination:
      Id: MySNSTopic
    Permissions:
```

```
      - Write
```

L'exemple suivant utilise la ressource AWS::Serverless::Connector pour qu'une rubrique Amazon SNS écrive dans une fonction Lambda, qui écrit à son tour dans un tableau Amazon DynamoDB, avec toutes les ressources dans le même modèle :

```
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
```

Le AWS CloudFormation modèle transformé de l'exemple ci-dessus est le suivant :

```
"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

Référence à une ressource utilisée par le type de ressource AWS::Serverless::Connector.

> **Note**
>
> Pour les ressources du même modèle, fournissez l'`Id`. Pour les ressources qui ne se trouvent pas dans le même modèle, utilisez une combinaison d'autres propriétés. Pour de plus amples informations, veuillez consulter AWS SAM référence du connecteur.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```yaml
Arn: String
Id: String
Name: String
Qualifier: String
QueueUrl: String
ResourceId: String
RoleName: String
Type: String
```

Propriétés

`Arn`

ARN d'une ressource.

Type : chaîne

Obligatoire : Conditionnelle

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Id`

ID logique d'une ressource dans le même modèle.

> **Note**
>
> Lorsque cela `Id` est spécifié, si le connecteur génère des politiques Gestion des identités et des accès AWS (IAM), le rôle IAM associé à ces politiques sera déduit de la ressource. `Id` Lorsque la propriété `Id` n'est pas spécifiée, renseignez la propriété `RoleName` de la ressource permettant aux connecteurs d'attacher les politiques IAM générées à un rôle IAM.

Type : chaîne

Obligatoire : Conditionnelle

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Name`

Nom d'une ressource.

Type : chaîne

Obligatoire : Conditionnelle

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Qualifier`

Qualificateur d'une ressource qui réduit sa portée. `Qualifier` remplace la valeur `*` à la fin d'une contrainte de ressource ARN. Pour obtenir un exemple, consultez Appel d'une fonction Lambda par API Gateway.

> **Note**
>
> La définition du qualificateur varie selon le type de ressource. Pour obtenir la liste des types de ressource source et de destination pris en charge, consultez AWS SAM référence du connecteur.

Type : chaîne

Obligatoire : Conditionnelle

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`QueueUrl`

URL de la file d'attente Amazon SQS. Cette propriété s'applique uniquement aux ressources Amazon SQS.

Type : chaîne

Obligatoire : Conditionnelle

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`ResourceId`

ID d'une ressource. Par exemple, l'ID de l'API API Gateway.

Type : chaîne

Obligatoire : Conditionnelle

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`RoleName`

Nom du rôle associé à une ressource.

> **Note**
>
> Quand la propriété `Id` est spécifiée, si le connecteur génère des politiques IAM, le rôle IAM associé à ces politiques est déduit de la ressource `Id`. Lorsque la propriété `Id` n'est pas spécifiée, renseignez la propriété `RoleName` de la ressource permettant aux connecteurs d'attacher les politiques IAM générées à un rôle IAM.

Type : chaîne

Obligatoire : Conditionnelle

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Type`

CloudFormation Type de ressource. Pour de plus amples informations, veuillez consulter la Référence des types de propriété et de ressource AWS.

Type : chaîne

Obligatoire : Conditionnelle

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

Appel d'une fonction Lambda par API Gateway

L'exemple suivant utilise la AWS::Serverless::Connector ressource pour autoriser Amazon API Gateway à appeler une AWS Lambda fonction.

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

Référence à une ressource source utilisée par le type de ressource AWS::Serverless::Connector.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
Qualifier: String
```

Propriétés

`Qualifier`

Qualificateur d'une ressource qui réduit sa portée. `Qualifier` remplace la valeur `*` à la fin d'une contrainte de ressource ARN.

> **Note**
>
> La définition du qualificateur varie selon le type de ressource. Pour obtenir la liste des types de ressource source et de destination pris en charge, consultez AWS SAM référence du connecteur.

Type : chaîne

Obligatoire : Conditionnelle

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

L'exemple suivant utilise des connecteurs intégrés pour définir une ressource source avec une propriété autre que**`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::Function

Crée une AWS Lambda fonction, un rôle d'exécution Gestion des identités et des accès AWS (IAM) et des mappages de sources d'événements qui déclenchent la fonction.

La AWS::Serverless::Function ressource prend également en charge l'attribut `Metadata` resource, ce qui vous permet de demander de AWS SAM créer des environnements d'exécution personnalisés dont votre application a besoin. Pour plus d'informations sur la création d'une exécution personnalisée, consultez Création de fonctions Lambda avec des environnements d'exécution personnalisés dans AWS SAM.

> **Note**
>
> Lorsque vous déployez vers AWS CloudFormation, vos AWS SAM ressources sont AWS SAM transformées en CloudFormation ressources. Pour de plus amples informations, veuillez consulter CloudFormation Ressources générées pour AWS SAM.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

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
```

Propriétés

`Architectures`

Architecture du jeu d'instructions de la fonction.

Pour en savoir plus sur cette propriété, consultez Architectures du jeu d'instructions Lambda dans leguide du développeur AWS Lambda .

Valeurs valides : Une de `x86_64` ou `arm64`

Type : liste

Obligatoire : non

Par défaut : `x86_64`

CloudFormation compatibilité : cette propriété est transmise directement à la `Architectures` propriété d'une `AWS::Lambda::Function` ressource.

`AssumeRolePolicyDocument`

Ajoute un AssumeRolePolicyDocument pour la valeur par défaut créée `Role` pour cette fonction. Si cette propriété n'est pas spécifiée, AWS SAM ajoute un rôle d'assume par défaut pour cette fonction.

Type : JSON

Obligatoire : non

CloudFormation compatibilité : cette propriété est similaire à celle `AssumeRolePolicyDocument` d'une `AWS::IAM::Role` ressource. AWS SAM ajoute cette propriété au rôle IAM généré pour cette fonction. Si l'Amazon Resource Name (ARN) d'un rôle est fourni pour cette fonction, cette propriété ne fait rien.

`AutoPublishAlias`

Le nom de l'alias Lambda. Pour plus d'informations sur l'utilisation des alias Lambda, consultez Alias de fonction Lambda dans le Guide du développeur AWS Lambda . Pour obtenir des exemples qui utilisent cette propriété, consultez Déploiement progressif d'applications sans serveur avec AWS SAM.

AWS SAM génère AWS::Lambda::Versionet fournit AWS::Lambda::Aliasdes ressources lorsque cette propriété est définie. Pour plus d'informations sur ce scénario, consultez AutoPublishAlias la propriété est spécifiée. Pour des informations générales sur les CloudFormation ressources générées, consultezCloudFormation Ressources générées pour AWS SAM.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`AutoPublishAliasAllProperties`

Spécifie quand une nouvelle version `AWS::Lambda::Version` est créée. Lorsque l'option est `true`, une nouvelle version Lambda est créée lorsqu'une propriété de la fonction Lambda est modifiée. Lorsque l'option est `false`, une nouvelle version Lambda est créée uniquement lorsque l'une des propriétés suivantes est modifiée :

- `Environment`, `MemorySize`, ou `SnapStart`.

- Toute modification entraînant une mise à jour de la propriété du `Code`, telle que `CodeDict`, `ImageUri` ou `InlineCode`.

Cette propriété nécessite que `AutoPublishAlias` soit défini.

Si `AutoPublishCodeSha256` est également spécifié, son comportement est prioritaire sur `AutoPublishAliasAllProperties: true`.

Type : valeur booléenne

Obligatoire : non

Valeur par défaut : `false`

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`AutoPublishCodeSha256`

Lorsqu'elle est utilisée, cette chaîne fonctionne avec la `CodeUri` valeur pour déterminer si une nouvelle version de Lambda doit être publiée. Cette propriété est souvent utilisée pour résoudre le problème de déploiement suivant : un package de déploiement est stocké dans un emplacement Amazon S3 et est remplacé par un nouveau package de déploiement avec un code de fonction Lambda mis à jour, mais la `CodeUri` propriété reste inchangée (par opposition au téléchargement du nouveau package de déploiement vers un nouvel emplacement Amazon S3 et `CodeUri` remplacé par le nouvel emplacement).

Ce problème est marqué par un AWS SAM modèle présentant les caractéristiques suivantes :

- L'`DeploymentPreference`objet est configuré pour des déploiements progressifs (comme décrit dansDéploiement progressif d'applications sans serveur avec AWS SAM)
- La `AutoPublishAlias` propriété est définie et ne change pas entre les déploiements
- La `CodeUri` propriété est définie et ne change pas entre les déploiements.

Dans ce scénario, la mise à jour des résultats `AutoPublishCodeSha256` entraîne la création d'une nouvelle version Lambda. Toutefois, le nouveau code de fonction déployé sur Amazon S3 n'est pas reconnu. Pour reconnaître le nouveau code de fonction, pensez à utiliser la gestion des versions dans votre compartiment Amazon S3. Spécifiez la propriété `Version` de votre fonction Lambda et configurez votre compartiment pour qu'il utilise toujours le dernier Package de déploiement.

Dans ce scénario, pour déclencher le déploiement progressif avec succès, vous devez fournir une valeur unique pour `AutoPublishCodeSha256`.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`CapacityProviderConfig`

Configure le fournisseur de capacité auquel les versions publiées de la fonction seront associées. Cela permet à la fonction de s'exécuter sur des instances EC2 appartenant au client et gérées par des instances gérées par Lambda.

Type : CapacityProviderConfig

Obligatoire : non

CloudFormation compatibilité : SAM aplatit la propriété transmise à la `CapacityProviderConfig` propriété d'une `AWS::Lambda::Function` ressource et reconstruit la structure imbriquée.

`CodeSigningConfigArn`

L'ARN de la ressource AWS::Lambda::CodeSigningConfig, utilisée pour activer la signature de code pour cette fonction. Pour plus d'informations sur la signature de code, consultez Configurer la signature de code pour votre AWS SAM application.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `CodeSigningConfigArn` propriété d'une `AWS::Lambda::Function` ressource.

`CodeUri`

Code pour la fonction. Les valeurs acceptées incluent :

- l'URI Amazon S3 de la fonction ; Par exemple, `s3://bucket-123456789/sam-app/1234567890abcdefg`.

- le chemin local vers la fonction. Par exemple, `hello_world/`.
- Un objet FunctionCode.

> **Note**
>
> Si vous fournissez l'URI Amazon S3 d'une fonction ou l'objet FunctionCode, vous devez référencer un package de déploiement Lambda valide.
> Si vous fournissez un chemin de fichier local, utilisez la CLI AWS SAM pour charger le fichier local lors du déploiement. Pour en savoir plus, veuillez consulter la section Comment AWS SAM télécharge les fichiers locaux lors du déploiement.
> Si vous utilisez des fonctions intrinsèques dans les `CodeUri` propriétés, vous ne AWS SAM pourrez pas analyser correctement les valeurs. Envisagez plutôt d'utiliser AWS::LanguageExtensions transform.

Type : [chaîne de caractères | FunctionCode]

Obligatoire : selon les conditions. Lorsque `PackageType` est défini sur `Zip`, un `CodeUri` ou un `InlineCode` est requis.

CloudFormation compatibilité : cette propriété est similaire à celle `Code` d'une `AWS::Lambda::Function` ressource. Les propriétés imbriquées d'Amazon S3 sont nommées différemment.

`DeadLetterQueue`

Configure une rubrique Amazon Simple Notification Service (Amazon SNS) ou une file d'attente Amazon Simple Queue Service (Amazon SQS) où Lambda envoie les événements qu'il ne peut pas traiter. Pour en savoir plus sur la fonctionnalité de la file d'attente de lettres mortes, consultez File d'attente de lettres mortes dans le Guide du développeur AWS Lambda .

> **Note**
>
> Si la source d'événements de votre fonction Lambda est une file d'attente Amazon SQS, configurez une file d'attente de lettres mortes pour la file d'attente source, et pas pour la fonction Lambda. La file d'attente de lettres mortes que vous configurez pour une fonction est utilisée pour la file d'attente d'invocations asynchrones de la fonction et pas pour les files d'attente source d'événement.

Type : Carte | DeadLetterQueue

Obligatoire : non

CloudFormation compatibilité : cette propriété est similaire à celle `DeadLetterConfig` d'une `AWS::Lambda::Function` ressource. Dans CloudFormation le type est dérivé du`TargetArn`, tandis que dans AWS SAM vous devez transmettre le type avec le`TargetArn`.

`DeploymentPreference`

Les paramètres permettant d'activer les déploiements Lambda progressifs.

Si un `DeploymentPreference` objet est spécifié, AWS SAM crée un AWS::CodeDeploy::Applicationappelé `ServerlessDeploymentApplication` (un par pile), un AWS::CodeDeploy::DeploymentGroupappelé `<function-logical-id>``DeploymentGroup` et un AWS::IAM::RoleappeléCodeDeployServiceRole.

Type : DeploymentPreference

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Voir aussi : pour plus d'informations sur cette propriété, consultez Déploiement progressif d'applications sans serveur avec AWS SAM.

`Description`

Description de la fonction.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Description` propriété d'une `AWS::Lambda::Function` ressource.

`DurableConfig`

Configuration pour des fonctions durables. Permet une exécution dynamique avec des fonctionnalités de point de contrôle et de rediffusion automatiques.

Type : DurableConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Environment`

La configuration de l'environnement d'exécution.

Type : Environment

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Environment` propriété d'une `AWS::Lambda::Function` ressource.

`EphemeralStorage`

Objet qui spécifie l'espace disque, en Mo, disponible pour votre fonction Lambda dans `/tmp`.

Pour plus d'informations sur cette propriété, consultez Environnement d'exécution Lambda dans le Guide du développeur AWS Lambda .

Type : EphemeralStorage

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `EphemeralStorage` propriété d'une `AWS::Lambda::Function` ressource.

`EventInvokeConfig`

L'objet qui décrit l'événement appelle la configuration sur une fonction Lambda.

Type : EventInvokeConfiguration

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Events`

Spécifie les événements qui déclenchent cette fonction. Les événements sont constitués d'un type et d'un ensemble de propriétés qui dépendent du type.

Type : EventSource

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`FileSystemConfigs`

Liste des FileSystemConfigobjets qui spécifient les paramètres de connexion pour un système de fichiers Amazon Elastic File System (Amazon EFS).

Si votre modèle contient une ressource AWS::EFS::MountTarget, vous devez également préciser un `DependsOn` attribut pour vous assurer que la cible de montage est créée ou mise à jour avant la fonction.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `FileSystemConfigs` propriété d'une `AWS::Lambda::Function` ressource.

`FunctionName`

Nom de la fonction. Si vous ne spécifiez aucun nom, un nom unique sera généré pour vous.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `FunctionName` propriété d'une `AWS::Lambda::Function` ressource.

`FunctionScalingConfig`

Configure le comportement de dimensionnement des fonctions Lambda exécutées sur des fournisseurs de capacité. Définit le nombre minimal et maximal d'environnements d'exécution.

Type : FunctionScalingConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `FunctionScalingConfig` propriété d'une `AWS::Lambda::Function` ressource.

`FunctionUrlConfig`

Objet qui décrit l'URL d'une fonction. Une URL de fonction est un point de terminaison HTTPS que vous pouvez utiliser pour appeler votre fonction.

Pour plus d'informations, consultez la section Fonction URLs dans le guide du AWS Lambda développeur.

Type : FunctionUrlConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Handler`

La fonction dans votre code qui est appelée pour commencer l'exécution. Cette propriété n'est requise que si la propriété `PackageType` est définie sur `Zip`.

Type : chaîne

Obligatoire : Conditionnelle

CloudFormation compatibilité : cette propriété est transmise directement à la `Handler` propriété d'une `AWS::Lambda::Function` ressource.

`ImageConfig`

Objet utilisé pour configurer les paramètres d'image du conteneur Lambda. Pour plus d'informations sur l'utilisation de Lambda, consultez la section Mise en route avec Lambda du Guide du développeur AWS Lambda .

Type : ImageConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `ImageConfig` propriété d'une `AWS::Lambda::Function` ressource.

`ImageUri`

L'URI du référentiel Amazon Elastic Container Registry (Amazon ECR) pour l'image de conteneur de la fonction Lambda. Cette propriété s'applique uniquement si la propriété `PackageType` est définie sur `Image`, sinon, elle est ignorée. Pour plus d'informations, consultez la section Utilisation d'images de conteneur avec Lambda dans le Guide du développeur AWS Lambda .

> **Note**
>
> Si la `PackageType` propriété est définie sur`Image`, l'une des deux `ImageUri` est obligatoire ou vous devez créer votre application avec `Metadata` les entrées nécessaires dans le fichier AWS SAM modèle. Pour de plus amples informations, veuillez consulter Construit par défaut avec AWS SAM.

Créer votre application avec les `Metadata` nécessaires, les entrées sont prioritaires sur `ImageUri`, donc si vous précisez les deux, alors `ImageUri` sera ignoré.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `ImageUri` propriété du type de `AWS::Lambda::Function` `Code` données.

`InlineCode`

Le code de fonction Lambda qui est écrit directement dans le modèle. Cette propriété s'applique uniquement si la propriété `PackageType` est définie sur `Zip`, sinon, elle est ignorée.

> **Note**
>
> Si la propriété `PackageType` est définie sur `Zip` (par défaut), alors l'un des `CodeUri` ou `InlineCode` est requis.

Type : chaîne

Obligatoire : Conditionnelle

CloudFormation compatibilité : cette propriété est transmise directement à la `ZipFile` propriété du type de `AWS::Lambda::Function` `Code` données.

`KmsKeyArn`

L'ARN d'une clé AWS Key Management Service (AWS KMS) que Lambda utilise pour chiffrer et déchiffrer les variables d'environnement de votre fonction.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `KmsKeyArn` propriété d'une `AWS::Lambda::Function` ressource.

`Layers`

Liste des éléments `LayerVersion` ARNs que cette fonction doit utiliser. L'ordre spécifié ici est l'ordre dans lequel ils seront importés lors de l'exécution de la fonction Lambda. La version est soit un ARN complet incluant la version, soit une référence à une LayerVersion ressource. Par exemple, une référence à un `LayerVersion` will sera `!Ref MyLayer` alors qu'un ARN complet incluant la version le sera`arn:aws:lambda:`*`region`*`:`*`account-id`*`:layer:`*`layer-name`*`:`*`version`*.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Layers` propriété d'une `AWS::Lambda::Function` ressource.

`LoggingConfig`

Les paramètres de configuration Amazon CloudWatch Logs de la fonction.

Type : LoggingConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `LoggingConfig`propriété d'une `AWS::Lambda::Function` ressource.

`MemorySize`

La taille de la mémoire en Mo allouée par invocation de la fonction.

Type : entier

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `MemorySize` propriété d'une `AWS::Lambda::Function` ressource.

`PackageType`

Le package de déploiement de la fonction Lambda. Pour plus d'informations, consultez Packages de déploiement M Lambda dans le Guide du développeur AWS Lambda .

Remarques :

1. Si cette propriété est définie sur `Zip`(par défaut), alors soit `CodeUri` ou `InlineCodes`'applique, et `ImageUri` est ignoré.

2. Si cette propriété est définie sur `Image`, alors uniquement `ImageUri` s'applique, et `CodeUri` et `InlineCode` sont ignorés. Le référentiel Amazon ECR requis pour stocker l'image du conteneur de la fonction peut être créé automatiquement par le AWS SAMCLI. Pour de plus amples informations, veuillez consulter sam deploy.

Valeurs valides : `Zip` ou `Image`

Type : chaîne

Obligatoire : non

Par défaut : `Zip`

CloudFormation compatibilité : cette propriété est transmise directement à la `PackageType` propriété d'une `AWS::Lambda::Function` ressource.

`PermissionsBoundary`

L'ARN d'une limite d'autorisations à utiliser pour le rôle d'exécution de cette fonction. Cette propriété ne fonctionne que si le rôle est généré pour vous.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `PermissionsBoundary` propriété d'une `AWS::IAM::Role` ressource.

`Policies`

Politiques d'autorisation pour cette fonction. Les politiques seront ajoutées au rôle d'exécution par défaut Gestion des identités et des accès AWS (IAM) de la fonction.

Cette propriété accepte une valeur unique ou une liste de valeurs. Les valeurs autorisées sont les suivantes :

- Modèles de politique AWS SAM.
- L'ARN d'une politique gérée par AWS ou d'une politique gérée par le client.
- Le nom d'une politique AWS gérée dans la liste suivante.
- Une politique IAM en ligne formatée dans YAML sous forme de mappage.

> **Note**
>
> Si vous définissez la propriété `Role`, cette propriété est ignorée.

Type : chaîne | liste | carte

Obligatoire : non

CloudFormation compatibilité : cette propriété est similaire à celle `Policies` d'une `AWS::IAM::Role` ressource.

`PublishToLatestPublished`

Spécifie s'il faut publier la dernière version de la fonction lors de la mise à jour de la fonction.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `PublishToLatestPublished` propriété d'une `AWS::Lambda::Function` ressource.

`PropagateTags`

Indiquez s'il faut ou non transmettre les balises de la propriété `Tags` aux ressources AWS::Serverless::Function que vous avez générées. Spécifiez `True` pour la propagation des balises dans vos ressources générées.

Type : valeur booléenne

Obligatoire : non

Par défaut : `False`

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`ProvisionedConcurrencyConfig`

La configuration de la simultanéité allouée pour l'alias d'une fonction.

> **Note**
>
> `ProvisionedConcurrencyConfig` peut être spécifiée uniquement si `AutoPublishAlias` est définie. Si vous ne le faites pas, une erreur se produit.

Type : ProvisionedConcurrencyConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `ProvisionedConcurrencyConfig` propriété d'une `AWS::Lambda::Alias` ressource.

`RecursiveLoop`

État de la configuration de détection de boucle récursive de votre fonction.

Lorsque cette valeur est définie sur `Allow` et que Lambda détecte que votre fonction est invoquée dans le cadre d'une boucle récursive, aucune action n'est entreprise.

Lorsque cette valeur est définie sur `Terminate` et que Lambda détecte que votre fonction est invoquée dans le cadre d'une boucle récursive, il arrête l'appel de votre fonction et vous en informe.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `RecursiveLoop` propriété de la `AWS::Lambda::Function` ressource.

`ReservedConcurrentExecutions`

Le nombre maximum d'exécutions simultanées à réserver pour la fonction.

Pour plus d'informations sur cette propriété, consultez Mise à échelle de fonction Lambda dans le Guide du développeur AWS Lambda .

Type : entier

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `ReservedConcurrentExecutions` propriété d'une `AWS::Lambda::Function` ressource.

`Role`

L'ARN d'un rôle IAM à utiliser comme rôle d'exécution de cette fonction.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est similaire à celle `Role` d'une `AWS::Lambda::Function` ressource. Cela est obligatoire dans CloudFormation mais pas dans AWS SAM. Si aucun rôle n'est spécifié, un rôle est créé pour vous avec une ID logique de `<function-logical-id>Role`.

`RolePath`

Chemin d'accès du rôle d'exécution IAM de la fonction.

Utilisez cette propriété lorsque le rôle est généré pour vous. Ne l'utilisez pas lorsque le rôle est spécifié avec la propriété `Role`.

Type : chaîne

Obligatoire : Conditionnelle

CloudFormation compatibilité : cette propriété est transmise directement à la `Path` propriété d'une `AWS::IAM::Role` ressource.

`Runtime`

Identifiant de l'exécution de la fonction. Cette propriété n'est requise que si la propriété `PackageType` est définie sur `Zip`.

> **Note**
>
> Si vous spécifiez l'`provided`identifiant de cette propriété, vous pouvez utiliser l'attribut `Metadata` resource pour demander de AWS SAM créer le runtime personnalisé requis par cette fonction. Pour plus d'informations sur la création d'une exécution personnalisée, consultez Création de fonctions Lambda avec des environnements d'exécution personnalisés dans AWS SAM.

Type : chaîne

Obligatoire : Conditionnelle

CloudFormation compatibilité : cette propriété est transmise directement à la `Runtime` propriété d'une `AWS::Lambda::Function` ressource.

`RuntimeManagementConfig`

Configurez les options de gestion de l'exécution pour vos fonctions Lambda, telles que les mises à jour de l'environnement d'exécution, le comportement d'annulation et la sélection d'une version d'exécution spécifique. Pour en savoir plus, consultez les mises à jour du temps d'exécution Lambda dans le Guide du développeur AWS Lambda .

Type : RuntimeManagementConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `RuntimeManagementConfig` propriété d'une `AWS::Lambda::Function` ressource.

`SnapStart`

Créez un instantané de toute nouvelle version de fonction Lambda. Un instantané est un état mis en cache de votre fonction initialisée, y compris de toutes ses dépendances. La fonction n'est

initialisée qu'une seule fois et l'état mis en cache est réutilisé pour tous les appels futurs, ce qui améliore les performances de l'application en réduisant le nombre de fois que votre fonction doit être initialisée. Pour en savoir plus, consultez la section Améliorer les performances de démarrage avec Lambda SnapStart dans le guide du AWS Lambda développeur.

Type : SnapStart

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `SnapStart` propriété d'une `AWS::Lambda::Function` ressource.

`SourceKMSKeyArn`

Représente un ARN de clé KMS utilisé pour chiffrer le code de fonction ZIP du client.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `SourceKMSKeyArn` propriété d'un type de `AWS::Lambda::Function` `Code` données.

`Tags`

Un mappage (chaîne à chaîne) qui spécifie les balises ajoutées à cette fonction. Pour en savoir plus sur les clés et les valeurs valides pour les étiquettes, consultez Exigences relatives à la clé et à la valeur des étiquettes dans le guide du développeur AWS Lambda .

Lorsque la pile est créée, ajoute AWS SAM automatiquement une `lambda:createdBy:SAM` balise à cette fonction Lambda et aux rôles par défaut générés pour cette fonction.

Type: carte (map)

Obligatoire : non

CloudFormation compatibilité : cette propriété est similaire à celle `Tags` d'une `AWS::Lambda::Function` ressource. La `Tags` propriété in AWS SAM est constituée de paires clé-valeur (alors que dans CloudFormation cette propriété, elle consiste en une liste d'`Tag`objets). Ajoute également AWS SAM automatiquement une `lambda:createdBy:SAM` balise à cette fonction Lambda et aux rôles par défaut générés pour cette fonction.

`TenancyConfig`

Configuration pour le mode d'isolation des locataires Lambda. Garantit que les environnements d'exécution ne sont jamais partagés entre différents locataires IDs, offrant ainsi une isolation au niveau du calcul pour les applications multi-locataires.

Type : TenancyConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `TenancyConfig` propriété d'une `AWS::Lambda::Function` ressource.

`Timeout`

La durée maximale en secondes pendant laquelle la fonction peut s'exécuter avant qu'elle ne soit arrêtée.

Type : entier

Obligatoire : non

Par défaut : 3

CloudFormation compatibilité : cette propriété est transmise directement à la `Timeout` propriété d'une `AWS::Lambda::Function` ressource.

`Tracing`

La chaîne qui spécifie le mode de traçage X-Ray de la fonction.

- `Active` : active le suivi X-Ray pour la fonction.
- `Disabled` : désactive X-Ray pour la fonction.
- `PassThrough` : active le suivi X-Ray pour la fonction. La décision d'échantillonnage est déléguée aux services en aval.

Si défini sur `Active` ou `PassThrough` et que la propriété `Role` n'est pas spécifiée, AWS SAM ajoute la stratégie `arn:aws:iam::aws:policy/AWSXrayWriteOnlyAccess` au rôle d'exécution Lambda qu'il crée pour vous.

Pour plus d'informations sur X-Ray, consultez la section Utiliser AWS Lambda avec AWS X-Ray dans le manuel du AWS Lambda développeur.

Valeurs valides : [`Active`|`Disabled`|`PassThrough`]

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est similaire à celle `TracingConfig` d'une `AWS::Lambda::Function` ressource.

`VersionDescription`

Spécifie le champ `Description` qui est ajouté sur la nouvelle ressource de version Lambda.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Description` propriété d'une `AWS::Lambda::Version` ressource.

`VersionDeletionPolicy`

Spécifie la politique de suppression pour la ressource de version Lambda créée lorsqu'elle `AutoPublishAlias` est définie. Cela permet de contrôler si la ressource de version est conservée ou supprimée lorsque la pile est supprimée.

Valeurs valides : `Delete`, `Retain` ou `Snapshot`

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent. Il définit l'`DeletionPolicy`attribut de la `AWS::Lambda::Version` ressource générée.

`VpcConfig`

La configuration qui permet à cette fonction d'accéder aux ressources privées dans votre VPC (Virtual Private Cloud).

Type : VpcConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `VpcConfig` propriété d'une `AWS::Lambda::Function` ressource.

Valeurs renvoyées

Réf

Lorsque l'ID logique de cette ressource est fournie à la fonction intrinsèque `Ref`, elle renvoie le nom de la ressource Lambda sous-jacente.

Pour plus d'informations sur l'utilisation de la fonction `Ref`, consultez `Ref` dans le Guide de l'utilisateur AWS CloudFormation .

Ventilateur : GetAtt

`Fn::GetAtt` renvoie une valeur pour un attribut de ce type indiqué. Voici les attributs disponibles et des exemples de valeurs de retour.

Pour plus d'informations sur l'utilisation de `Fn::GetAtt`, consultez `Fn::GetAtt` dans le Guide de l'utilisateur AWS CloudFormation .

`Arn`

L'ARN de la fonction Lambda sous-jacente.

Exemples

Fonction simple

Ce qui suit est un exemple basique d'une ressource AWS::Serverless::Function du type de package `Zip` (par défaut) et du code de fonction dans un compartiment Amazon S3.

YAML

```
Type: AWS::Serverless::Function
Properties:
  Handler: index.handler
  Runtime: python3.9
  CodeUri: s3://bucket-name/key-name
```

Exemple de propriétés de fonction

Voici un exemple d'une AWS::Serverless::Function de type de package `Zip` (par défaut) qui utilise `InlineCode`, `Layers`, `Tracing`, `Policies`, `Amazon EFS`, et une source d'événement `Api`.

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 exemple

Voici un exemple d'`ImageConfig` pour une fonction Lambda de type de package `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 exemples

Une fonction Lambda configurée pour mettre à jour son environnement d'exécution en fonction du comportement actuel :

```
TestFunction
  Type: AWS::Serverless::Function
  Properties:
    ...
    Runtime: python3.9
    RuntimeManagementConfig:
      UpdateRuntimeOn: Auto
```

Une fonction Lambda configurée pour mettre à jour son environnement d'exécution lorsque la fonction est mise à jour :

```
TestFunction
  Type: AWS::Serverless::Function
  Properties:
    ...
    Runtime: python3.9
    RuntimeManagementConfig:
      UpdateRuntimeOn: FunctionUpdate
```

Une fonction Lambda configurée pour mettre à jour manuellement son environnement d'exécution :

```
TestFunction
```

```
  Type: AWS::Serverless::Function
  Properties:
    ...
    Runtime: python3.9
    RuntimeManagementConfig:
      RuntimeVersionArn: arn:aws:lambda:us-
east-1::runtime:4c459dd0104ee29ec65dcad056c0b3ddbe20d6db76b265ade7eda9a066859b1e
      UpdateRuntimeOn: Manual
```

SnapStart exemples

Exemple de fonction Lambda SnapStart activée pour les futures versions :

```
TestFunc
  Type: AWS::Serverless::Function
  Properties:
    ...
    SnapStart:
      ApplyOn: PublishedVersions
```

TenancyConfig exemples

Exemple de fonction Lambda avec le mode d'isolation des locataires activé :

```
TestFunction
  Type: AWS::Serverless::Function
  Properties:
    ...
    TenancyConfig:
      TenantIsolationMode: PER_TENANT
```

DeadLetterQueue

Spécifie une file d'attente SQS ou une rubrique SNS à laquelle (AWS Lambda Lambda) envoie des événements lorsqu'elle ne peut pas les traiter. Pour en savoir plus sur la fonctionnalité de file d'attente de lettres mortes, consultez File d'attente de lettres mortes dans le Guide du développeur AWS Lambda .

SAM ajoutera automatiquement l'autorisation appropriée à votre rôle d'exécution de fonction Lambda pour permettre au service Lambda d'accéder à la ressource. sqs : SendMessage sera ajouté pour les files d'attente SQS et SNS:Publish pour les rubriques SNS.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
TargetArn: String
Type: String
```

Propriétés

`TargetArn`

L'Amazon Resource Name (ARN) d'une file d'attente Amazon SQS ou d'une rubrique Amazon SNS.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `TargetArn` propriété du type de `AWS::Lambda::Function` `DeadLetterConfig` données.

`Type`

Le type de file d'attente de lettres mortes.

Valeurs valides : `SNS`, `SQS`

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

DeadLetterQueue

Exemple de file d'attente de lettres mortes pour une rubrique SNS.

YAML

```
DeadLetterQueue:
  Type: SNS
  TargetArn: arn:aws:sns:us-east-2:123456789012:my-topic
```

DeploymentPreference

Spécifie les configurations permettant d'activer les déploiements Lambda progressifs. Pour plus d'informations sur les configurations de déploiement progressif Lambda, consultez Déploiement progressif d'applications sans serveur avec AWS SAM.

> **Note**
>
> Vous devez spécifier un `AutoPublishAlias` dans votre fonction AWS::Serverless::Function pour utiliser un objet `DeploymentPreference`. Si vous ne le faites pas, une erreur se produit.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
  Alarms: List
  Enabled: Boolean
  Hooks: Hooks
  PassthroughCondition: Boolean
  Role: String
  TriggerConfigurations: List
  Type: String
```

Propriétés

`Alarms`

Liste des CloudWatch alarmes que vous souhaitez déclencher en cas d'erreur provoquée par le déploiement.

Cette propriété accepte la fonction intrinsèque `Fn::If`. Reportez-vous à la section Exemples au bas de cette rubrique pour obtenir un exemple de modèle qui utilise `Fn::If`.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Enabled`

Indique si cette préférence de déploiement est activée.

Type : valeur booléenne

Obligatoire : non

Valeur par défaut : VRAI

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Hooks`

Validation des fonctions Lambda qui sont exécutées avant et après le déplacement du trafic.

Type : crochets

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`PassthroughCondition`

Si la valeur est True, et si cette préférence de déploiement est activée, la condition de la fonction sera transmise à la CodeDeploy ressource générée. En règle générale, vous devez définir cette valeur sur True (Vrai). Dans le cas contraire, la CodeDeploy ressource serait créée même si la condition de la fonction prend la valeur False.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Role`

Un ARN de rôle IAM qui CodeDeploy sera utilisé pour le transfert du trafic. Un rôle IAM ne sera pas créé si cela est fourni.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`TriggerConfigurations`

Une liste des configurations de déclenchement que vous souhaitez associer au groupe de déploiement. Utilisé pour notifier une rubrique SNS sur les événements du cycle de vie.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `TriggerConfigurations` propriété d'une `AWS::CodeDeploy::DeploymentGroup` ressource.

`Type`

Il existe actuellement deux catégories de types de déploiement : Linéaire et Canary. Pour plus d'informations sur les types de déploiements, consultez Déploiement progressif d'applications sans serveur avec AWS SAM.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

DeploymentPreference avec des crochets avant et après le trafic.

Exemple de préférence de déploiement contenant des crochets pré- et post-trafic.

YAML

```
DeploymentPreference:
  Enabled: true
  Type: Canary10Percent10Minutes
  Alarms:
    - !Ref: AliasErrorMetricGreaterThanZeroAlarm
    - !Ref: LatestVersionErrorMetricGreaterThanZeroAlarm
  Hooks:
    PreTraffic:
      !Ref: PreTrafficLambdaFunction
    PostTraffic:
      !Ref: PostTrafficLambdaFunction
```

DeploymentPreference avec fonction intrinsèque Fn : :If

Exemple de préférence de déploiement qui utilise `Fn::If` pour configurer les alarmes. Dans cet exemple,`Alarm1` sera configuré si `MyCondition` est `true`, et `Alarm2` et `Alarm5` seront configurés si `MyCondition` est `false`.

YAML

```
DeploymentPreference:
  Enabled: true
  Type: Canary10Percent10Minutes
  Alarms:
    Fn::If:
      - MyCondition
      - - Alarm1
      - - Alarm2
        - Alarm5
```

Hooks

Validation des fonctions Lambda qui sont exécutées avant et après le déplacement du trafic.

> **Note**
>
> Les fonctions Lambda référencées dans cette propriété configurent l'objet `CodeDeployLambdaAliasUpdate` de la ressource AWS::Lambda::Alias qui en résulte. Pour plus d'informations, consultez la section CodeDeployLambdaAliasUpdate Politique du guide de AWS CloudFormation l'utilisateur.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
PostTraffic: String
PreTraffic: String
```

Propriétés

`PostTraffic`

Fonction Lambda qui est exécutée après le déplacement du trafic.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`PreTraffic`

Fonction Lambda qui est exécutée avant le déplacement du trafic.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

Crochets

Exemple de fonctions de crochet

YAML

```
Hooks:
  PreTraffic:
    Ref: PreTrafficLambdaFunction
  PostTraffic:
    Ref: PostTrafficLambdaFunction
```

DurableConfig

Configure les paramètres d'exécution durables des AWS Lambda fonctions. Les fonctions durables peuvent fonctionner jusqu'à un an et contrôler automatiquement la progression des points de travail, ce qui permet des flux de travail de longue durée et des applications tolérantes aux pannes. Pour plus d'informations sur les fonctions durables, consultez la section Fonctions durables Lambda dans le Guide du AWS Lambda développeur.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
  ExecutionTimeout: Integer
  RetentionPeriodInDays: Integer
```

Propriétés

`ExecutionTimeout`

Durée (en secondes) pendant laquelle Lambda permet à une fonction durable de s'exécuter avant de l'arrêter. Le maximum est d'une année de 366 jours ou 31 622 400 secondes.

Type : entier

Obligatoire : oui

Minimum : 1

Maximum : 31622400

CloudFormation compatibilité : cette propriété est transmise directement à la `ExecutionTimeout` propriété du type de `AWS::Lambda::Function` `DurableConfig` données.

`RetentionPeriodInDays`

Le nombre de jours après la clôture d'une exécution durable pendant lesquels Lambda conserve son historique, de un à 90 jours. La durée par défaut est de 14 jours.

Type : entier

Obligatoire : non

Par défaut : 14

Minimum : 1

Maximum : 90

CloudFormation compatibilité : cette propriété est transmise directement à la `RetentionPeriodInDays` propriété du type de `AWS::Lambda::Function` `DurableConfig` données.

Exemples

DurableConfig

Exemple de configuration durable pour une fonction avec un délai d'exécution d'une heure et une période de rétention de 7 jours.

YAML

```
DurableConfig:
  ExecutionTimeout: 3600
  RetentionPeriodInDays: 7
```

EventInvokeConfiguration

Options de configuration pour les appels Lambda Alias ou Version Asynchrones.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
DestinationConfig: EventInvokeDestinationConfiguration
MaximumEventAgeInSeconds: Integer
MaximumRetryAttempts: Integer
```

Propriétés

`DestinationConfig`

Objet de configuration qui spécifie la destination d'un événement après son traitement par Lambda.

Type : EventInvokeDestinationConfiguration

Obligatoire : non

CloudFormation compatibilité : cette propriété est similaire à celle `DestinationConfig` d'une `AWS::Lambda::EventInvokeConfig` ressource. SAM nécessite un paramètre supplémentaire, « Type », qui n'existe pas dans CloudFormation.

`MaximumEventAgeInSeconds`

Âge maximum d'une demande que Lambda envoie à une fonction pour traitement.

Type : entier

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `MaximumEventAgeInSeconds` propriété d'une `AWS::Lambda::EventInvokeConfig` ressource.

`MaximumRetryAttempts`

Nombre maximal de tentatives autorisées avant que la fonction renvoie une erreur.

Type : entier

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `MaximumRetryAttempts` propriété d'une `AWS::Lambda::EventInvokeConfig` ressource.

Exemples

MaximumEventAgeInSeconds

MaximumEventAgeInSeconds exemple

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

Objet de configuration qui spécifie la destination d'un événement après son traitement par Lambda.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
  OnFailure: OnFailure
```

```
  OnSuccess: OnSuccess
```

Propriétés

`OnFailure`

Destination des événements dont le traitement a échoué.

Type : OnFailure

Obligatoire : non

CloudFormation compatibilité : cette propriété est similaire à celle `OnFailure` d'une `AWS::Lambda::EventInvokeConfig` ressource. Nécessite `Type`, une propriété SAM-only supplémentaire.

`OnSuccess`

Destination des événements traités avec succès.

Type : OnSuccess

Obligatoire : non

CloudFormation compatibilité : cette propriété est similaire à celle `OnSuccess` d'une `AWS::Lambda::EventInvokeConfig` ressource. Nécessite `Type`, une propriété SAM-only supplémentaire.

Exemples

OnSuccess

OnSuccess exemple

YAML

```
EventInvokeConfig:
  DestinationConfig:
    OnSuccess:
      Type: SQS
      Destination: arn:aws:sqs:us-west-2:012345678901:my-queue
    OnFailure:
```

```
      Type: Lambda
      Destination: !GetAtt DestinationLambda.Arn
```

OnFailure

Destination des événements dont le traitement a échoué.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
Destination: String
Type: String
```

Propriétés

`Destination`

Amazon Resource Name (ARN) de la ressource de destination.

Type : chaîne

Obligatoire : Conditionnelle

CloudFormation compatibilité : cette propriété est similaire à celle `OnFailure` d'une `AWS::Lambda::EventInvokeConfig` ressource. SAM ajoute toutes les autorisations nécessaires au rôle IAM généré automatiquement associé à cette fonction pour accéder à la ressource référencée dans cette propriété.

Remarques supplémentaires : Si le type est Lambda/EventBridge, Destination est obligatoire.

`Type`

Type de la ressource référencée dans la destination. Les types pris en charge sont `SQS` `SNSS3`,`Lambda`, et`EventBridge`.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Remarques supplémentaires : Si le type est SQS/SNS et que la `Destination` propriété est laissée vide, la SQS/SNS ressource est générée automatiquement par SAM. Pour référencer la ressource, utilisez `<function-logical-id>.DestinationQueue` pour SQS ou `<function-logical-id>.DestinationTopic` pour SNS. Si le type est Lambda/EventBridge, `Destination` c'est obligatoire.

Exemples

EventInvoke Exemple de configuration avec des destinations SQS et Lambda

Dans cet exemple, aucune destination n'est donnée pour la OnSuccess configuration SQS. SAM crée donc implicitement une file d'attente SQS et ajoute les autorisations nécessaires. Dans cet exemple également, une destination pour une ressource Lambda déclarée dans le fichier modèle est spécifiée dans la OnFailure configuration. SAM ajoute donc les autorisations nécessaires à cette fonction Lambda pour appeler la fonction Lambda de destination.

YAML

```
EventInvokeConfig:
  DestinationConfig:
    OnSuccess:
      Type: SQS
    OnFailure:
      Type: Lambda
      Destination: !GetAtt DestinationLambda.Arn  # Arn of a Lambda function declared
 in the template file.
```

EventInvoke Exemple de configuration avec destination SNS

Dans cet exemple, une destination est donnée pour une rubrique SNS déclarée dans le fichier modèle de OnSuccess configuration.

YAML

```
EventInvokeConfig:
  DestinationConfig:
    OnSuccess:
```

```
      Type: SNS
      Destination:
        Ref: DestinationSNS       # Arn of an SNS topic declared in the tempate file
```

OnSuccess

Destination des événements traités avec succès.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
Destination: String
Type: String
```

Propriétés

`Destination`

L'Amazon Resource Name (ARN) de la ressource de destination.

Type : chaîne

Obligatoire : Conditionnelle

CloudFormation compatibilité : cette propriété est similaire à celle `OnSuccess` d'une `AWS::Lambda::EventInvokeConfig` ressource. SAM ajoute toutes les autorisations nécessaires au rôle IAM généré automatiquement associé à cette fonction pour accéder à la ressource référencée dans cette propriété.

Remarques supplémentaires : Si le type est Lambda/EventBridge, Destination est obligatoire.

`Type`

Type de la ressource référencée dans la destination. Les types pris en charge sont `SQS` `SNSS3`,`Lambda`, et`EventBridge`.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Remarques supplémentaires : Si le type est SQS/SNS et que la `Destination` propriété est laissée vide, la SQS/SNS ressource est générée automatiquement par SAM. Pour référencer la ressource, utilisez `<function-logical-id>.DestinationQueue` pour SQS ou `<function-logical-id>.DestinationTopic` pour SNS. Si le type est Lambda/EventBridge, `Destination` c'est obligatoire.

Exemples

EventInvoke Exemple de configuration avec des destinations SQS et Lambda

Dans cet exemple, aucune destination n'est donnée pour la OnSuccess configuration SQS. SAM crée donc implicitement une file d'attente SQS et ajoute les autorisations nécessaires. Dans cet exemple également, une destination pour une ressource Lambda déclarée dans le fichier modèle est spécifiée dans la OnFailure configuration. SAM ajoute donc les autorisations nécessaires à cette fonction Lambda pour appeler la fonction Lambda de destination.

YAML

```
EventInvokeConfig:
  DestinationConfig:
    OnSuccess:
      Type: SQS
    OnFailure:
      Type: Lambda
      Destination: !GetAtt DestinationLambda.Arn  # Arn of a Lambda function declared
 in the template file.
```

EventInvoke Exemple de configuration avec destination SNS

Dans cet exemple, une destination est donnée pour une rubrique SNS déclarée dans le fichier modèle de OnSuccess configuration.

YAML

```
EventInvokeConfig:
```

```
  DestinationConfig:
    OnSuccess:
      Type: SNS
      Destination:
        Ref: DestinationSNS       # Arn of an SNS topic declared in the tempate file
```

EventSource

L'objet décrivant la source des événements qui déclenchent la fonction. Chaque événement se compose d'un type et d'un ensemble de propriétés qui dépendent de ce type. Pour plus d'informations sur les propriétés de chaque source d'événement, consultez la rubrique correspondant à ce type.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
 Properties: AlexaSkill | Api | CloudWatchEvent | CloudWatchLogs | Cognito
| DocumentDB | DynamoDB | EventBridgeRule | HttpApi | IoTRule | Kinesis | MQ | MSK
| S3 | Schedule | ScheduleV2 | SelfManagedKafka | SNS | SQS
 Type: String
```

Propriétés

`Properties`

Objet décrivant les propriétés de ce mappage de cet événement. L'ensemble de propriétés doit être conforme au Type défini.

Type : AlexaSkill| Api | | CloudWatchEvent| Cognito CloudWatchLogs| DocumentDB | DynamoDB | | | Io | Kinesis | MQ EventBridgeRule HttpApi ???| MSK TRule | S3 | Planification| ScheduleV2 | | SNS | SQS SelfManagedKafka

Obligatoire : oui

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Type

Type d'événement.

Valeurs valides : `AlexaSkill`, `Api`, `CloudWatchEvent`, `CloudWatchLogs`, `Cognito`, `DocumentDB`, `DynamoDB`, `EventBridgeRule`, `HttpApi`, `IoTRule`, `Kinesis`, `MQ`, `MSK`, `S3`, `Schedule`, `ScheduleV2`, `SelfManagedKafka`, `SNS`, `SQS`

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

APIEvent

Exemple d'utilisation d'un événement API

YAML

```
ApiEvent:
  Type: Api
  Properties:
    Method: get
    Path: /group/{user}
    RestApiId:
      Ref: MyApi
```

AlexaSkill

L'objet décrivant un type de source d'événement `AlexaSkill`.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
  SkillId: String
```

Propriétés

`SkillId`

L'Alexa Skill ID (ID de compétence Alexa) pour votre compétence Alexa. Pour plus d'informations sur l'ID de compétence, consultez Configurer le déclencheur pour une fonction Lambda dans la documentation du Kit de compétences Alexa.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

AlexaSkillTrigger

Exemple d'événement de compétence Alexa

YAML

```
AlexaSkillEvent:
  Type: AlexaSkill
```

Api

L'objet décrivant un type de source d'événement `Api`. Si une ressource AWS::Serverless::Api est définie, les valeurs de chemin d'accès et de méthode doivent correspondre à une opération dans la définition OpenAPI de l'API.

Si aucune AWS::Serverless::Api n'est définie, l'entrée et la sortie de la fonction sont une représentation de la demande HTTP et de la réponse HTTP.

Par exemple, à l'aide de l' JavaScript API, le code d'état et le corps de la réponse peuvent être contrôlés en renvoyant un objet avec les clés statusCode et body.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```yaml
Auth: ApiFunctionAuth
Method: String
Path: String
RequestModel: RequestModel
RequestParameters: List of [ String | RequestParameter ]
RestApiId: String
ResponseTransferMode: String
TimeoutInMillis: Integer
```

Propriétés

`Auth`

Configuration d'authentification pour cette méthode Api+Chemin+Méthode spécifique.

Utile pour remplacer la configuration d'autorisation du paramètre de `DefaultAuthorizer` de l'API sur un chemin d'accès individuel, lorsqu'aucun `DefaultAuthorizer` n'est spécifié, ou pour remplacer le paramètre `ApiKeyRequired` par défaut.

Type : ApiFunctionAuth

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Method`

Méthode HTTP pour laquelle cette fonction est appelée. Les options incluent `DELETEGET`,`HEAD`,`OPTIONS`,`PATCH`,`POST`,`PUT`, et`ANY`. Reportez-vous à la section Configurer une méthode HTTP dans le guide du développeur d'API Gateway pour plus de détails.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Path`

Chemin d'accès d'URI pour lequel cette fonction est appelée. Doit commencer par `/`.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`RequestModel`

Demander un modèle à utiliser pour cette formule Api+Chemin+Méthode spécifique. Cela devrait faire référence au nom d'un modèle spécifié dans la section `Models` d'une ressource AWS::Serverless::Api.

Type : RequestModel

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`RequestParameters`

Configuration des paramètres de demande pour cette formule Api+Chemin+Méthode spécifique. Tous les noms de paramètres doivent commencer par `method.request` et doivent être limités à `method.request.header`, `method.request.querystring`, ou `method.request.path`.

Une liste peut contenir à la fois des chaînes de noms de paramètres et RequestParameterdes objets. Pour les chaînes, les propriétés `Required` et `Caching` prendront par défaut la valeur `false`.

Type : Liste de [String | RequestParameter]

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`RestApiId`

Identifiant d'une RestApi ressource, qui doit contenir une opération avec le chemin et la méthode donnés. Cela est généralement défini pour faire référence à une ressource AWS::Serverless::Api, qui est définie dans ce modèle.

Si vous ne définissez pas cette propriété, AWS SAM crée une AWS::Serverless::Api ressource par défaut à l'aide d'un `OpenApi` document généré. Cette ressource contient une union de tous les chemins et méthodes définis par `Api` dans le même modèle qui ne spécifient pas un `RestApiId`.

Cela ne peut pas référencer une ressource AWS::Serverless::Api définie dans un autre modèle.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`ResponseTransferMode`

Mode de transfert de réponse pour l'intégration de la fonction Lambda. Définissez cette valeur `RESPONSE_STREAM` pour activer le streaming des réponses Lambda via API Gateway, ce qui permet à la fonction de renvoyer les réponses aux clients. Lorsqu'il est défini sur`RESPONSE_STREAM`, API Gateway utilise l'API Lambda InvokeWithResponseStreaming .

Type : chaîne

Obligatoire : non

Valeurs valides : `BUFFERED` | `RESPONSE_STREAM`

CloudFormation compatibilité : Cette propriété est transmise directement à la `ResponseTransferMode`propriété d'un`AWS::ApiGateway::Method Integration`.

`TimeoutInMillis`

Délai d'attente personnalisé compris entre 50 et 29 000 millisecondes.

> **Note**
>
> Lorsque vous spécifiez cette propriété, cela AWS SAM modifie votre définition OpenAPI. La définition OpenAPI doit être spécifiée en ligne à l'aide de la propriété `DefinitionBody`.

Type : entier

Obligatoire : non

Par défaut : 29 000 millisecondes ou 29 secondes

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

Exemple de base

YAML

```
Events:
  ApiEvent:
    Type: Api
    Properties:
      Path: /path
      Method: get
      RequestParameters:
        - method.request.header.Authorization
        - method.request.querystring.keyword:
            Required: true
            Caching: false
```

ApiFunctionAuth

Configure l'autorisation au niveau de l'événement, pour une API, un chemin et une méthode spécifiques.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
  ApiKeyRequired: Boolean
  AuthorizationScopes: List
  Authorizer: String
  InvokeRole: String
```

```
  OverrideApiAuth: Boolean
  ResourcePolicy: ResourcePolicyStatement
```

Propriétés

`ApiKeyRequired`

Nécessite une clé API pour cette API, ce chemin d'accès et cette méthode.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`AuthorizationScopes`

Les étendues d'autorisation à appliquer à cette API, ce chemin d'accès et cette méthode.

Les étendues que vous spécifiez remplaceront toutes les étendues appliquées par la propriété `DefaultAuthorizer` si vous l'avez spécifiée.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Authorizer`

Le `Authorizer` pour une fonction spécifique.

Si un mécanisme d'autorisation global est spécifié pour votre ressource `AWS::Serverless::Api`, vous pouvez remplacer le mécanisme d'autorisation en définissant `Authorizer` sur `NONE`. Pour obtenir un exemple, consultez Remplacer un mécanisme d'autorisation global pour votre API REST Amazon API Gateway.

> **Note**
>
> Si vous utilisez la propriété `DefinitionBody` d'une ressource `AWS::Serverless::Api` pour décrire votre API, vous devez utiliser

`OverrideApiAuth` avec `Authorizer` pour remplacer votre mécanisme d'autorisation global. Pour plus d'informations, consultez `OverrideApiAuth`.

Valeurs valides : `AWS_IAMNONE`, ou l'identifiant logique de tout autorisateur défini dans votre AWS SAM modèle.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`InvokeRole`

Spécifie le `InvokeRole` à utiliser pour l'autorisation `AWS_IAM`.

Type : chaîne

Obligatoire : non

Par défaut : `CALLER_CREDENTIALS`

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Informations complémentaires : `CALLER_CREDENTIALS` mappe avec `arn:aws:iam::`*`:<user>/`*, qui utilise les informations d'identification de l'appelant pour appeler le point de terminaison.

`OverrideApiAuth`

Définissez-la sur `true` pour remplacer la configuration du mécanisme d'autorisation global de votre ressource `AWS::Serverless::Api`. Cette propriété n'est requise que si vous spécifiez un mécanisme d'autorisation global et que vous utilisez la propriété `DefinitionBody` d'une ressource `AWS::Serverless::Api` pour décrire votre API.

> **Note**
>
> Lorsque vous spécifiez `OverrideApiAuth` commetrue, AWS SAM cela remplacera votre autorisateur global par toutes les valeurs fournies pour `ApiKeyRequiredAuthorizer`, ou. `ResourcePolicy` C'est pourquoi au moins une de

ces propriétés doit également être spécifiée lors de l'utilisation de `OverrideApiAuth`. Pour obtenir un exemple, consultez Remplacer un autorisateur global lorsque DefinitionBody for est spécifié AWS::Serverless::Api.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`ResourcePolicy`

Configurez la stratégie de ressources pour ce chemin d'accès sur une API.

Type : ResourcePolicyStatement

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

Fonction-Auth (Autorisation fonction)

L'exemple suivant spécifie l'autorisation au niveau de la fonction.

YAML

```
Auth:
  ApiKeyRequired: true
  Authorizer: NONE
```

Remplacer un mécanisme d'autorisation global pour votre API REST Amazon API Gateway

Vous pouvez spécifier un mécanisme d'autorisation global pour votre ressource `AWS::Serverless::Api`. Voici un exemple de configuration d'un mécanisme d'autorisation global par défaut :

```
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
```

Pour remplacer l'autorisateur par défaut pour votre AWS Lambda fonction, vous pouvez spécifier `Authorizer` comme. `NONE` Voici un exemple :

```
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
```

Remplacer un autorisateur global lorsque DefinitionBody for est spécifié AWS::Serverless::Api

Lorsque vous utilisez la propriété `DefinitionBody` pour décrire votre ressource `AWS::Serverless::Api`, la méthode de remplacement précédente ne fonctionne pas. Voici un exemple d'utilisation de la propriété `DefinitionBody` pour une ressource `AWS::Serverless::Api` :

```
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
```

Pour annuler le mécanisme d'autorisation global, utilisez la propriété `OverrideApiAuth`. Voici un exemple qui utilise `OverrideApiAuth` pour remplacer le mécanisme d'autorisation global par la valeur fournie pour `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

Configure une stratégie de ressource pour toutes les méthodes et chemins d'accès d'une API. Pour plus d'informations sur les stratégies de ressources, consultez Contrôle de l'accès à une API avec des stratégies de ressources API Gateway dans le Guide du développeur API.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
AwsAccountBlacklist: List
AwsAccountWhitelist: List
CustomStatements: List
IntrinsicVpcBlacklist: List
IntrinsicVpcWhitelist: List
IntrinsicVpceBlacklist: List
IntrinsicVpceWhitelist: List
IpRangeBlacklist: List
IpRangeWhitelist: List
SourceVpcBlacklist: List
SourceVpcWhitelist: List
```

Propriétés

`AwsAccountBlacklist`

Les AWS comptes à bloquer.

Type : liste de chaînes

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`AwsAccountWhitelist`

Les AWS comptes à autoriser. Pour obtenir un exemple d'utilisation de cette propriété, consultez la section Exemples en bas de cette page.

Type : liste de chaînes

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`CustomStatements`

Une liste des instructions de stratégie de ressource personnalisées à appliquer à cette API. Pour obtenir un exemple d'utilisation de cette propriété, consultez la section Exemples en bas de cette page.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`IntrinsicVpcBlacklist`

La liste des clouds privés virtuels (VPCs) à bloquer, où chaque VPC est spécifié comme référence, telle qu'une référence dynamique ou une fonction `Ref` intrinsèque. Pour obtenir un exemple d'utilisation de cette propriété, consultez la section Exemples en bas de cette page.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`IntrinsicVpcWhitelist`

La liste des VPCs à autoriser, où chaque VPC est spécifié comme référence, telle qu'une référence dynamique ou une fonction `Ref` intrinsèque.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`IntrinsicVpceBlacklist`

Liste des points de terminaison VPC à bloquer, où chaque point de terminaison VPC est spécifié comme une référence telle qu'une Référence dynamique ou la fonction intrinsèque `Ref`.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`IntrinsicVpceWhitelist`

Liste des points de terminaison VPC à autoriser, où chaque point de terminaison VPC est spécifié en tant que référence telle qu'une Référence dynamique ou la fonction intrinsèque `Ref`. Pour obtenir un exemple d'utilisation de cette propriété, consultez la section Exemples en bas de cette page.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`IpRangeBlacklist`

L'adresse IP ou les plages d'adresses à bloquer. Pour obtenir un exemple d'utilisation de cette propriété, consultez la section Exemples en bas de cette page.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`IpRangeWhitelist`

L'adresse IP ou les plages d'adresses à autoriser.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`SourceVpcBlacklist`

Les points de terminaison VPC ou VPC source à bloquer. Les noms de VPC source doivent commencer par `"vpc-"` et les noms de point de terminaison VPC source doivent commencer par `"vpce-"`. Pour obtenir un exemple d'utilisation de cette propriété, consultez la section Exemples en bas de cette page.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`SourceVpcWhitelist`

Les points de terminaison VPC ou VPC source à autoriser. Les noms de VPC source doivent commencer par `"vpc-"` et les noms de point de terminaison VPC source doivent commencer par `"vpce-"`.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

Exemples de stratégie basée sur les ressources

L'exemple suivant bloque deux adresses IP et un VPC source, et autorise un AWS compte.

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

Configure un modèle de demande pour une formule Api+Chemin+Méthode spécifique.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
Model: String
Required: Boolean
ValidateBody: Boolean
ValidateParameters: Boolean
```

Propriétés

`Model`

Nom d'un modèle défini dans la propriété Modèles de la AWS::Serverless::Api.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Required`

Ajoute une `required` propriété dans la section des paramètres de la OpenApi définition pour le point de terminaison d'API donné.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`ValidateBody`

Indique si API Gateway utilise le `Model` pour valider le corps de la demande. Pour de plus amples informations, veuillez consulter Activer la validation de demande de base pour une API sur API Gateway dans le Manuel du développeur API Gateway.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`ValidateParameters`

Spécifie si API Gateway utilise le `Model` pour valider les paramètres du chemin de la requête, les chaînes de requête et les en-têtes. Pour de plus amples informations, veuillez consulter Activer la validation de demande de base pour une API sur API Gateway dans le Manuel du développeur API Gateway.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

Modèle de demande

Exemple de modèle de demande

YAML

```
RequestModel:
  Model: User
  Required: true
  ValidateBody: true
  ValidateParameters: true
```

RequestParameter

Configurez le paramètre de demande pour une formule Api+Chemin+Méthode spécifique.

La propriété `Required` ou `Caching` doit être spécifiée pour le paramètre de demande

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
Caching: Boolean
Required: Boolean
```

Propriétés

`Caching`

Ajoute une `cacheKeyParameters` section à la OpenApi définition d'API Gateway

Type : valeur booléenne

Obligatoire : Conditionnelle

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Required`

Ce champ indique si un paramètre est nécessaire.

Type : valeur booléenne

Obligatoire : Conditionnelle

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

Paramètre de demande

Exemple de définition des paramètres de demande

YAML

```
RequestParameters:
  - method.request.header.Authorization:
      Required: true
      Caching: true
```

CloudWatchEvent

L'objet décrivant un type de source d'événement `CloudWatchEvent`.

AWS Serverless Application Model (AWS SAM) génère une AWS::Events::Ruleressource lorsque ce type d'événement est défini.

Remarque importante : EventBridgeRule c'est le type de source d'événements préféré à utiliser, à la place de`CloudWatchEvent`. `EventBridgeRule`et `CloudWatchEvent` utilisez le même service, la même API et les mêmes CloudFormation ressources sous-jacents. Cependant, AWS SAM ajoutera la prise en charge des nouvelles fonctionnalités uniquement à`EventBridgeRule`.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
Enabled: Boolean
EventBusName: String
Input: String
InputPath: String
Pattern: EventPattern
State: String
```

Propriétés

`Enabled`

Indique si la règle est activée.

Pour désactiver la règle, définissez cette propriété sur `false`.

> **Note**
>
> Spécifiez la propriété `Enabled` ou la propriété `State`, mais pas les deux.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est similaire à celle `State` d'une `AWS::Events::Rule` ressource. Si cette propriété est définie sur `true` then`ENABLED`, elle est AWS SAM transmise dans le cas contraire`DISABLED`.

`EventBusName`

Le bus d'événements à associer à cette règle. Si vous omettez cette propriété, AWS SAM utilise le bus d'événements par défaut.

Type : chaîne

Obligatoire : non

Par défaut : bus d'événement par défaut

CloudFormation compatibilité : cette propriété est transmise directement à la `EventBusName` propriété d'une `AWS::Events::Rule` ressource.

`Input`

Texte JSON valide transmis à la cible. Si vous utilisez cette propriété, aucun élément du texte de l'événement lui-même n'est transmis à la cible.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Input` propriété d'une `AWS::Events::Rule Target` ressource.

`InputPath`

Lorsque vous ne voulez pas transmettre l'événement correspondant complet, utilisez la propriété `InputPath` pour décrire quelles parties de l'événement transmettre à la cible.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `InputPath` propriété d'une `AWS::Events::Rule Target` ressource.

`Pattern`

Décrit les événements qui sont acheminés vers la cible spécifiée. Pour plus d'informations, consultez la section Événements et modèles d'événements EventBridge dans le guide de EventBridge l'utilisateur Amazon.

Type : EventPattern

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `EventPattern` propriété d'une `AWS::Events::Rule` ressource.

`State`

État de la règle.

Valeurs acceptées : `DISABLED | ENABLED`

> **Note**
>
> Spécifiez la propriété `Enabled` ou la propriété `State`, mais pas les deux.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `State` propriété d'une `AWS::Events::Rule` ressource.

Exemples

CloudWatchEvent

Voici un exemple de type de source d'événement `CloudWatchEvent`.

YAML

```
CWEvent:
  Type: CloudWatchEvent
  Properties:
```

```
    Enabled: false
    Input: '{"Key": "Value"}'
    Pattern:
      detail:
        state:
          - running
```

CloudWatchLogs

L'objet décrivant un type de source d'événement `CloudWatchLogs`.

Cet événement génère une ressource AWS::Logs::SubscriptionFilter, puis précise un filtre d'abonnement et l'associe au groupe de journaux indiqué.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
FilterPattern: String
LogGroupName: String
```

Propriétés

`FilterPattern`

Les expressions de filtrage qui limitent ce qui est livré à la AWS ressource de destination. Pour plus d'informations sur la syntaxe de modèle de filtre, consultez Syntaxe de filtre et de modèle.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `FilterPattern` propriété d'une `AWS::Logs::SubscriptionFilter` ressource.

`LogGroupName`

Groupe de journaux auquel associer le filtre d'abonnement. Tous les événements de journal chargés dans ce groupe de journaux sont filtrés et transmis à la AWS ressource spécifiée si le modèle de filtre correspond aux événements du journal.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `LogGroupName` propriété d'une `AWS::Logs::SubscriptionFilter` ressource.

Exemples

Filtres d'abonnement dans les journaux CloudWatch Logs

Exemple de filtres d'abonnement dans les journaux CloudWatch Logs

YAML

```
CWLog:
  Type: CloudWatchLogs
  Properties:
    LogGroupName:
      Ref: CloudWatchLambdaLogsGroup
    FilterPattern: My pattern
```

Cognito

L'objet décrivant un type de source d'événement `Cognito`.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
  Trigger: List
  UserPool: String
```

Propriétés

`Trigger`

Informations relatives à la configuration des déclencheurs Lambda pour le nouveau pool d'utilisateurs.

Type : liste

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `LambdaConfig` propriété d'une `AWS::Cognito::UserPool` ressource.

`UserPool`

Référence à UserPool définie dans le même modèle

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

Événement Cognito

Exemple d'événement Cognito

YAML

```
CognitoUserPoolPreSignup:
  Type: Cognito
  Properties:
    UserPool:
      Ref: MyCognitoUserPool
    Trigger: PreSignUp
```

DocumentDB

L'objet décrivant un type de source d'événement `DocumentDB`. Pour plus d'informations, consultez la section Utilisation AWS Lambda avec Amazon DocumentDB dans le manuel du AWS Lambda développeur.

Syntaxe

Pour déclarer cette entité dans votre AWS SAM modèle, utilisez la syntaxe suivante.

YAML

```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
```

Propriétés

`BatchSize`

Nombre maximum d'éléments à récupérer dans un seul lot.

Type : entier

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `BatchSize` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`Cluster`

Amazon Resource Name (ARN) du cluster Amazon DocumentDB.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `EventSourceArn` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`CollectionName`

Le nom de la collection à consommer dans la base de données. Si vous ne spécifiez pas de collection, Lambda consomme toutes les collections.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `CollectionName` propriété d'un type de `AWS::Lambda::EventSourceMapping` `DocumentDBEventSourceConfig` données.

`DatabaseName`

Le nom de la base de données à consommer dans le cluster Amazon DocumentDB.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `DatabaseName` propriété d'un type de `AWS::Lambda::EventSourceMapping` `DocumentDBEventSourceConfig` données.

`Enabled`

Si la valeur est `true`, le mappage de source d'événement est actif. Pour suspendre l'interrogation et l'appel, définissez ce paramètre sur `false`.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Enabled` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`FilterCriteria`

Un objet qui définit les critères permettant de déterminer si Lambda doit traiter un événement. Pour de plus amples informations, veuillez consulter le filtrage d'événements Lambda dans le Guide du développeur AWS Lambda .

Type : FilterCriteria

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `FilterCriteria` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`FullDocument`

Détermine ce qu'Amazon DocumentDB envoie à votre flux d'événements lors des opérations de mise à jour des documents. S'il est défini sur `UpdateLookup`, Amazon DocumentDB envoie un delta décrivant les modifications, ainsi qu'une copie de l'intégralité du document. Dans le cas contraire, Amazon DocumentDB n'envoie qu'un document partiel contenant les modifications.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `FullDocument` propriété d'un type de `AWS::Lambda::EventSourceMapping` `DocumentDBEventSourceConfig` données.

`KmsKeyArn`

Le nom de ressource Amazon (ARN) de la clé permettant de chiffrer les informations relatives à cet événement.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `KmsKeyArn` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`MaximumBatchingWindowInSeconds`

Intervalle de temps maximal (en secondes) pour collecter des enregistrements avant d'invoquer la fonction.

Type : entier

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `MaximumBatchingWindowInSeconds` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`SecretsManagerKmsKeyId`

L'identifiant de clé AWS Key Management Service (AWS KMS) d'une clé gérée par le client par AWS Secrets Manager. Requise lorsque vous utilisez une clé gérée par le client à partir de Secrets Manager avec un rôle d'exécution Lambda qui n'inclut pas l'autorisation `kms:Decrypt`.

La valeur de cette propriété est un UUID. Par exemple: `1abc23d4-567f-8ab9-cde0-1fab234c5d67`.

Type : chaîne

Obligatoire : Conditionnelle

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`SourceAccessConfigurations`

Un tableau du protocole d'authentification ou de l'hôte virtuel. Spécifiez-le à l'aide du type de SourceAccessConfigurationsdonnées.

Pour le type de source d'événement `DocumentDB`, le seul type de configuration valide est `BASIC_AUTH`.

- `BASIC_AUTH` – Le secret Secrets Manager qui stocke vos informations d'identification d'agent. Pour ce type, les informations d'identification doivent être au format suivant : `{"username": "your-username", "password": "your-password"}`. Un seul objet de type `BASIC_AUTH` est autorisé.

Type : liste

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `SourceAccessConfigurations` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`StartingPosition`

Position de début de la lecture dans le flux.

- `AT_TIMESTAMP` : spécifier l'heure à partir de laquelle la lecture des enregistrements doit commencer.
- `LATEST` : lire uniquement les nouveaux enregistrements.
- `TRIM_HORIZON` : traiter tous les enregistrements disponibles.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `StartingPosition` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`StartingPositionTimestamp`

L'heure à partir de laquelle commencer la lecture, en secondes au format horaire Unix. Définissez `StartingPositionTimestamp` lorsque `StartingPosition` est défini sur `AT_TIMESTAMP`.

Type : double

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `StartingPositionTimestamp` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

Exemples

Source d'événement Amazon DocumentDB

```
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

L'objet décrivant un type de source d'événement `DynamoDB`. Pour plus d'informations, consultez la section Utilisation AWS Lambda avec Amazon DynamoDB dans AWS Lambda le manuel du développeur.

AWS SAM génère une AWS::Lambda::EventSourceMappingressource lorsque ce type d'événement est défini.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

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
```

Propriétés

`BatchSize`

Nombre maximum d'éléments à récupérer dans un seul lot.

Type : entier

Obligatoire : non

Par défaut : 100

CloudFormation compatibilité : cette propriété est transmise directement à la `BatchSize` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

Minimum : `1`

Maximum : `1000`

`BisectBatchOnFunctionError`

Si la fonction renvoie une erreur, fractionnez le lot en deux et recommencez.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `BisectBatchOnFunctionError` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`DestinationConfig`

Une file d'attente Amazon Simple Queue Service (Amazon SQS) ou une rubrique de destination Amazon Simple Notification Service (Amazon SNS) pour les enregistrements ignorés.

Type : DestinationConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `DestinationConfig` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`Enabled`

Désactive le mappage de source d'événement pour suspendre l'interrogation et l'appel.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Enabled` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`FilterCriteria`

Objet qui définit les critères permettant de déterminer si Lambda doit traiter un événement. Pour de plus amples informations, veuillez consulter AWS Lambda le filtrage d'événements dans le AWS Lambda Manuel du développeur.

Type : FilterCriteria

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `FilterCriteria` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`FunctionResponseTypes`

Une liste des type de réponse actuellement appliquées au mappage de la source d'événement. Pour plus d'informations, veuillez consulter la rubrique Signalement des échecs d'éléments par lot dans le Guide du développeur AWS Lambda .

Valeurs valides : `ReportBatchItemFailures`

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `FunctionResponseTypes` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`KmsKeyArn`

Le nom de ressource Amazon (ARN) de la clé permettant de chiffrer les informations relatives à cet événement.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `KmsKeyArn` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`MaximumBatchingWindowInSeconds`

Intervalle de temps maximal (en secondes) pour collecter des enregistrements avant d'invoquer la fonction.

Type : entier

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `MaximumBatchingWindowInSeconds` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`MaximumRecordAgeInSeconds`

L'âge maximal d'une demande que Lambda envoie à une fonction pour traitement.

Type : entier

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `MaximumRecordAgeInSeconds` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`MaximumRetryAttempts`

Nombre maximum de tentatives autorisées lorsque la fonction renvoie une erreur.

Type : entier

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `MaximumRetryAttempts` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`MetricsConfig`

Une configuration optionnelle pour obtenir des métriques améliorées pour les mappages de sources d'événements qui capturent chaque étape du traitement. Pour obtenir un exemple, consultez MetricsConfig événement.

Type : MetricsConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `MetricsConfig` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`ParallelizationFactor`

Le nombre de lots à traiter simultanément à partir de chaque partition.

Type : entier

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `ParallelizationFactor` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`StartingPosition`

Position de début de la lecture dans le flux.

- `AT_TIMESTAMP` : spécifier l'heure à partir de laquelle la lecture des enregistrements doit commencer.
- `LATEST` : lire uniquement les nouveaux enregistrements.
- `TRIM_HORIZON` : traiter tous les enregistrements disponibles.

Valeurs valides : `AT_TIMESTAMP` | `LATEST` | `TRIM_HORIZON`

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `StartingPosition` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`StartingPositionTimestamp`

L'heure à partir de laquelle commencer la lecture, en secondes au format horaire Unix. Définissez `StartingPositionTimestamp` lorsque `StartingPosition` est défini sur `AT_TIMESTAMP`.

Type : double

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `StartingPositionTimestamp` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`Stream`

L'ARN (Amazon Resource Name) du flux de DynamoDB.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `EventSourceArn` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`TumblingWindowInSeconds`

La durée d'une fenêtre de traitement en secondes. La plage valide est de 1 à 900 (15 minutes).

Pour plus d'informations, consultez Fenêtres bascules dans le Guide du développeur AWS Lambda .

Type : entier

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `TumblingWindowInSeconds` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

Exemples

MetricsConfig événement

Voici un exemple de ressource qui utilise la `MetricsConfig` propriété pour capturer chaque étape du traitement pour ses mappages de sources d'événements.

```
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
```

Source d'événement DynamoDB pour une table DynamoDB existante

Source d'événement DynamoDB pour une table DynamoDB qui existe déjà dans un compte. 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
```

Événement DynamoDB pour la table DynamoDB déclarée dans le modèle

Événement DynamoDB pour une table DynamoDB déclarée dans le même fichier de modèle.

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

L'objet décrivant un type de source d'`EventBridgeRule`événement, qui définit votre fonction sans serveur comme cible d'une EventBridge règle Amazon. Pour plus d'informations, consultez Qu'est-ce qu'Amazon EventBridge ? dans le guide de EventBridge l'utilisateur Amazon.

AWS SAM génère une AWS::Events::Ruleressource lorsque ce type d'événement est défini. AWS SAM crée également une `AWS::Lambda::Permission` ressource, qui est nécessaire pour `EventBridgeRule` pouvoir appeler Lambda.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```yaml
DeadLetterConfig: DeadLetterConfig
EventBusName: String
Input: String
InputPath: String
InputTransformer: InputTransformer
Pattern: EventPattern
RetryPolicy: RetryPolicy
RuleName: String
State: String
Target: Target
```

Propriétés

`DeadLetterConfig`

Configurez la file d'attente Amazon Simple Queue Service (Amazon SQS) dans EventBridge laquelle les événements sont envoyés après l'échec d'un appel cible. L'invocation peut échouer, par exemple, lors de l'envoi d'un événement à une fonction Lambda qui n'existe pas ou EventBridge lorsque les autorisations sont insuffisantes pour appeler la fonction Lambda. Pour plus d'informations, consultez la politique relative aux nouvelles tentatives relatives aux événements et l'utilisation des files d'attente contenant des lettres mortes dans le guide de l'utilisateur Amazon. EventBridge

> **Note**
>
> Le type de ressource AWS::Serverless::Function a un type de données similaire, `DeadLetterQueue`, qui gère les échecs qui se produisent après l'invocation réussie de la fonction Lambda cible. Des exemples de ces types d'échecs incluent la limitation Lambda, ou les erreurs renvoyées par la fonction cible Lambda. Pour en savoir plus sur la propriété de fonction `DeadLetterQueue`, consultez File d'attente de lettres mortes dans le Guide du développeur AWS Lambda .

Type : DeadLetterConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est similaire à celle du type de `AWS::Events::Rule Target` données. `DeadLetterConfig` La AWS SAM version de cette propriété inclut des sous-propriétés supplémentaires, au cas où vous souhaiteriez AWS SAM créer la file d'attente de lettres mortes pour vous.

`EventBusName`

Le bus d'événements à associer à cette règle. Si vous omettez cette propriété, AWS SAM utilise le bus d'événements par défaut.

Type : chaîne

Obligatoire : non

Par défaut : bus d'événement par défaut

CloudFormation compatibilité : cette propriété est transmise directement à la `EventBusName` propriété d'une `AWS::Events::Rule` ressource.

`Input`

Texte JSON valide transmis à la cible. Si vous utilisez cette propriété, aucun élément du texte de l'événement lui-même n'est transmis à la cible.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Input` propriété d'une `AWS::Events::Rule Target` ressource.

`InputPath`

Lorsque vous ne voulez pas transmettre l'événement correspondant complet, utilisez la propriété `InputPath` pour décrire quelles parties de l'événement transmettre à la cible.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `InputPath` propriété d'une `AWS::Events::Rule Target` ressource.

`InputTransformer`

Paramètres qui vous permettent de fournir une entrée personnalisée à une cible en fonction de certaines données d'événement. Vous pouvez extraire une ou plusieurs paires clé-valeur à partir de l'événement, puis utiliser ces données pour envoyer l'entrée personnalisée à la cible. Pour plus d'informations, consultez la section Transformation des EventBridge entrées Amazon dans le guide de EventBridge l'utilisateur Amazon.

Type : InputTransformer

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `InputTransformer` propriété d'un type de `AWS::Events::Rule Target` données.

`Pattern`

Décrit les événements qui sont acheminés vers la cible spécifiée. Pour plus d'informations, consultez les EventBridgeévénements Amazon et les modèles EventBridge d'événements dans le guide de EventBridge l'utilisateur Amazon.

Type : EventPattern

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `EventPattern` propriété d'une `AWS::Events::Rule` ressource.

`RetryPolicy`

Objet `RetryPolicy` qui inclut des informations sur les paramètres de politique de nouvelle tentative. Pour plus d'informations, consultez la politique relative aux nouvelles tentatives relatives aux événements et l'utilisation des files d'attente contenant des lettres mortes dans le guide de l'utilisateur Amazon. EventBridge

Type : RetryPolicy

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `RetryPolicy` propriété du type de `AWS::Events::Rule` `Target` données.

`RuleName`

Le nom de la règle .

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Name` propriété d'une `AWS::Events::Rule` ressource.

`State`

État de la règle.

Valeurs acceptées : `DISABLED` | `ENABLED` | `ENABLED_WITH_ALL_CLOUDTRAIL_MANAGEMENT_EVENTS`

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `State` propriété d'une `AWS::Events::Rule` ressource.

`Target`

La AWS ressource qui est EventBridge invoquée lorsqu'une règle est déclenchée. Vous pouvez utiliser cette propriété pour spécifier l'ID logique de la cible. Si cette propriété n'est pas spécifiée, AWS SAM génère l'ID logique de la cible.

Type : cible

Obligatoire : non

CloudFormation compatibilité : cette propriété est similaire à celle `Targets` d'une `AWS::Events::Rule` ressource. `Amazon EC2 RebootInstances API call`est un exemple de propriété cible. La version AWS SAM de cette propriété vous permet uniquement de spécifier l'ID logique d'une seule cible.

Exemples

EventBridgeRule

Voici un exemple de type de source d'événement `EventBridgeRule`.

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

L'objet utilisé pour spécifier la file d'attente Amazon Simple Queue Service (Amazon SQS) EventBridge où envoie les événements après l'échec d'un appel cible. L'appel peut échouer, par exemple, lors de l'envoi d'un événement à une fonction Lambda qui n'existe pas, ou en cas d'autorisations insuffisantes pour appeler la fonction Lambda. Pour plus d'informations, consultez la politique relative aux nouvelles tentatives relatives aux événements et l'utilisation des files d'attente contenant des lettres mortes dans le guide de l'utilisateur Amazon. EventBridge

> **Note**
>
> Le type de ressource AWS::Serverless::Function a un type de données similaire, `DeadLetterQueue`, qui gère les échecs qui se produisent après l'appel réussi de la fonction Lambda cible. Des exemples de ce type de défaillance incluent la limitation Lambda ou les erreurs renvoyées par la fonction cible Lambda. Pour en savoir plus sur la propriété de fonction `DeadLetterQueue`, consultez File d'attente de lettres mortes dans le Guide du développeur AWS Lambda .

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
Arn: String
QueueLogicalId: String
Type: String
```

Propriétés

`Arn`

L'Amazon Resource Name (ARN) de la file d'attente Amazon SQS spécifiée comme cible pour la file d'attente de lettres mortes.

> **Note**
>
> Spécifiez la propriété `Type` ou la propriété `Arn`, mais pas les deux.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Arn` propriété du type de `AWS::Events::Rule` `DeadLetterConfig` données.

`QueueLogicalId`

Le nom personnalisé de la file d'attente de lettres mortes qui la AWS SAM crée `Type` est spécifié.

> **Note**
>
> Si la propriété `Type` n'est pas définie, cette propriété est ignorée.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Type`

Type de la file d'attente. Lorsque cette propriété est définie, crée AWS SAM automatiquement une file d'attente de lettres mortes et y joint la politique basée sur les ressources nécessaire pour autoriser la ressource à envoyer des événements à la file d'attente.

> **Note**
>
> Spécifiez la propriété `Type` ou la propriété `Arn`, mais pas les deux.

Valeurs valides : `SQS`

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

DeadLetterConfig

DeadLetterConfig

YAML

```
DeadLetterConfig:
  Type: SQS
  QueueLogicalId: MyDLQ
```

Target

Configure la AWS ressource qui est EventBridge invoquée lorsqu'une règle est déclenchée.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
  Id: String
```

Propriétés

`Id`

L'ID logique de la cible.

La valeur de `Id` peut inclure des caractères alphanumériques, des points (`.`), des tirets (`-`) et des tirets du bas (`_`).

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `Id` propriété du type de `AWS::Events::Rule` `Target` données.

Exemples

Cible

YAML

```
EBRule:
```

```
  Type: EventBridgeRule
  Properties:
    Target:
      Id: MyTarget
```

HttpApi

L'objet décrivant une source d'événement avec type HttpApi.

Si une OpenApi définition du chemin et de la méthode spécifiés existe dans l'API, SAM ajoutera la section Intégration et sécurité Lambda (le cas échéant) pour vous.

Si aucune OpenApi définition pour le chemin et la méthode spécifiés n'existe dans l'API, SAM créera cette définition pour vous.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
ApiId: String
Auth: HttpApiFunctionAuth
Method: String
Path: String
PayloadFormatVersion: String
RouteSettings: RouteSettings
TimeoutInMillis: Integer
```

Propriétés

`ApiId`

Identificateur d'une ressource AWS::Serverless::HttpApi définie dans ce modèle.

Si elle n'est pas définie, une AWS::Serverless::HttpApi ressource par défaut est créée appelée à `ServerlessHttpApi` l'aide d'un OpenApi document généré contenant une union de tous les chemins et méthodes définis par les événements Api définis dans ce modèle qui ne spécifient pas de`ApiId`.

Cela ne peut pas référencer une ressource AWS::Serverless::HttpApi définie dans un autre modèle.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Auth`

Configuration d'authentification pour cette méthode Api+Chemin+Méthode spécifique.

Utile pour remplacer la configuration d'autorisation du paramètre de `DefaultAuthorizer` de l'API sur un chemin d'accès individuel, lorsqu'aucun `DefaultAuthorizer` n'est spécifié.

Type : HttpApiFunctionAuth

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Method`

Méthode HTTP pour laquelle cette fonction est appelée.

$Si non`Path`and`Method`sont spécifiés, SAM créera un chemin d'API par défaut qui achemine toute requête qui ne mappe pas à un autre point de terminaison vers cette fonction Lambda. Seul un de ces chemins par défaut peut exister par API.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Path`

Chemin d'accès d'URI pour lequel cette fonction est appelée. Doit commencer par `/`.

Si aucun `Path` et `Method` ne sont spécifiés, SAM créera un chemin d'API par défaut qui acheminera toute demande qui ne mappe pas à un autre point de terminaison vers cette fonction Lambda. Seul un de ces chemins par défaut peut exister par API.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`PayloadFormatVersion`

Spécifie le format de la charge utile envoyée à une intégration.

REMARQUE : PayloadFormatVersion nécessite que SAM modifie votre définition d'OpenAPI, de sorte que cela ne fonctionne qu'avec les éléments en ligne OpenApi définis dans la propriété. `DefinitionBody`

Type : chaîne

Obligatoire : non

Par défaut : 2.0

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`RouteSettings`

Les paramètres d'acheminement par acheminement pour cette API HTTP. Pour plus d'informations sur les paramètres d'itinéraire, consultez AWS::ApiGatewayV2::Stage RouteSettingsle guide du développeur d'API Gateway.

Remarque : S' RouteSettings ils sont spécifiés à la fois dans la source de HttpApi ressource et dans la source d'événement, AWS SAM fusionnez-les avec les propriétés de la source d'événements prioritaires.

Type : RouteSettings

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `RouteSettings` propriété d'une `AWS::ApiGatewayV2::Stage` ressource.

`TimeoutInMillis`

Délai d'attente personnalisé compris entre 50 et 29 000 millisecondes.

REMARQUE : TimeoutInMillis nécessite que SAM modifie votre définition d'OpenAPI, de sorte que cela ne fonctionne qu'avec les éléments en ligne OpenApi définis dans la propriété. `DefinitionBody`

Type : entier

Obligatoire : non

Par défaut : 5000

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

HttpApi Événement par défaut

HttpApi Evénement utilisant le chemin par défaut. Tous les chemins d‘accès et méthodes non mappés sur cette API seront acheminés vers ce point de terminaison.

YAML

```
Events:
  HttpApiEvent:
    Type: HttpApi
```

HttpApi

HttpApi Evénement utilisant un chemin et une méthode spécifiques.

YAML

```
Events:
  HttpApiEvent:
    Type: HttpApi
    Properties:
      Path: /
      Method: GET
```

HttpApi Autorisation

HttpApi Événement utilisant un autorisateur.

YAML

```
Events:
  HttpApiEvent:
    Type: HttpApi
    Properties:
      Path: /authenticated
      Method: GET
      Auth:
        Authorizer: OpenIdAuth
        AuthorizationScopes:
          - scope1
          - scope2
```

HttpApiFunctionAuth

Configure l'autorisation au niveau de l'événement.

Configurez l'autorisation pour une formule Api+Chemin+Méthode spécifique.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
  AuthorizationScopes: List
  Authorizer: String
```

Propriétés

`AuthorizationScopes`

Les étendues d'autorisation à appliquer à cette API, ce chemin d'accès et cette méthode.

Les portées répertoriées ici remplaceront toutes les portées appliquées par le `DefaultAuthorizer` s'il en existe un.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Authorizer`

Le `Authorizer` pour une Fonction spécifique. Pour utiliser l'autorisation IAM, spécifiez `AWS_IAM` et spécifiez `true` pour `EnableIamAuthorizer` dans la section `Globals` de votre modèle.

Si vous avez spécifié un mécanisme d'autorisation global sur l'API et que vous souhaitez rendre publique une fonction spécifique, remplacez en définissant `Authorizer` sur `NONE`.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

Fonction-Auth (Autorisation fonction)

Spécification de l'autorisation au niveau de la fonction

YAML

```
Auth:
  Authorizer: OpenIdAuth
  AuthorizationScopes:
    - scope1
    - scope2
```

autorisation IAM

Spécifie l'autorisation IAM au niveau de l'événement. Pour utiliser l'autorisation `AWS_IAM` au niveau de l'événement, vous devez également spécifier `true` pour `EnableIamAuthorizer` dans la section `Globals` de votre modèle. Pour de plus amples informations, veuillez consulter Section « Globales » du modèle AWS SAM.

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

L'objet décrivant un type de source d'événement `IoTRule`.

Crée une AWS::IoT::TopicRuleressource pour déclarer une AWS IoT règle. Pour plus d'informations, consultez la documentation CloudFormation.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
  AwsIotSqlVersion: String
  Sql: String
```

Propriétés

`AwsIotSqlVersion`

Version du moteur de règles SQL à utiliser lors de l'évaluation de la règle.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `AwsIotSqlVersion` propriété d'une `AWS::IoT::TopicRule TopicRulePayload` ressource.

`Sql`

Instruction SQL utilisée pour interroger la rubrique. Pour plus d'informations, consultez la Référence SQL AWS IoT dans le Guide du développeur AWS IoT .

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `Sql` propriété d'une `AWS::IoT::TopicRule TopicRulePayload` ressource.

Exemples

Règle IOT

Exemples de règle IOT

YAML

```
IoTRule:
  Type: IoTRule
  Properties:
    Sql: SELECT * FROM 'topic/test'
```

Kinesis

L'objet décrivant un type de source d'événement `Kinesis`. Pour plus d'informations, consultez la section Utilisation AWS Lambda avec Amazon Kinesis dans le Guide du AWS Lambda développeur.

AWS SAM génère une AWS::Lambda::EventSourceMappingressource lorsque ce type d'événement est défini.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

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
```

Propriétés

`BatchSize`

Nombre maximum d'éléments à récupérer dans un seul lot.

Type : entier

Obligatoire : non

Par défaut : 100

CloudFormation compatibilité : cette propriété est transmise directement à la `BatchSize` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

Minimum : 1

Maximum : `10000`

`BisectBatchOnFunctionError`

Si la fonction renvoie une erreur, fractionnez le lot en deux et recommencez.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `BisectBatchOnFunctionError` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`DestinationConfig`

Une file d'attente Amazon Simple Queue Service (Amazon SQS) ou une rubrique de destination Amazon Simple Notification Service (Amazon SNS) pour les enregistrements ignorés.

Type : DestinationConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `DestinationConfig` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`Enabled`

Désactive le mappage de source d'événement pour suspendre l'interrogation et l'appel.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Enabled` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`FilterCriteria`

Objet qui définit les critères permettant de déterminer si Lambda doit traiter un événement. Pour de plus amples informations, veuillez consulter AWS Lambda le filtrage d'événements dans le AWS Lambda Manuel du développeur.

Type : FilterCriteria

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `FilterCriteria` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`FunctionResponseTypes`

Une liste des type de réponse actuellement appliquées au mappage de la source d'événement. Pour plus d'informations, veuillez consulter la rubrique Signalement des échecs d'éléments par lot dans le Guide du développeur AWS Lambda .

Valeurs valides : `ReportBatchItemFailures`

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `FunctionResponseTypes` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`KmsKeyArn`

Le nom de ressource Amazon (ARN) de la clé permettant de chiffrer les informations relatives à cet événement.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `KmsKeyArn` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`MaximumBatchingWindowInSeconds`

Intervalle de temps maximal (en secondes) pour collecter des enregistrements avant d'invoquer la fonction.

Type : entier

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `MaximumBatchingWindowInSeconds` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`MaximumRecordAgeInSeconds`

L'âge maximal d'une demande que Lambda envoie à une fonction pour traitement.

Type : entier

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `MaximumRecordAgeInSeconds` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`MaximumRetryAttempts`

Nombre maximum de tentatives autorisées lorsque la fonction renvoie une erreur.

Type : entier

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `MaximumRetryAttempts` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`MetricsConfig`

Une configuration optionnelle pour obtenir des métriques améliorées pour les mappages de sources d'événements qui capturent chaque étape du traitement. Pour obtenir un exemple, consultez MetricsConfig événement.

Type : MetricsConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `MetricsConfig` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`ParallelizationFactor`

Le nombre de lots à traiter simultanément à partir de chaque partition.

Type : entier

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `ParallelizationFactor` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`StartingPosition`

Position de début de la lecture dans le flux.

- `AT_TIMESTAMP` : spécifier l'heure à partir de laquelle la lecture des enregistrements doit commencer.
- `LATEST` : lire uniquement les nouveaux enregistrements.
- `TRIM_HORIZON` : traiter tous les enregistrements disponibles.

Valeurs valides : `AT_TIMESTAMP` | `LATEST` | `TRIM_HORIZON`

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `StartingPosition` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`StartingPositionTimestamp`

L'heure à partir de laquelle commencer la lecture, en secondes au format horaire Unix. Définissez `StartingPositionTimestamp` lorsque `StartingPosition` est défini sur `AT_TIMESTAMP`.

Type : double

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `StartingPositionTimestamp` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`Stream`

L'Amazon Resource Name (ARN) du flux de données ou d'un consommateur de flux.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `EventSourceArn` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`TumblingWindowInSeconds`

La durée d'une fenêtre de traitement en secondes. La plage valide est de 1 à 900 (15 minutes).

Pour plus d'informations, consultez Fenêtres bascules dans le Guide du développeur AWS Lambda .

Type : entier

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `TumblingWindowInSeconds` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

Exemples

MetricsConfig événement

Voici un exemple de ressource qui utilise la `MetricsConfig` propriété pour capturer chaque étape du traitement pour ses mappages de sources d'événements.

```
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
```

Source de l'événement Kinesis

Voici un exemple de source d'événement Kinesis.

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

L'objet décrivant un type de source d'événement `MQ`. Pour plus d'informations, consultez Utilisation AWS Lambda avec Amazon MQ dans le Guide du développeur AWS Lambda .

AWS Serverless Application Model (AWS SAM) génère une AWS::Lambda::EventSourceMappingressource lorsque ce type d'événement est défini.

> **Note**
>
> Pour avoir une file d'attente Amazon MQ dans un cloud privé virtuel (VPC) qui se connecte à une fonction Lambda dans un réseau public, le rôle d'exécution de votre fonction doit inclure les autorisations suivantes :
>
> - `ec2:CreateNetworkInterface`
> - `ec2:DeleteNetworkInterface`
> - `ec2:DescribeNetworkInterfaces`
> - `ec2:DescribeSecurityGroups`
> - `ec2:DescribeSubnets`
> - `ec2:DescribeVpcs`
>
> Pour plus d'informations, consultez Autorisations de rôle d'exécution dans le Guide du développeur AWS Lambda .

Syntaxe

Pour déclarer cette entité dans votre AWS SAM modèle, utilisez la syntaxe suivante.

YAML

```
BatchSize: Integer
Broker: String
DynamicPolicyName: Boolean
Enabled: Boolean
```

```
  FilterCriteria: FilterCriteria
  KmsKeyArn: String
  MaximumBatchingWindowInSeconds: Integer
  Queues: List
  SecretsManagerKmsKeyId: String
  SourceAccessConfigurations: List
```

Propriétés

`BatchSize`

Nombre maximum d'éléments à récupérer dans un seul lot.

Type : entier

Obligatoire : non

Par défaut : 100

CloudFormation compatibilité : cette propriété est transmise directement à la `BatchSize` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

Minimum : `1`

Maximum : `10000`

`Broker`

Amazon Resource Name (ARN) de l'agent Amazon MQ.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `EventSourceArn` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`DynamicPolicyName`

Par défaut, le nom de la politique Gestion des identités et des accès AWS (IAM) est `SamAutoGeneratedAMQPolicy` destiné à la rétrocompatibilité. Spécifiez `true` pour utiliser un nom généré automatiquement pour votre politique IAM. Ce nom inclura l'ID logique de la source d'événements Amazon MQ.

> **Note**
>
> Lorsque vous utilisez plus d'une source d'événements Amazon MQ, spécifiez la valeur `true` pour éviter la duplication des noms de politique IAM.

Type : valeur booléenne

Obligatoire : non

Par défaut : `false`

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Enabled`

Si la valeur est `true`, le mappage de source d'événement est actif. Pour suspendre l'interrogation et l'appel, définissez ce paramètre sur `false`.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Enabled` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`FilterCriteria`

Objet qui définit les critères permettant de déterminer si Lambda doit traiter un événement. Pour de plus amples informations, veuillez consulter AWS Lambda le filtrage d'événements dans le AWS Lambda Manuel du développeur.

Type : FilterCriteria

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `FilterCriteria` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`KmsKeyArn`

Le nom de ressource Amazon (ARN) de la clé permettant de chiffrer les informations relatives à cet événement.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `KmsKeyArn` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`MaximumBatchingWindowInSeconds`

Intervalle de temps maximal (en secondes) pour collecter des enregistrements avant d'invoquer la fonction.

Type : entier

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `MaximumBatchingWindowInSeconds` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`Queues`

Le nom de la file d'attente de destination de l'agent Amazon MQ à consommer.

Type : liste

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `Queues` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`SecretsManagerKmsKeyId`

L'identifiant de clé AWS Key Management Service (AWS KMS) d'une clé gérée par le client provenant de AWS Secrets Manager. Requise lorsque vous utilisez une clé gérée par le client à partir de Secrets Manager avec un rôle d'exécution Lambda qui n'inclut pas l'autorisation `kms:Decrypt`.

La valeur de cette propriété est un UUID. Par exemple: `1abc23d4-567f-8ab9-cde0-1fab234c5d67`.

Type : chaîne

Obligatoire : Conditionnelle

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`SourceAccessConfigurations`

Un tableau du protocole d'authentification ou de l'hôte virtuel. Spécifiez-le à l'aide du type de SourceAccessConfigurationsdonnées.

Pour le type de source d'événement `MQ`, les seuls types de configuration valides sont `BASIC_AUTH` et `VIRTUAL_HOST`.

- **`BASIC_AUTH`** – Le secret Secrets Manager qui stocke vos informations d'identification d'agent. Pour ce type, les informations d'identification doivent être au format suivant : `{"username": "your-username", "password": "your-password"}`. Un seul objet de type `BASIC_AUTH` est autorisé.
- **`VIRTUAL_HOST`** – Le nom de l'hôte virtuel dans votre agent RabbitMQ. Lambda utilisera cet hôte RabbitMQ comme source d'événement. Un seul objet de type `VIRTUAL_HOST` est autorisé.

Type : liste

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `SourceAccessConfigurations` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

Exemples

Source d'événement Amazon MQ

Voici un exemple de type de source d'événement `MQ` pour un agent Amazon MQ.

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

L'objet décrivant un type de source d'événement `MSK`. Pour plus d'informations, consultez la section Utilisation AWS Lambda avec Amazon MSK dans le manuel du AWS Lambda développeur.

AWS Serverless Application Model (AWS SAM) génère une AWS::Lambda::EventSourceMappingressource lorsque ce type d'événement est défini.

Pour utiliser Schema Registry, vous devez définir des autorisations de rôle IAM spécifiques pour votre fonction. Voir Configuration complète avec des rôles IAM pour un exemple de configuration requise.

Syntaxe

Pour déclarer cette entité dans votre AWS SAM modèle, utilisez la syntaxe suivante.

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
```

Propriétés

`BatchSize`

Nombre maximal d'enregistrements dans chaque lot que Lambda extrait de votre flux ou file d'attente et envoie à votre fonction. Lambda transmet tous les enregistrements du lot à la fonction en un seul appel, dans la limite de charge utile pour une invocation synchrone (6 Mo).

Par défaut : 100

Plage valide : valeur minimale de 1. Valeur maximale de 10 000.

Type : entier

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `BatchSize` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`BisectBatchOnFunctionError`

Si la fonction renvoie une erreur, fractionnez le lot en deux et recommencez.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `BisectBatchOnFunctionError` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`ConsumerGroupId`

Chaîne qui configure la façon dont les événements seront lus à partir des rubriques Kafka.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `AmazonManagedKafkaConfiguration` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`DestinationConfig`

Objet de configuration qui spécifie la destination d'un événement après son traitement par Lambda.

Utilisez cette propriété pour spécifier la destination des invocations ayant échoués à partir de la source d'événements Amazon MSK.

Type : DestinationConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `DestinationConfig` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`Enabled`

Désactive le mappage de source d'événement pour suspendre l'interrogation et l'appel.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Enabled` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`FilterCriteria`

Objet qui définit les critères permettant de déterminer si Lambda doit traiter un événement. Pour de plus amples informations, veuillez consulter AWS Lambda le filtrage d'événements dans le AWS Lambda Manuel du développeur.

Type : FilterCriteria

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `FilterCriteria` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`FunctionResponseTypes`

Une liste des type de réponse actuellement appliquées au mappage de la source d'événement. Pour plus d'informations, veuillez consulter la rubrique Signalement des échecs d'éléments par lot dans le Guide du développeur AWS Lambda .

Valeurs valides : `ReportBatchItemFailures`

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `FunctionResponseTypes` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`KmsKeyArn`

Le nom de ressource Amazon (ARN) de la clé permettant de chiffrer les informations relatives à cet événement.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `KmsKeyArn` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`MaximumBatchingWindowInSeconds`

Intervalle de temps maximal (en secondes) pour collecter des enregistrements avant d'invoquer la fonction.

Type : entier

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `MaximumBatchingWindowInSeconds` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`MaximumRecordAgeInSeconds`

L'âge maximal d'une demande que Lambda envoie à une fonction pour traitement.

Type : entier

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `MaximumRecordAgeInSeconds` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`MaximumRetryAttempts`

Nombre maximum de tentatives autorisées lorsque la fonction renvoie une erreur.

Type : entier

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `MaximumRetryAttempts` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`LoggingConfig`

Objet de configuration qui spécifie la configuration de journalisation pour le mappage des sources d'événements.

Type : LoggingConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `LoggingConfig` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`MetricsConfig`

Objet de configuration qui spécifie la configuration des métriques pour le mappage des sources d'événements.

Type : MetricsConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `MetricsConfig` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`ProvisionedPollerConfig`

Configuration pour augmenter le nombre de sondeurs utilisés pour calculer les mappages de sources d'événements. Cette configuration permet un minimum de 1 sondeur et un maximum de 2 000 sondeurs. Pour un exemple, reportez-vous àProvisionedPollerConfig exemple.

Type : ProvisionedPollerConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `ProvisionedPollerConfig` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`SchemaRegistryConfig`

Configuration pour l'utilisation d'un registre de schémas avec la source d'événements Kafka.

> **Note**
>
> Cette fonctionnalité doit `ProvisionedPollerConfig` être configurée.

Type : SchemaRegistryConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `AmazonManagedKafkaEventSourceConfig` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`SourceAccessConfigurations`

Tableau du protocole d'authentification, composants VPC ou hôte virtuel pour sécuriser et définir votre source d'événement.

Valeurs valides : `CLIENT_CERTIFICATE_TLS_AUTH`

Type : liste de propriétés SourceAccessConfiguration

Obligatoire : non

CloudFormation compatibilité : cette propriété fait partie de la AmazonManagedKafkaEventSourceConfigpropriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`StartingPosition`

Position de début de la lecture dans le flux.

- `AT_TIMESTAMP` : spécifier l'heure à partir de laquelle la lecture des enregistrements doit commencer.
- `LATEST` : lire uniquement les nouveaux enregistrements.

- `TRIM_HORIZON` : traiter tous les enregistrements disponibles.

Valeurs valides : `AT_TIMESTAMP` | `LATEST` | `TRIM_HORIZON`

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `StartingPosition` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`StartingPositionTimestamp`

L'heure à partir de laquelle commencer la lecture, en secondes au format horaire Unix. Définissez `StartingPositionTimestamp` lorsque `StartingPosition` est défini sur `AT_TIMESTAMP`.

Type : double

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `StartingPositionTimestamp` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`Stream`

L'Amazon Resource Name (ARN) du flux de données ou d'un consommateur de flux.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `EventSourceArn` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`Topics`

Nom de la rubrique Kafka.

Type : liste

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `Topics` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

Exemples

Configuration complète avec des rôles IAM

L'exemple suivant montre une configuration complète, y compris la configuration de rôle IAM requise pour utiliser 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 exemple

```
ProvisionedPollerConfig:
  MinimumPollers: 1
  MaximumPollers: 200
```

Exemple Amazon MSK pour un cluster existant

Voici un exemple de type de source d'événement `MSK` pour un cluster Amazon MSK qui existe déjà dans un Compte AWS.

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
```

Exemple Amazon MSK pour un cluster déclaré dans le même modèle

Voici un exemple de type de source d'événement `MSK` pour un cluster Amazon MSK déclaré dans le même fichier modèle.

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
```

Source d'événements MSK avec registre de schémas

Voici un exemple de type de source d'`MSK`événement configuré avec un registre de schémas.

```
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
```

Source d'événements MSK avec registre de schémas Confluent

Voici un exemple de type de source d'`MSK`événement configuré avec un registre de schémas Confluent.

```
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

L'objet décrivant un type de source d'événement `S3`.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
Bucket: String
Events: String | List
Filter: NotificationFilter
```

Propriétés

`Bucket`

Nom du compartiment S3. Ce compartiment doit exister dans le même modèle.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est similaire à celle `BucketName` d'une `AWS::S3::Bucket` ressource. Ce champ est obligatoire dans SAM. Ce champ accepte uniquement une référence au compartiment S3 créé dans ce modèle

`Events`

L'événement de compartiment Amazon S3 pour lequel appeler la fonction Lambda. Consultez Types d'événements pris en charge par Amazon S3 pour obtenir la liste des valeurs valides.

Type : chaîne | liste

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `Event` propriété du type de `AWS::S3::Bucket` `LambdaConfiguration` données.

`Filter`

Les règles de filtrage qui déterminent quels objets Amazon S3 appellent la fonction Lambda. Pour en savoir plus sur le filtrage par nom de clé Amazon S3, consultez Configuration des notifications d'événement Amazon S3 dans le Guide de l'utilisateur Amazon Simple Storage Service.

Type : NotificationFilter

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Filter` propriété du type de `AWS::S3::Bucket` `LambdaConfiguration` données.

Exemples

Événement S3

Exemple d'événement S3.

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

L'objet décrivant un type de source d'`Schedule`événement, qui définit votre fonction sans serveur comme cible d'une EventBridge règle Amazon qui se déclenche selon un calendrier. Pour plus d'informations, consultez Qu'est-ce qu'Amazon EventBridge ? dans le guide de EventBridge l'utilisateur Amazon.

AWS Serverless Application Model (AWS SAM) génère une AWS::Events::Ruleressource lorsque ce type d'événement est défini.

> **Note**
>
> EventBridge propose désormais une nouvelle fonctionnalité de planification, Amazon EventBridge Scheduler. Amazon EventBridge Scheduler est un planificateur sans serveur qui vous permet de créer, d'exécuter et de gérer des tâches à partir d'un service géré centralisé.

EventBridge Schedulerest hautement personnalisable et offre une évolutivité améliorée par rapport aux règles EventBridge planifiées, avec un ensemble plus large d'opérations d'API cibles et Services AWS.
Nous vous recommandons d'utiliser EventBridge Scheduler pour invoquer des cibles selon un calendrier. Pour définir ce type de source d'événement dans vos AWS SAM modèles, consultezScheduleV2.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
DeadLetterConfig: DeadLetterConfig
Description: String
Enabled: Boolean
Input: String
Name: String
RetryPolicy: RetryPolicy
Schedule: String
State: String
```

Propriétés

`DeadLetterConfig`

Configurez la file d'attente Amazon Simple Queue Service (Amazon SQS) dans EventBridge laquelle les événements sont envoyés après l'échec d'un appel cible. L'invocation peut échouer, par exemple, lors de l'envoi d'un événement à une fonction Lambda qui n'existe pas ou EventBridge lorsque les autorisations sont insuffisantes pour appeler la fonction Lambda.
Pour plus d'informations, consultez la politique relative aux nouvelles tentatives relatives aux événements et l'utilisation des files d'attente contenant des lettres mortes dans le guide de l'utilisateur Amazon. EventBridge

> **Note**
>
> Le type de ressource AWS::Serverless::Function a un type de données similaire, `DeadLetterQueue`, qui gère les échecs qui se produisent après l'invocation réussie de la

fonction Lambda cible. Des exemples de ces types d'échecs incluent la limitation Lambda, ou les erreurs renvoyées par la fonction cible Lambda. Pour en savoir plus sur la propriété de fonction `DeadLetterQueue`, consultez File d'attente de lettres mortes dans le Guide du développeur AWS Lambda .

Type : DeadLetterConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est similaire à celle du type de `AWS::Events::Rule Target` données. `DeadLetterConfig` La AWS SAM version de cette propriété inclut des sous-propriétés supplémentaires, au cas où vous souhaiteriez AWS SAM créer la file d'attente de lettres mortes pour vous.

`Description`

Description de la règle.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Description` propriété d'une `AWS::Events::Rule` ressource.

`Enabled`

Indique si la règle est activée.

Pour désactiver la règle, définissez cette propriété sur `false`.

> **Note**
>
> Spécifiez la propriété `Enabled` ou la propriété `State`, mais pas les deux.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est similaire à celle `State` d'une `AWS::Events::Rule` ressource. Si cette propriété est définie sur `true` then`ENABLED`, elle est AWS SAM transmise dans le cas contraire`DISABLED`.

`Input`

Texte JSON valide transmis à la cible. Si vous utilisez cette propriété, aucun élément du texte de l'événement lui-même n'est transmis à la cible.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Input` propriété d'une `AWS::Events::Rule Target` ressource.

`Name`

Le nom de la règle . Si vous ne spécifiez pas de nom, CloudFormation génère un identifiant physique unique et utilise cet identifiant comme nom de règle.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Name` propriété d'une `AWS::Events::Rule` ressource.

`RetryPolicy`

Objet `RetryPolicy` qui inclut des informations sur les paramètres de politique de nouvelle tentative. Pour plus d'informations, consultez la politique relative aux nouvelles tentatives relatives aux événements et l'utilisation des files d'attente contenant des lettres mortes dans le guide de l'utilisateur Amazon. EventBridge

Type : RetryPolicy

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `RetryPolicy` propriété du type de `AWS::Events::Rule Target` données.

`Schedule`

Expression de planification qui détermine quand et à quelle fréquence la règle s'exécute. Pour plus d'informations, consultez Expression de planification des règles.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `ScheduleExpression` propriété d'une `AWS::Events::Rule` ressource.

`State`

État de la règle.

Valeurs acceptées : `DISABLED | ENABLED`

> **Note**
>
> Spécifiez la propriété `Enabled` ou la propriété `State`, mais pas les deux.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `State` propriété d'une `AWS::Events::Rule` ressource.

Exemples

CloudWatch Planifier un événement

CloudWatch Exemple de planification d'un événement

YAML

```
CWSchedule:
  Type: Schedule
  Properties:
    Schedule: 'rate(1 minute)'
    Name: TestSchedule
    Description: test schedule
    Enabled: false
```

DeadLetterConfig

L'objet utilisé pour spécifier la file d'attente Amazon Simple Queue Service (Amazon SQS) EventBridge où envoie les événements après l'échec d'un appel cible. L'appel peut échouer, par exemple, lors de l'envoi d'un événement à une fonction Lambda qui n'existe pas, ou en cas d'autorisations insuffisantes pour appeler la fonction Lambda. Pour plus d'informations, consultez la politique relative aux nouvelles tentatives relatives aux événements et l'utilisation des files d'attente contenant des lettres mortes dans le guide de l'utilisateur Amazon. EventBridge

> **Note**
>
> Le type de ressource AWS::Serverless::Function a un type de données similaire, `DeadLetterQueue`, qui gère les échecs qui se produisent après l'appel réussi de la fonction Lambda cible. Des exemples de ce type de défaillance incluent la limitation Lambda ou les erreurs renvoyées par la fonction cible Lambda. Pour en savoir plus sur la propriété de fonction `DeadLetterQueue`, consultez File d'attente de lettres mortes dans le Guide du développeur AWS Lambda .

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```yaml
Arn: String
QueueLogicalId: String
Type: String
```

Propriétés

`Arn`

L'Amazon Resource Name (ARN) de la file d'attente Amazon SQS spécifiée comme cible pour la file d'attente de lettres mortes.

> **Note**
>
> Spécifiez la propriété `Type` ou la propriété `Arn`, mais pas les deux.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Arn` propriété du type de `AWS::Events::Rule` `DeadLetterConfig` données.

`QueueLogicalId`

Le nom personnalisé de la file d'attente de lettres mortes qui la AWS SAM crée `Type` est spécifié.

> **Note**
>
> Si la propriété `Type` n'est pas définie, cette propriété est ignorée.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Type`

Type de la file d'attente. Lorsque cette propriété est définie, crée AWS SAM automatiquement une file d'attente de lettres mortes et y joint la politique basée sur les ressources nécessaire pour autoriser la ressource à envoyer des événements à la file d'attente.

> **Note**
>
> Spécifiez la propriété `Type` ou la propriété `Arn`, mais pas les deux.

Valeurs valides : `SQS`

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

DeadLetterConfig

DeadLetterConfig

YAML

```
DeadLetterConfig:
  Type: SQS
  QueueLogicalId: MyDLQ
```

ScheduleV2

L'objet décrivant un type de source d'`ScheduleV2`événement, qui définit votre fonction sans serveur comme cible d'un événement Amazon EventBridge Scheduler qui se déclenche selon un calendrier. Pour plus d'informations, consultez Qu'est-ce qu'Amazon EventBridge Scheduler ? dans le guide de l'utilisateur EventBridge du planificateur.

AWS Serverless Application Model (AWS SAM) génère une AWS::Scheduler::Schedule ressource lorsque ce type d'événement est défini.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

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
```

Propriétés

`DeadLetterConfig`

Configurez la file d'attente Amazon Simple Queue Service (Amazon SQS) dans EventBridge laquelle les événements sont envoyés après l'échec d'un appel cible. L'invocation peut échouer, par exemple, lors de l'envoi d'un événement à une fonction Lambda qui n'existe pas ou EventBridge lorsque les autorisations sont insuffisantes pour appeler la fonction Lambda. Pour plus d'informations, consultez la section Configuration d'une file d'attente de lettres mortes pour le EventBridge planificateur dans le guide de l'utilisateur du EventBridge planificateur.

> **Note**
>
> Le type de ressource AWS::Serverless::Function a un type de données similaire, `DeadLetterQueue`, qui gère les échecs qui se produisent après l'invocation réussie de la fonction Lambda cible. Des exemples de ces types d'échecs incluent la limitation Lambda, ou les erreurs renvoyées par la fonction cible Lambda. Pour en savoir plus sur la propriété de fonction `DeadLetterQueue`, consultez File d'attente de lettres mortes dans le Guide du développeur AWS Lambda .

Type : DeadLetterConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est similaire à celle du type de `AWS::Scheduler::Schedule Target` données. `DeadLetterConfig` La AWS SAM version de cette propriété inclut des sous-propriétés supplémentaires, au cas où vous souhaiteriez AWS SAM créer la file d'attente de lettres mortes pour vous.

`Description`

Description de la planification.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Description` propriété d'une `AWS::Scheduler::Schedule` ressource.

`EndDate`

Date, au format UTC, avant laquelle la planification peut invoquer sa cible. En fonction de l'expression de récurrence de la planification, les appels peuvent s'arrêter avant, ou au moment de, la EndDate spécifiée.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `EndDate` propriété d'une `AWS::Scheduler::Schedule` ressource.

`FlexibleTimeWindow`

Permet de configurer une fenêtre dans laquelle une planification peut être appelée.

Type : FlexibleTimeWindow

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `FlexibleTimeWindow` propriété d'une `AWS::Scheduler::Schedule` ressource.

`GroupName`

Nom du groupe de planifications à associer à cette planification. S'il n'est pas défini, le groupe par défaut est utilisé.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `GroupName` propriété d'une `AWS::Scheduler::Schedule` ressource.

`Input`

Texte JSON valide transmis à la cible. Si vous utilisez cette propriété, aucun élément du texte de l'événement lui-même n'est transmis à la cible.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Input` propriété d'une `AWS::Scheduler::Schedule Target` ressource.

`KmsKeyArn`

ARN d'une clé KMS utilisée pour chiffrer les données client.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `KmsKeyArn` propriété d'une `AWS::Scheduler::Schedule` ressource.

`Name`

Nom du calendrier. Si vous ne spécifiez pas de nom, AWS SAM génère un nom au format *`Function-Logical-IDEvent-Source-Name`* et utilise cet ID pour le nom du planning.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Name` propriété d'une `AWS::Scheduler::Schedule` ressource.

`OmitName`

Par défaut, AWS SAM génère et utilise un nom de calendrier au format de*`<Function-logical-ID><event-source-name>`*. Définissez cette propriété `true` pour CloudFormation générer un identifiant physique unique et utilisez-le plutôt pour le nom du calendrier.

Type : valeur booléenne

Obligatoire : non

Par défaut : `false`

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`PermissionsBoundary`

ARN de la politique utilisée pour définir la limite d'autorisations du rôle.

> **Note**
>
> S'il `PermissionsBoundary` est défini, les mêmes limites AWS SAM seront appliquées au rôle IAM cible du calendrier du planificateur.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `PermissionsBoundary` propriété d'une `AWS::IAM::Role` ressource.

`RetryPolicy`

Objet RetryPolicy qui inclut des informations sur les paramètres de politique de nouvelle tentative.

Type : RetryPolicy

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `RetryPolicy` propriété du type de `AWS::Scheduler::Schedule Target` données.

`RoleArn`

L'ARN du rôle IAM que le EventBridge planificateur utilisera pour la cible lorsque le calendrier est invoqué.

Type : RoleArn

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `RoleArn` propriété du type de `AWS::Scheduler::Schedule Target` données.

`ScheduleExpression`

Expression de planification qui détermine quand et à quelle fréquence l'événement de planification du planificateur s'exécute.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `ScheduleExpression` propriété d'une `AWS::Scheduler::Schedule` ressource.

`ScheduleExpressionTimezone`

Fuseau horaire dans lequel l'expression de planification est évaluée.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `ScheduleExpressionTimezone` propriété d'une `AWS::Scheduler::Schedule` ressource.

`StartDate`

Date, au format UTC, après laquelle la planification peut commencer à appeler une cible. En fonction de l'expression de récurrence de la planification, les appels peuvent se dérouler après, ou au moment de, la StartDate spécifiée.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `StartDate` propriété d'une `AWS::Scheduler::Schedule` ressource.

`State`

État de la planification du planificateur.

Valeurs acceptées : `DISABLED | ENABLED`

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `State` propriété d'une `AWS::Scheduler::Schedule` ressource.

Exemples

Exemple de base de définition d'une ressource ScheduleV2

```
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**
>
> L'ID physique généré par ScheduleV2 n'inclut pas le nom de la pile.

SelfManagedKafka

L'objet décrivant un type de source d'événement `SelfManagedKafka`. Pour plus d'informations, consultez la section Utilisation AWS Lambda avec Apache Kafka autogéré dans le Guide du AWS Lambda développeur.

AWS Serverless Application Model (AWS SAM) génère une AWS::Lambda::EventSourceMappingressource lorsque ce type d'événement est défini.

Pour utiliser Schema Registry, vous devez définir des autorisations de rôle IAM spécifiques pour votre fonction. Voir Configuration complète avec des rôles IAM pour un exemple de configuration requise.

Syntaxe

Pour déclarer cette entité dans votre AWS SAM modèle, utilisez la syntaxe suivante.

YAML

```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
```

Propriétés

`BatchSize`

Nombre maximal d’enregistrements dans chaque lot que Lambda extrait de votre flux et envoie à votre fonction.

Type : entier

Obligatoire : non

Par défaut : 100

CloudFormation compatibilité : cette propriété est transmise directement à la `BatchSize` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

Minimum : `1`

Maximum : `10000`

`BisectBatchOnFunctionError`

Si la fonction renvoie une erreur, fractionnez le lot en deux et recommencez.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `BisectBatchOnFunctionError` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`ConsumerGroupId`

Chaîne qui configure la façon dont les événements seront lus à partir des rubriques Kafka.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `SelfManagedKafkaEventSourceConfig` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`DestinationConfig`

Objet de configuration qui spécifie la destination d'un événement après son traitement par Lambda.

Utilisez cette propriété pour spécifier la destination des invocations échouées de la source d'événements Kafka autogérée.

Type : DestinationConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `DestinationConfig` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`Enabled`

Désactive le mappage de source d'événement pour suspendre l'interrogation et l'appel.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Enabled` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`FilterCriteria`

Objet qui définit les critères permettant de déterminer si Lambda doit traiter un événement. Pour de plus amples informations, veuillez consulter AWS Lambda le filtrage d'événements dans le AWS Lambda Manuel du développeur.

Type : FilterCriteria

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `FilterCriteria` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`KafkaBootstrapServers`

La liste des serveurs d'amorçage pour vos agents Kafka. Inclure le port, par exemple `broker.example.com:`*`xxxx`*

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`FunctionResponseTypes`

Une liste des type de réponse actuellement appliquées au mappage de la source d'événement. Pour plus d'informations, veuillez consulter la rubrique Signalement des échecs d'éléments par lot dans le Guide du développeur AWS Lambda .

Valeurs valides : `ReportBatchItemFailures`

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `FunctionResponseTypes` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`KmsKeyArn`

Le nom de ressource Amazon (ARN) de la clé permettant de chiffrer les informations relatives à cet événement.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `KmsKeyArn` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`LoggingConfig`

La configuration de journalisation de votre source d'événements.

Type : LoggingConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `LoggingConfig` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`MaximumRecordAgeInSeconds`

L'âge maximal d'une demande que Lambda envoie à une fonction pour traitement.

Type : entier

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `MaximumRecordAgeInSeconds` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`MetricsConfig`

La configuration des métriques pour votre source d'événements.

Type : MetricsConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `MetricsConfig` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`MaximumRetryAttempts`

Nombre maximum de tentatives autorisées lorsque la fonction renvoie une erreur.

Type : entier

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `MaximumRetryAttempts` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`ProvisionedPollerConfig`

Configuration pour augmenter le nombre de sondeurs utilisés pour calculer les mappages de sources d'événements. Cette configuration permet un minimum de 1 sondeur et un maximum de 2 000 sondeurs. Pour un exemple, reportez-vous à ProvisionedPollerConfig exemple

Type : ProvisionedPollerConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `ProvisionedPollerConfig` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`SchemaRegistryConfig`

Configuration pour l'utilisation d'un registre de schémas avec la source d'événements Kafka autogérée.

> **Note**
>
> Cette fonctionnalité doit `ProvisionedPollerConfig` être configurée.

Type : SchemaRegistryConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `SelfManagedKafkaEventSourceConfig` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`SourceAccessConfigurations`

Tableau du protocole d'authentification, composants VPC ou hôte virtuel pour sécuriser et définir votre source d'événement.

Valeurs valides : `BASIC_AUTH | CLIENT_CERTIFICATE_TLS_AUTH | SASL_SCRAM_256_AUTH | SASL_SCRAM_512_AUTH | SERVER_ROOT_CA_CERTIFICATE`

Type : liste de propriétés SourceAccessConfiguration

Obligatoire : oui

CloudFormation compatibilité : cette propriété fait partie de la `SelfManagedKafkaEventSourceConfig` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`StartingPosition`

Position de début de la lecture dans le flux.

- `AT_TIMESTAMP` : spécifier l'heure à partir de laquelle la lecture des enregistrements doit commencer.
- `LATEST` : lire uniquement les nouveaux enregistrements.
- `TRIM_HORIZON` : traiter tous les enregistrements disponibles.

Valeurs valides : `AT_TIMESTAMP` | `LATEST` | `TRIM_HORIZON`

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `StartingPosition` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`StartingPositionTimestamp`

L'heure à partir de laquelle commencer la lecture, en secondes au format horaire Unix. Définissez `StartingPositionTimestamp` lorsque `StartingPosition` est défini sur `AT_TIMESTAMP`.

Type : double

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `StartingPositionTimestamp` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`Topics`

Nom de la rubrique Kafka.

Type : liste

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `Topics` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

Exemples

Configuration complète avec des rôles IAM

L'exemple suivant montre une configuration complète, y compris la configuration de rôle IAM requise pour utiliser 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 exemple

```
ProvisionedPollerConfig:
  MinimumPollers: 1
  MaximumPollers: 200
```

Source d’événements Kafka autogéré

Voici un exemple de type de source d'événement `SelfManagedKafka`.

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
```

Source d'événements Kafka autogérée avec registre de schémas AWS Glue

Voici un exemple de type de source d'`SelfManagedKafka`événement configuré avec AWS Glue Schema Registry.

```
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
```

Source d'événements Kafka autogérée avec registre de schémas Confluent

Voici un exemple de type de source d'`SelfManagedKafka`événement configuré avec Confluent Schema Registry.

```
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

L'objet décrivant un type de source d'événement `SNS`.

SAM génère une ressource AWS::SNS::Subscription lorsque ce type d'événement est défini.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
FilterPolicy: SnsFilterPolicy
FilterPolicyScope: String
RedrivePolicy: Json
Region: String
SqsSubscription: Boolean | SqsSubscriptionObject
Topic: String
```

Propriétés

`FilterPolicy`

Politique de filtre JSON attribuée à l'abonnement. Pour plus d'informations, consultez GetSubscriptionAttributesle manuel Amazon Simple Notification Service API Reference.

Type : SnsFilterPolicy

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `FilterPolicy` propriété d'une `AWS::SNS::Subscription` ressource.

`FilterPolicyScope`

Cet attribut vous permet de choisir l'étendue du filtrage en utilisant l'un des types de valeurs de chaîne suivants :

- `MessageAttributes` : le filtre est appliqué aux attributs du message.
- `MessageBody` : le filtre est appliqué sur le corps du message.

Type : chaîne

Obligatoire : non

Par défaut : `MessageAttributes`

CloudFormation compatibilité : cette propriété est transmise directement à la `FilterPolicyScope` propriété d'une `AWS::SNS::Subscription` ressource.

`RedrivePolicy`

Lorsque ce paramètre est spécifié, envoie des messages non livrables à la file d'attente de lettres mortes Amazon SQS spécifiée. Les messages qui ne peuvent pas être remis en raison d'erreurs client (par exemple, lorsque le point de terminaison abonné est inaccessible) ou d'erreurs de serveur (par exemple, lorsque le service qui alimente le point de terminaison abonné devient indisponible) sont conservés dans la file d'attente des lettres mortes pour une analyse ou un retraitement ultérieurs.

Pour de plus amples informations sur la stratégie de redirection et les files d'attente de lettres mortes, veuillez consulter Files d'attente de lettres mortes Amazon SQS dans le Manuel du développeur Amazon Simple Queue Service.

Type : Json

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `RedrivePolicy` propriété d'une `AWS::SNS::Subscription` ressource.

`Region`

Pour les abonnements entre régions, la région dans laquelle le topic réside.

Si aucune région n'est spécifiée, CloudFormation utilise la région de l'appelant par défaut.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Region` propriété d'une `AWS::SNS::Subscription` ressource.

`SqsSubscription`

Définissez cette propriété sur VRAI ou spécifiez `SqsSubscriptionObject` pour activer le traitement par lots des notifications de rubrique SNS dans une file d'attente SQS. Le fait de définir cette propriété sur `true` crée une nouvelle file d'attente SQS, alors que la spécification d'un `SqsSubscriptionObject` utilise une file d'attente SQS existante.

Type : Booléen | SqsSubscriptionObject

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Topic`

ARN de la rubrique à laquelle s'abonner.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `TopicArn` propriété d'une `AWS::SNS::Subscription` ressource.

Exemples

Exemple de source d'événement SNS

Exemple de source d'événement SNS

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

Spécifiez une option de file d'attente SQS existante pour l'événement SNS

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```yaml
BatchSize: String
Enabled: Boolean
QueueArn: String
QueuePolicyLogicalId: String
QueueUrl: String
```

Propriétés

`BatchSize`

Le nombre maximal d'éléments à récupérer dans un seul lot pour la file d'attente SQS.

Type : chaîne

Obligatoire : non

Par défaut : 10

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Enabled`

Désactive le mappage de source d'événement SQS pour suspendre l'interrogation et l'appel.

Type : valeur booléenne

Obligatoire : non

Valeur par défaut : VRAI

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`QueueArn`

Spécifiez un arn de file d'attente SQS existant.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`QueuePolicyLogicalId`

Donnez un nom LogicalID personnalisé à la AWS::SQS::QueuePolicyressource.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`QueueUrl`

Spécifiez l'URL de la file d'attente associée à la propriété `QueueArn`.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

Événement SQS existant pour SNS

Exemple d'ajout d'une file d'attente SQS existante pour l'abonnement à une rubrique SNS.

YAML

```
QueuePolicyLogicalId: CustomQueuePolicyLogicalId
QueueArn:
  Fn::GetAtt: MyCustomQueue.Arn
QueueUrl:
  Ref: MyCustomQueue
BatchSize: 5
```

SQS

L'objet décrivant un type de source d'événement `SQS`. Pour plus d'informations, consultez la section Utilisation AWS Lambda avec Amazon SQS dans le manuel du AWS Lambda développeur.

SAM génère une ressource AWS::Lambda::EventSourceMapping lorsque ce type d'événement est défini.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
BatchSize: Integer
Enabled: Boolean
FilterCriteria: FilterCriteria
FunctionResponseTypes: List
KmsKeyArn: String
MaximumBatchingWindowInSeconds: Integer
MetricsConfig: MetricsConfig
ProvisionedPollerConfig: ProvisionedPollerConfig
Queue: String
ScalingConfig: ScalingConfig
```

Propriétés

`BatchSize`

Nombre maximum d'éléments à récupérer dans un seul lot.

Type : entier

Obligatoire : non

Par défaut : 10

CloudFormation compatibilité : cette propriété est transmise directement à la `BatchSize` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

Minimum : `1`

Maximum : `10000`

`Enabled`

Désactive le mappage de source d'événement pour suspendre l'interrogation et l'appel.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Enabled` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`FilterCriteria`

Objet qui définit les critères permettant de déterminer si Lambda doit traiter un événement. Pour de plus amples informations, veuillez consulter AWS Lambda le filtrage d'événements dans le AWS Lambda Manuel du développeur.

Type : FilterCriteria

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `FilterCriteria` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`FunctionResponseTypes`

Une liste des type de réponse actuellement appliquées au mappage de la source d'événement. Pour plus d'informations, veuillez consulter la rubrique Signalement des échecs d'éléments par lot dans le Guide du développeur AWS Lambda .

Valeurs valides : `ReportBatchItemFailures`

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `FunctionResponseTypes` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`KmsKeyArn`

Le nom de ressource Amazon (ARN) de la clé permettant de chiffrer les informations relatives à cet événement.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `KmsKeyArn` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`MaximumBatchingWindowInSeconds`

L'intervalle de temps maximal (en secondes) pour collecter des enregistrements avant d'appeler la fonction.

Type : entier

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `MaximumBatchingWindowInSeconds` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`MetricsConfig`

Une configuration optionnelle permettant d'obtenir des métriques améliorées pour les mappages de sources d'événements qui capturent chaque étape du traitement. Pour obtenir un exemple, consultez MetricsConfig événement.

Type : MetricsConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `MetricsConfig` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`ProvisionedPollerConfig`

Configuration pour augmenter le nombre de sondeurs utilisés pour calculer les mappages de sources d'événements. Cette configuration permet un minimum de 2 sondeurs et un maximum de 2 000 sondeurs. Pour un exemple, reportez-vous àProvisionedPollerConfig exemple.

Type : ProvisionedPollerConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `ProvisionedPollerConfig` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`Queue`

L'ARN de la file d'attente.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `EventSourceArn` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

`ScalingConfig`

Mise à l'échelle de la configuration des observateurs SQS pour contrôler le taux d'appels et définir le nombre maximal d'appels simultanés.

Type : `ScalingConfig`

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `ScalingConfig` propriété d'une `AWS::Lambda::EventSourceMapping` ressource.

Exemples

MetricsConfig événement

Voici un exemple de ressource qui utilise la `MetricsConfig` propriété pour capturer chaque étape du traitement pour ses mappages de sources d'événements.

```
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
```

Événement SQS de base

```
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"]}'
```

Configurer des rapports partiels par lots pour votre file d'attente SQS

```
Events:
  SQSEvent:
    Type: SQS
    Properties:
      Enabled: true
      FunctionResponseTypes:
        - ReportBatchItemFailures
      Queue: !GetAtt MySqsQueue.Arn
      BatchSize: 10
```

Fonction Lambda avec un événement SQS pour lequel la mise à l'échelle est configurée

```
MyFunction:
  Type: AWS::Serverless::Function
  Properties:
    ...
    Events:
      MySQSEvent:
        Type: SQS
        Properties:
          ...
          ScalingConfig:
```

```
            MaximumConcurrency: 10
```

ProvisionedPollerConfig exemple

```
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

Package de déploiement d'une fonction Lambda.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
  Bucket: String
  Key: String
  Version: String
```

Propriétés

`Bucket`

Un compartiment Amazon S3 dans la même AWS région que votre fonction.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `S3Bucket` propriété du type de `AWS::Lambda::Function` `Code` données.

`Key`

Clé Amazon S3 du package de déploiement.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `S3Key` propriété du type de `AWS::Lambda::Function` `Code` données.

`Version`

Pour les objets versionnés, la version de l'objet de package de déploiement à utiliser.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `S3ObjectVersion` propriété du type de `AWS::Lambda::Function` `Code` données.

Exemples

FunctionCode

`CodeUri`: exemple de code de fonction

YAML

```
CodeUri:
  Bucket: sam-s3-demo-bucket-name
  Key: mykey-name
  Version: 121212
```

FunctionUrlConfig

Crée une URL de AWS Lambda fonction avec les paramètres de configuration spécifiés. Une URL de la fonction Lambda est un point de terminaison HTTPS que vous pouvez utiliser pour appeler votre fonction.

Par défaut, l'URL de la fonction que vous avez créée utilise la version `$LATEST` de votre fonction Lambda. Si vous spécifiez un `AutoPublishAlias` pour votre fonction Lambda, le point de terminaison se connecte à l'alias de fonction spécifié.

Pour plus d'informations, consultez la section Fonction Lambda URLs dans le Guide du AWS Lambda développeur.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
AuthType: String
Cors: Cors
InvokeMode: String
```

Propriétés

`AuthType`

Le type d'autorisation pour votre URL de fonction. Pour utiliser Gestion des identités et des accès AWS (IAM) pour autoriser les demandes, définissez sur. `AWS_IAM` Pour un accès ouvert, définissez la valeur sur `NONE`.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `AuthType` propriété d'une `AWS::Lambda::Url` ressource.

`Cors`

Les paramètres du partage de ressources cross-origin (CORS) pour l'URL de votre fonction.

Type : Cors

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Cors` propriété d'une `AWS::Lambda::Url` ressource.

`InvokeMode`

Mode selon lequel votre URL de la fonction sera appelée. Pour que votre fonction renvoie la réponse une fois l'appel terminé, définissez la valeur sur `BUFFERED`. Pour que votre fonction diffuse la réponse, définissez la valeur sur `RESPONSE_STREAM`. La valeur par défaut est `BUFFERED`.

Valeurs valides : `BUFFERED` ou `RESPONSE_STREAM`

Type : chaîne

Obligatoire : non

AWS CloudFormation compatibilité : cette propriété est transmise directement à la `InvokeMode`propriété d'une `AWS::Lambda::Url` ressource.

Exemples

URL de fonction

L'exemple suivant crée une fonction Lambda avec une URL de la fonction. L'URL de fonction utilise l'autorisation IAM.

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

Configure le fournisseur de capacité auquel les versions publiées de la fonction seront associées. Cela permet à la fonction de s'exécuter sur des EC2 instances appartenant au client et gérées par Lambda.

> **Note**
>
> Cette configuration détermine le type de calcul d'une fonction et doit être spécifiée lors du premier déploiement de la fonction. Il est impossible de l'ajouter ou de le supprimer une fois que la ressource fonctionnelle a été créée.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
Arn: String
ExecutionEnvironmentMemoryGiBPerVCpu: Float
PerExecutionEnvironmentMaxConcurrency: Integer
```

Propriétés

`Arn`

L'ARN du fournisseur de capacité à utiliser pour cette fonction.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est propre à SAM.

`ExecutionEnvironmentMemoryGiBPerVCpu`

Rapport entre la mémoire (en GiB) et le vCPU pour chaque environnement d'exécution.

> **Note**
>
> Le ratio de mémoire par processeur ne peut pas dépasser la mémoire totale de la fonction de 2048 Mo. Les memory-to-CPU ratios pris en charge sont de 2 Go, 4 Go ou 8 Go par processeur.

Type : flotteur

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `ExecutionEnvironmentMemoryGiBPerVCpu` propriété d'une `AWS::Lambda::Function` ressource.

`PerExecutionEnvironmentMaxConcurrency`

Nombre maximal d'exécutions simultanées par environnement d'exécution (sandbox).

Type : entier

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `PerExecutionEnvironmentMaxConcurrency` propriété d'une `AWS::Lambda::Function` ressource.

Exemples

Configuration du fournisseur de capacité

L'exemple suivant montre une configuration de fournisseur de capacité qui fait référence à une ressource de fournisseur de capacité.

```
CapacityProviderConfig:
  Arn: !GetAtt MyCapacityProvider.Arn
  ExecutionEnvironmentMemoryGiBPerVCpu: 4.0
```

```
  PerExecutionEnvironmentMaxConcurrency: 100
```

FunctionScalingConfig

Configure le comportement de dimensionnement pour les versions de fonctions Lambda, en contrôlant le nombre d'environnements d'exécution (sandboxes) pouvant être créés. Cette configuration s'applique à la fois à $LATEST.PUBLISHED et aux versions de fonctions numériques.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
MinExecutionEnvironments: Integer
MaxExecutionEnvironments: Integer
```

Propriétés

`MinExecutionEnvironments`

Le nombre minimal d'environnements d'exécution à gérer pour la version de la fonction.

Type : entier

Obligatoire : non

Par défaut : `3`

Minimum : `0`

CloudFormation compatibilité : cette propriété est transmise directement à la `MinExecutionEnvironments` propriété d'une `AWS::Lambda::Function` ressource.

`MaxExecutionEnvironments`

Nombre maximal d'environnements d'exécution pouvant être créés pour la version de la fonction.

Type : entier

Obligatoire : non

Par défaut : `3`

Minimum : `0`

CloudFormation compatibilité : cette propriété est transmise directement à la `MaxExecutionEnvironments` propriété d'une `AWS::Lambda::Function` ressource.

Exemples

Configuration du dimensionnement des fonctions

L'exemple suivant montre une configuration de dimensionnement des fonctions avec des environnements d'exécution minimale et maximale.

```
FunctionScalingConfig:
  MinExecutionEnvironments: 5
  MaxExecutionEnvironments: 100
```

AWS::Serverless::GraphQLApi

Utilisez le type de `AWS::Serverless::GraphQLApi` ressource AWS Serverless Application Model (AWS SAM) pour créer et configurer une AWS AppSync GraphQL API pour votre application sans serveur.

Pour en savoir plus AWS AppSync, voir Qu'est-ce que c'est AWS AppSync ? dans le Guide AWS AppSync du développeur.

Syntaxe

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
```

Propriétés

`ApiKeys`

Créez une clé unique qui peut être utilisée pour effectuer des opérations GraphQL nécessitant une clé d'API.

Type : ApiKeys

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Auth`

Configurez l'authentification pour votre API GraphQL.

Type : Authentification

Obligatoire : oui

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Cache`

Entrée d'une opération `CreateApiCache`.

Type : AWS::AppSync::ApiCache

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la AWS::AppSync::ApiCacheressource.

`DataSources`

Créez des sources de données auxquelles les fonctions AWS AppSync peuvent se connecter. AWS SAM prend en charge Amazon DynamoDB et les sources de données AWS Lambda .

Type : DataSource

Obligatoire : oui

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`DomainName`

Nom de domaine personnalisé pour votre API GraphQL.

Type : AWS::AppSync::DomainName

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la AWS::AppSync::DomainNameressource. AWS SAM génère automatiquement la AWS::AppSync::DomainNameApiAssociationressource.

`Functions`

Configurez les fonctions GraphQL APIs pour effectuer certaines opérations.

Type : Fonction

Obligatoire : oui

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Logging`

Configure la CloudWatch journalisation Amazon pour votre GraphQL API.

Si vous ne spécifiez pas cette propriété, AWS SAM générera `CloudWatchLogsRoleArn` et définira les valeurs suivantes :

- `ExcludeVerboseContent: true`
- `FieldLogLevel: ALL`

Pour désactiver la journalisation, spécifiez les informations suivantes :

```
Logging: false
```

Type : LogConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `LogConfig` propriété d'une `AWS::AppSync::GraphQLApi` ressource.

`LogicalId`

Le nom unique de votre API GraphQL.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `Name` propriété d'une `AWS::AppSync::GraphQLApi` ressource.

`Name`

Nom de votre API GraphQL. Spécifiez cette propriété pour remplacer la valeur `LogicalId`.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Name` propriété d'une `AWS::AppSync::GraphQLApi` ressource.

`Resolvers`

Configurez des résolveurs pour les champs de votre API GraphQL. AWS SAM prend en charge les résolveurs de pipelines JavaScript.

Type : Résolveur

Obligatoire : oui

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`SchemaInline`

Représentation texte d'un schéma GraphQL au format SDL.

Type : chaîne

Obligatoire : selon les conditions. Vous devez spécifier `SchemaInline` ou `SchemaUri`.

CloudFormation compatibilité : cette propriété est transmise directement à la `Definition` propriété d'une `AWS::AppSync::GraphQLSchema` ressource.

`SchemaUri`

L'URI du compartiment Amazon Simple Storage Service (Amazon S3) du schéma ou le chemin d'accès à un dossier local.

Si vous spécifiez un chemin d'accès à un dossier local, le fichier CloudFormation doit d'abord être chargé sur Amazon S3 avant le déploiement. Vous pouvez utiliser la CLI AWS SAM pour faciliter ce processus. Pour de plus amples informations, veuillez consulter Comment AWS SAM télécharge les fichiers locaux lors du déploiement.

Type : chaîne

Obligatoire : selon les conditions. Vous devez spécifier `SchemaInline` ou `SchemaUri`.

CloudFormation compatibilité : cette propriété est transmise directement à la `DefinitionS3Location` propriété d'une `AWS::AppSync::GraphQLSchema` ressource.

`Tags`

Balises (paires clé-valeur) pour cette API GraphQL. Utilisez les balises pour identifier et classer les ressources.

Type : liste des éléments Tag

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Tag` propriété d'une `AWS::AppSync::GraphQLApi` ressource.

`XrayEnabled`

Indiquez si vous souhaitez utiliser le traçage aux rayons X AWS pour cette ressource.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `XrayEnabled` propriété d'une `AWS::AppSync::GraphQLApi` ressource.

Valeurs renvoyées

Pour obtenir la liste des valeurs renvoyées, reportez-vous AWS::Serverless::GraphQLApiau Guide de l'CloudFormation utilisateur.

Exemples

GraphQL API avec source de données DynamoDB

Dans cet exemple, nous créons une API GraphQL qui utilise une table DynamoDB comme source de données.

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!
}
```

template.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;
}
```

Voici notre code de résolution :

getPost.js

```
export function request(ctx) {
  return {};
}

export function response(ctx) {
  return ctx.prev.result;
}
```

API GraphQL avec une fonction Lambda comme source de données

Dans cet exemple, nous créons une API GraphQL qui utilise une fonction Lambda comme source de données.

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
}
```

Voici nos fonctions :

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

Créez une clé unique qui peut être utilisée pour effectuer des opérations GraphQL nécessitant une clé d'API.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
LogicalId:
  ApiKeyId: String
  Description: String
  ExpiresOn: Double
```

Propriétés

`ApiKeyId`

Nom unique de votre clé d'API. Spécifiez pour remplacer la valeur `LogicalId`.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `ApiKeyId` propriété d'une `AWS::AppSync::ApiKey` ressource.

`Description`

Description de votre clé d'API.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Description` propriété d'une `AWS::AppSync::ApiKey` ressource.

`ExpiresOn`

Heure après laquelle la clé API expire. La date est représentée en secondes depuis la date epoch, arrondie à l'heure la plus proche.

Type : double

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Expires` propriété d'une `AWS::AppSync::ApiKey` ressource.

`LogicalId`

Nom unique de votre clé d'API.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `ApiKeyId` propriété d'une `AWS::AppSync::ApiKey` ressource.

Auth

Configurez l'autorisation pour votre API GraphQL.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
Additional:
- AuthProvider
LambdaAuthorizer: LambdaAuthorizerConfig
OpenIDConnect: OpenIDConnectConfig
Type: String
UserPool: UserPoolConfig
```

Propriétés

`Additional`

Liste de types d'authentification supplémentaires pour votre API GraphQL.

Type : Liste des AuthProvider

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`LambdaAuthorizer`

Spécifiez la configuration d'autorisation facultative pour votre mécanisme d'autorisation de fonctions Lambda. Vous pouvez configurer cette propriété facultative lorsque `AWS_LAMBDA` est spécifié pour `Type`.

Type : LambdaAuthorizerConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `LambdaAuthorizerConfig` propriété d'une `AWS::AppSync::GraphQLApi` ressource.

`OpenIDConnect`

Spécifiez la configuration d'autorisation facultative pour votre service conforme à OpenID Connect. Vous pouvez configurer cette propriété facultative lorsque `OPENID_CONNECT` est spécifié pour `Type`.

Type : Open IDConnect Config

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `OpenIDConnectConfig` propriété d'une `AWS::AppSync::GraphQLApi` ressource.

`Type`

Type d'autorisation par défaut entre les applications et votre AWS AppSync GraphQL API.

Pour obtenir la liste et la description des valeurs autorisées, consultez la section Autorisation et authentification dans le Guide du développeur AWS AppSync .

Lorsque vous spécifiez un autorisateur Lambda (`AWS_LAMBDA`), il AWS SAM crée une politique Gestion des identités et des accès AWS (IAM) pour octroyer des autorisations entre votre API GraphQL et la fonction Lambda.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `AuthenticationType` propriété d'une `AWS::AppSync::GraphQLApi` ressource.

`UserPool`

Spécifiez la configuration d'autorisation facultative pour l'utilisation des groupes d'utilisateurs Amazon Cognito. Vous pouvez configurer cette propriété facultative lorsque `AMAZON_COGNITO_USER_POOLS` est spécifié pour `Type`.

Type : UserPoolConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `UserPoolConfig` propriété d'une `AWS::AppSync::GraphQLApi` ressource.

Exemples

Configurer un type d'autorisation par défaut et un type d'autorisation supplémentaire

Dans cet exemple, nous commençons par configurer un mécanisme d'autorisation Lambda comme type d'autorisation par défaut pour notre API GraphQL.

```
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
```

Ensuite, nous configurons des types d'autorisation supplémentaires pour notre API GraphQL en ajoutant les éléments suivants à notre modèle AWS SAM :

```
        Additional:
        - Type: AWS_IAM
        - Type: API_KEY
        - Type: OPENID_CONNECT
          OpenIDConnect:
```

```
            AuthTTL: 10
            ClientId: myId
            IatTTL: 10
            Issuer: prod
```

Cela donne le AWS SAM modèle suivant :

```
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

Configuration d'autorisation facultative pour vos types d'autorisation supplémentaires de l'API GraphQL.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
LambdaAuthorizer: LambdaAuthorizerConfig
```

```
OpenIDConnect: OpenIDConnectConfig
Type: String
UserPool: UserPoolConfig
```

Propriétés

`LambdaAuthorizer`

Spécifiez la configuration d'autorisation facultative pour votre autorisateur de AWS Lambda fonction. Vous pouvez configurer cette propriété facultative lorsque `AWS_LAMBDA` est spécifié pour `Type`.

Type : LambdaAuthorizerConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `LambdaAuthorizerConfig` propriété d'un `AWS::AppSync::GraphQLApi` `AdditionalAuthenticationProvider` objet.

`OpenIDConnect`

Spécifiez la configuration d'autorisation facultative pour votre service conforme à OpenID Connect. Vous pouvez configurer cette propriété facultative lorsque `OPENID_CONNECT` est spécifié pour `Type`.

Type : Open IDConnect Config

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `OpenIDConnectConfig` propriété d'un `AWS::AppSync::GraphQLApi` `AdditionalAuthenticationProvider` objet.

`Type`

Type d'autorisation par défaut entre les applications et votre AWS AppSync GraphQL API.

Pour obtenir la liste et la description des valeurs autorisées, consultez la section Autorisation et authentification dans le Guide du développeur AWS AppSync .

Lorsque vous spécifiez un autorisateur Lambda (`AWS_LAMBDA`), il AWS SAM crée une politique Gestion des identités et des accès AWS (IAM) pour octroyer des autorisations entre votre API GraphQL et la fonction Lambda.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `AuthenticationType` propriété d'un `AWS::AppSync::GraphQLApi` `AdditionalAuthenticationProvider` objet.

`UserPool`

Spécifiez la configuration d'autorisation facultative pour l'utilisation des groupes d'utilisateurs Amazon Cognito. Vous pouvez configurer cette propriété facultative lorsque `AMAZON_COGNITO_USER_POOLS` est spécifié pour `Type`.

Type : `UserPoolConfig`

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `UserPoolConfig` propriété d'un `AWS::AppSync::GraphQLApi` `AdditionalAuthenticationProvider` objet.

DataSource

Configurez une source de données à laquelle votre résolveur d'API GraphQL peut se connecter. Vous pouvez utiliser des modèles AWS Serverless Application Model (AWS SAM) pour configurer les connexions aux sources de données suivantes :

- Amazon DynamoDB
- AWS Lambda

Pour en savoir plus sur les sources de données, consultez Joindre une source de données dans le Guide du développeur AWS AppSync .

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
DynamoDb: DynamoDb
Lambda: Lambda
```

Propriétés

`DynamoDb`

Configurez une table DynamoDB comme source de données pour votre résolveur d'API GraphQL.

Type : DynamoDb

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Lambda`

Configurez une fonction Lambda en tant que source de données pour votre résolveur d'API GraphQL.

Type : Lambda

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

DynamoDb

Configurez une table Amazon DynamoDB comme source de données pour votre résolveur d'API GraphQL.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
LogicalId:
```

```
  DeltaSync: DeltaSyncConfig
  Description: String
  Name: String
  Permissions: List
  Region: String
  ServiceRoleArn: String
  TableArn: String
  TableName: String
  UseCallerCredentials: Boolean
  Versioned: Boolean
```

Propriétés

`DeltaSync`

Décrit une configuration Delta Sync.

Type : DeltaSyncConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `DeltaSyncConfig` propriété d'un `AWS::AppSync::DataSource DynamoDBConfig` objet.

`Description`

Description de la source de données.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Description` propriété d'une `AWS::AppSync::DataSource` ressource.

`LogicalId`

Nom unique de la source de données.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `Name` propriété d'une `AWS::AppSync::DataSource` ressource.

`Name`

Nom de la source de données. Spécifiez cette propriété pour remplacer la valeur `LogicalId`.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Name` propriété d'une `AWS::AppSync::DataSource` ressource.

`Permissions`

Accordez des autorisations à votre source de données à l'aide de Connecteurs AWS SAM . Vous pouvez fournir une entrée des façons suivantes :

- `Read`— Autorisez votre résolveur à lire votre source de données.
- `Write`— Autorisez votre résolveur à écrire dans votre source de données.

AWS SAM utilise une `AWS::Serverless::Connector` ressource qui est transformée lors du déploiement pour fournir vos autorisations. Pour en savoir plus sur les ressources générées, consultez CloudFormation ressources générées lorsque vous spécifiez AWS::Serverless::Connector.

> **Note**
>
> Vous pouvez spécifier `Permissions` ou `ServiceRoleArn`, mais pas les deux. Si aucune des deux n'est spécifiée, AWS SAM générera les valeurs par défaut de `Read` et`Write`. Pour révoquer l'accès à votre source de données, supprimez l'objet DynamoDB de votre modèle. AWS SAM

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent. Elle est similaire à la propriété`Permissions` d'une ressource `AWS::Serverless::Connector`.

`Region`

Le Région AWS de votre table DynamoDB. Si vous ne le spécifiez pas, AWS SAM utilise`AWS::Region`.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `AwsRegion` propriété d'un `AWS::AppSync::DataSource DynamoDBConfig` objet.

`ServiceRoleArn`

L'ARN du rôle de service Gestion des identités et des accès AWS (IAM) pour la source de données. Le système assume ce rôle lors de l'accès à la source de données.

Vous pouvez spécifier `Permissions` ou `ServiceRoleArn`, mais pas les deux.

Type : chaîne

Nécessaire : Non Si elle n'est pas spécifiée, AWS SAM applique la valeur par défaut pour`Permissions`.

CloudFormation compatibilité : cette propriété est transmise directement à la `ServiceRoleArn` propriété d'une `AWS::AppSync::DataSource` ressource.

`TableArn`

L'ARN de la table DynamoDB.

Type : chaîne

Obligatoire : selon les conditions. Si vous ne spécifiez pas `ServiceRoleArn`, `TableArn` est obligatoire.

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`TableName`

Nom de la table.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `TableName` propriété d'un `AWS::AppSync::DataSource DynamoDBConfig` objet.

`UseCallerCredentials`

Définir sur `true` pour utiliser IAM avec cette source de données.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `UseCallerCredentials` propriété d'un `AWS::AppSync::DataSource DynamoDBConfig` objet.

`Versioned`

Définir sur `true` pour utiliser la détection et la résolution des conflits et synchroniser avec cette source de données.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Versioned` propriété d'un `AWS::AppSync::DataSource DynamoDBConfig` objet.

Lambda

Configurez une AWS Lambda fonction en tant que source de données pour votre résolveur GraphQL d'API.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
LogicalId:
  Description: String
```

```
  FunctionArn: String
  Name: String
  ServiceRoleArn: String
```

Propriétés

`Description`

Description de la source de données.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Description` propriété d'une `AWS::AppSync::DataSource` ressource.

`FunctionArn`

ARN de la fonction Lambda

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `LambdaFunctionArn` propriété d'un `AWS::AppSync::DataSource LambdaConfig` objet.

`LogicalId`

Nom unique de la source de données.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `Name` propriété d'une `AWS::AppSync::DataSource` ressource.

`Name`

Nom de la source de données. Spécifiez cette propriété pour remplacer la valeur `LogicalId`.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Name` propriété d'une `AWS::AppSync::DataSource` ressource.

`ServiceRoleArn`

L'ARN du rôle de service Gestion des identités et des accès AWS (IAM) pour la source de données. Le système assume ce rôle lors de l'accès à la source de données.

> **Note**
>
> Pour révoquer l'accès à votre source de données, supprimez l'objet Lambda de votre modèle AWS SAM .

Type : chaîne

Nécessaire : Non Si ce n'est pas spécifié, AWS SAM fournira `Write` les autorisations à l'aide deConnecteurs AWS SAM .

CloudFormation compatibilité : cette propriété est transmise directement à la `ServiceRoleArn` propriété d'une `AWS::AppSync::DataSource` ressource.

Fonction

Configurez les fonctions GraphQL APIs pour effectuer certaines opérations.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
LogicalId:
  CodeUri: String
  DataSource: String
  Description: String
  Id: String
  InlineCode: String
  MaxBatchSize: Integer
  Name: String
  Runtime: Runtime
```

```
  Sync: SyncConfig
```

Propriétés

`CodeUri`

L'URI Amazon Simple Storage Service (Amazon S3) du code de fonction ou le chemin d'accès au dossier local.

Si vous spécifiez un chemin d'accès à un dossier local, le fichier CloudFormation doit d'abord être chargé sur Amazon S3 avant le déploiement. Vous pouvez utiliser la CLI AWS SAM pour faciliter ce processus. Pour de plus amples informations, veuillez consulter Comment AWS SAM télécharge les fichiers locaux lors du déploiement.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `CodeS3Location` propriété d'une `AWS::AppSync::FunctionConfiguration` ressource.

`DataSource`

Le nom de source de données que cette fonction va joindre.

- Pour référencer une source de données dans la ressource `AWS::Serverless::GraphQLApi`, spécifiez son ID logique.
- Pour référencer une source de données en dehors de la ressource `AWS::Serverless::GraphQLApi`, fournissez son attribut `Name` à l'aide de la fonction intrinsèque `Fn::GetAtt`. Par exemple, `!GetAtt MyLambdaDataSource.Name`.
- Pour référencer une source de données provenant d'une autre pile, utilisez `Fn::ImportValue`.

Si une variation de `[NONE | None | none]` est spécifiée, une `None` valeur AWS SAM sera générée pour l'`AWS::AppSync::DataSourceType`objet.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `DataSourceName` propriété d'une `AWS::AppSync::FunctionConfiguration` ressource.

`Description`

La description de votre fonction.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Description` propriété d'une `AWS::AppSync::FunctionConfiguration` ressource.

`Id`

L'ID de fonction d'une fonction située en dehors de la ressource `AWS::Serverless::GraphQLApi`.

- Pour référencer une fonction dans le même AWS SAM modèle, utilisez la fonction `Fn::GetAtt` intrinsèque. Par exemple `Id: !GetAtt createPostItemFunc.FunctionId`.
- Pour référencer une fonction provenant d'une autre pile, utilisez `Fn::ImportValue`.

Lors de l'utilisation`Id`, toutes les autres propriétés ne sont pas autorisées. AWS SAM transmettra automatiquement l'ID de fonction de votre fonction référencée.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`InlineCode`

Le code de fonction contenant les fonctions de demande et de réponse.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Code` propriété d'une `AWS::AppSync::FunctionConfiguration` ressource.

`LogicalId`

Le nom unique de votre fonction.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `Name` propriété d'une `AWS::AppSync::FunctionConfiguration` ressource.

`MaxBatchSize`

Le nombre maximal d'entrées de requêtes de résolveur qui seront envoyées à une seule fonction AWS Lambda dans une opération `BatchInvoke`.

Type : entier

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la MaxBatchSizepropriété d'une `AWS::AppSync::FunctionConfiguration` ressource.

`Name`

Nom de la fonction. Spécifiez pour remplacer la valeur `LogicalId`.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Name` propriété d'une `AWS::AppSync::FunctionConfiguration` ressource.

`Runtime`

Décrit un environnement d'exécution utilisé par une AWS AppSync fonction ou un résolveur de AWS AppSync pipeline. Spécifie le nom et la version d'exécution à utiliser.

Type : temps d'exécution

Obligatoire : oui

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent. Elle est similaire à la propriété`Runtime` d'une ressource `AWS::AppSync::FunctionConfiguration`.

`Sync`

Décrit une configuration Sync pour une fonction.

Spécifie la stratégie de détection de conflits et de résolution à utiliser lorsque la fonction est appelée.

Type : SyncConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `SyncConfig` propriété d'une `AWS::AppSync::FunctionConfiguration` ressource.

Environnement d'exécution

Le temps d'exécution de votre résolveur ou de votre fonction de pipeline. Spécifie le nom et la version d'exécution à utiliser.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
Name: String
Version: String
```

Propriétés

`Name`

Le nom du temps d'exécution à utiliser. Actuellement, la seule valeur autorisée est `APPSYNC_JS`.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `Name` propriété d'un `AWS::AppSync::FunctionConfiguration AppSyncRuntime` objet.

`Version`

La version du temps d'exécution à utiliser. Actuellement, la seule version autorisée est `1.0.0`.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `RuntimeVersion` propriété d'un `AWS::AppSync::FunctionConfiguration AppSyncRuntime` objet.

Résolveur

Configurez des résolveurs pour les champs de votre GraphQL API. AWS Serverless Application Model (AWS SAM) prend en charge les résolveurs de JavaScript pipeline.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
OperationType:
  LogicalId:
    Caching: CachingConfig
    CodeUri: String
    FieldName: String
    InlineCode: String
    MaxBatchSize: Integer
    Pipeline: List
    Runtime: Runtime
    Sync: SyncConfig
```

Propriétés

`Caching`

La configuration de mise en cache pour un résolveur dont la mise en cache est activée.

Type : CachingConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `CachingConfig` propriété d'une `AWS::AppSync::Resolver` ressource.

`CodeUri`

L'URI ou le chemin d'accès à un dossier local d'Amazon Simple Storage Service (Amazon S3) du code de la fonction de résolution.

Si vous spécifiez un chemin d'accès à un dossier local, le fichier CloudFormation doit d'abord être chargé sur Amazon S3 avant le déploiement. Vous pouvez utiliser la CLI AWS SAM pour faciliter ce processus. Pour de plus amples informations, veuillez consulter Comment AWS SAM télécharge les fichiers locaux lors du déploiement.

Si aucun `CodeUri` des deux `InlineCode` n'est fourni, AWS SAM générera `InlineCode` qui redirigera la demande vers la première fonction de pipeline et recevra la réponse de la dernière fonction de pipeline.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `CodeS3Location` propriété d'une `AWS::AppSync::Resolver` ressource.

`FieldName`

Le nom de votre résolveur. Spécifiez cette propriété pour remplacer la valeur `LogicalId`.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `FieldName` propriété d'une `AWS::AppSync::Resolver` ressource.

`InlineCode`

Code de votre résolveur contenant les fonctions de demande et de réponse.

Si aucun `CodeUri` des deux `InlineCode` n'est fourni, AWS SAM générera `InlineCode` qui redirigera la demande vers la première fonction de pipeline et recevra la réponse de la dernière fonction de pipeline.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Code` propriété d'une `AWS::AppSync::Resolver` ressource.

`LogicalId`

Le nom unique de votre résolveur. Dans un schéma GraphQL, le nom de votre résolveur doit correspondre au nom du champ pour lequel il est utilisé. Utilisez le même nom de champ pour `LogicalId`.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`MaxBatchSize`

Le nombre maximal d'entrées de requêtes de résolveur qui seront envoyées à une seule fonction AWS Lambda dans une opération `BatchInvoke`.

Type : entier

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `MaxBatchSize` propriété d'une `AWS::AppSync::Resolver` ressource.

`OperationType`

Type d'opération GraphQL associé à votre résolveur. Par exemple, `Query`, `Mutation` ou `Subscription`. Vous pouvez imbriquer plusieurs résolveurs `LogicalId` au sein d'un même `OperationType`.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `TypeName` propriété d'une `AWS::AppSync::Resolver` ressource.

`Pipeline`

Fonctions liées avec le résolveur de pipeline. Spécifiez les fonctions par ID logique dans une liste.

Type : liste

Obligatoire : oui

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent. Elle est similaire à la propriété `PipelineConfig` d'une ressource `AWS::AppSync::Resolver`.

`Runtime`

Le temps d'exécution de votre résolveur ou de votre fonction de pipeline. Spécifie le nom et la version à utiliser.

Type : Temps d'exécution

Obligatoire : oui

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent. Elle est similaire à la propriété `Runtime` d'une propriété `AWS::AppSync::Resolver`.

`Sync`

Décrit une configuration Sync pour un résolveur.

Spécifie la stratégie de détection de conflits et de résolution à utiliser lorsque le résolveur est appelé.

Type : SyncConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `SyncConfig` propriété d'une `AWS::AppSync::Resolver` ressource.

Exemples

Utilisez le code de fonction de résolution AWS SAM généré et enregistrez les champs sous forme de variables

Voici le schéma GraphQL de notre exemple :

```
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
}
```

Voici un extrait de notre AWS SAM modèle :

```
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
```

Dans notre AWS SAM modèle, nous ne spécifions pas `CodeUri` ou`InlineCode`. Lors du déploiement, génère AWS SAM automatiquement le code en ligne suivant pour notre résolveur :

```
export function request(ctx) {
  return {};
}

export function response(ctx) {
  return ctx.prev.result;
}
```

Ce code de résolution par défaut redirige la demande vers la première fonction de pipeline et reçoit la réponse de la dernière fonction de pipeline.

Dans notre première fonction de pipeline, nous pouvons utiliser le champ `args` fourni pour analyser l'objet de la requête et créer nos variables. Nous pouvons ensuite utiliser ces variables dans notre fonction. Voici un exemple de notre fonction `preprocessPostItem` :

```
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;
}
```

Environnement d’exécution

Le temps d'exécution de votre résolveur ou de votre fonction de pipeline. Spécifie le nom et la version d'exécution à utiliser.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```yaml
Name: String
Version: String
```

Propriétés

`Name`

Le nom du temps d'exécution à utiliser. Actuellement, la seule valeur autorisée est `APPSYNC_JS`.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `Name` propriété d'un `AWS::AppSync::Resolver AppSyncRuntime` objet.

`Version`

La version du temps d'exécution à utiliser. Actuellement, la seule version autorisée est `1.0.0`.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `RuntimeVersion` propriété d'un `AWS::AppSync::Resolver AppSyncRuntime` objet.

AWS::Serverless::HttpApi

Crée une API HTTP Amazon API Gateway, qui vous permet de créer RESTful APIs avec une latence et des coûts inférieurs à ceux de REST APIs. Pour plus d'informations, consultez la section Working with HTTP APIs dans le guide du développeur d'API Gateway.

Nous vous recommandons d'utiliser CloudFormation des hooks ou des politiques IAM pour vérifier que les ressources API Gateway sont associées à des autorisateurs afin de contrôler l'accès à celles-ci.

Pour plus d'informations sur l'utilisation CloudFormation des hooks, consultez la section Enregistrement des hooks dans le guide de l'utilisateur de la CloudFormation CLI et dans le apigw-enforce-authorizer GitHub référentiel.

Pour plus d'informations sur l'utilisation de politiques IAM, veuillez consulter Exiger que les routes d'API disposent d'une autorisation dans le Guide du développeur API Gateway.

> **Note**
>
> Lorsque vous déployez vers AWS CloudFormation, vos AWS SAM ressources sont AWS SAM transformées en CloudFormation ressources. Pour de plus amples informations, veuillez consulter CloudFormation Ressources générées pour AWS SAM.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

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
```

Propriétés

`AccessLogSettings`

Les paramètres pour la journalisation des accès dans une étape.

Type : AccessLogSettings

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `AccessLogSettings` propriété d'une `AWS::ApiGatewayV2::Stage` ressource.

`Auth`

Configure l'autorisation pour le contrôle de l'accès à votre API HTTP API Gateway.

Pour plus d'informations, consultez la section Contrôle de l'accès au protocole HTTP APIs avec les autorisateurs JWT dans le guide du développeur d'API Gateway.

Type : HttpApiAuth

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`CorsConfiguration`

Gère le partage de ressources entre origines (CORS) pour tous vos API Gateway HTTP. APIs Spécifiez le domaine à autoriser en tant que chaîne ou spécifiez un objet `HttpApiCorsConfiguration`. Notez que CORS nécessite AWS SAM de modifier votre définition OpenAPI. CORS ne fonctionne donc que si `DefinitionBody` la propriété est spécifiée.

Pour plus d'informations, consultez la Configuration CORS pour une API HTTP dans le Guide du développeur API Gateway.

> **Note**
>
> Il `CorsConfiguration` est défini à la fois dans une définition OpenAPI et au niveau de la propriété, puis AWS SAM fusionne les deux sources de configuration avec les propriétés prioritaires. Si cette propriété est définie sur `true`, alors toutes les origines sont autorisées.

Type : Chaîne | HttpApiCorsConfiguration

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`DefaultRouteSettings`

Les paramètres d'acheminement par défaut pour cette API HTTP. Ces paramètres s'appliquent à tous les acheminements sauf s'ils sont remplacés par le paramètre `RouteSettings` pour certains acheminements.

Type : RouteSettings

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `RouteSettings` propriété d'une `AWS::ApiGatewayV2::Stage` ressource.

`DefinitionBody`

La définition OpenAPI qui décrit votre API HTTP. Si vous ne spécifiez pas a `DefinitionUri` ou a`DefinitionBody`, AWS SAM génère un `DefinitionBody` pour vous en fonction de la configuration de votre modèle.

Type : JSON

Obligatoire : non

CloudFormation compatibilité : cette propriété est similaire à celle Body d'une `AWS::ApiGatewayV2::Api` ressource. Si certaines propriétés sont fournies, vous AWS SAM pouvez y insérer du contenu ou le modifier `DefinitionBody` avant qu'il ne soit transmis à CloudFormation. Les propriétés incluent le type `Auth` et le type `EventSource` HttpApi de la `AWS::Serverless::Function` ressource correspondante.

`DefinitionUri`

L'URI Amazon Simple Storage Service (Amazon S3), le chemin d'accès au fichier local ou l'objet d'emplacement de la définition OpenAPI qui définit l'API HTTP. L'objet Amazon S3 auquel cette propriété fait référence doit être un fichier de définition OpenAPI valide. Si vous ne spécifiez pas a `DefinitionUri` ou n'`DefinitionBody`êtes pas spécifié, AWS SAM génère un `DefinitionBody` pour vous en fonction de la configuration de votre modèle.

Si vous fournissez un chemin d'accès au fichier local, le modèle doit passer par le flux de travail qui inclut le fichier `sam deploy` ou la commande `sam package` pour que la définition soit correctement transformée.

Les fonctions intrinsèques ne sont pas prises en charge dans les fichiers de OpenApi définition externes auxquels vous faites référence`DefinitionUri`. Pour importer une OpenApi définition dans le modèle, utilisez la `DefinitionBody` propriété avec la transformation Include.

Type : Chaîne | HttpApiDefinition

Obligatoire : non

CloudFormation compatibilité : cette propriété est similaire à celle `BodyS3Location` d'une `AWS::ApiGatewayV2::Api` ressource. Les propriétés imbriquées d'Amazon S3 sont nommées différemment.

`Description`

Description de la ressource d'API HTTP.

Lorsque vous spécifiez`Description`, la OpenApi définition de la ressource d'API HTTP AWS SAM sera modifiée en définissant le `description` champ. Les scénarios suivants entraînent une erreur :

- La `DefinitionBody` propriété est spécifiée avec le `description` champ défini dans la définition de l'API ouverte. Cela entraîne un conflit de `description` champ qui AWS SAM ne sera pas résolu.
- La `DefinitionUri` propriété est spécifiée : elle AWS SAM ne modifiera pas une définition d'API ouverte extraite d'Amazon S3.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`DisableExecuteApiEndpoint`

Spécifie si les clients peuvent appeler votre API à l'aide du point de terminaison `execute-api` par défaut `https://{api_id}.execute-api.{region}.amazonaws.com`. Par défaut, les clients peuvent appeler votre API avec le point de terminaison par défaut. Pour exiger que les

clients utilisent uniquement un nom de domaine personnalisé pour appeler votre API, désactivez le point de terminaison par défaut.

Pour utiliser cette propriété, vous devez spécifier la `DefinitionBody` propriété au lieu de la `DefinitionUri` propriété ou définir `x-amazon-apigateway-endpoint-configuration` avec `disableExecuteApiEndpoint` dans votre définition OpenAPI.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est similaire à celle `DisableExecuteApiEndpoint` d'une `AWS::ApiGatewayV2::Api` ressource. Il est transmis directement à la propriété `disableExecuteApiEndpoint` d'une extension `x-amazon-apigateway-endpoint-configuration`, qui est ajoutée à la propriété `Body` d'une ressource `AWS::ApiGatewayV2::Api`.

`Domain`

Configure un domaine personnalisé pour cette API HTTP API Gateway.

Type : HttpApiDomainConfiguration

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`FailOnWarnings`

Spécifie si la création d'API HTTP doit être annulée (`true`) ou non (`false`) lorsqu'un avertissement est rencontré. La valeur par défaut est `false`.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `FailOnWarnings` propriété d'une `AWS::ApiGatewayV2::Api` ressource.

`Name`

Nom de la ressource d'API HTTP.

Lorsque vous le spécifiez`Name`, AWS SAM il modifiera la définition OpenAPI de la ressource d'API HTTP en définissant le `title` champ. Les scénarios suivants entraînent une erreur :

- La `DefinitionBody` propriété est spécifiée avec le `title` champ défini dans la définition de l'API ouverte. Cela entraîne un conflit de `title` champ qui AWS SAM ne sera pas résolu.
- La `DefinitionUri` propriété est spécifiée : elle AWS SAM ne modifiera pas une définition d'API ouverte extraite d'Amazon S3.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`PropagateTags`

Indiquez s'il faut ou non transmettre les balises de la propriété `Tags` aux ressources AWS::Serverless::HttpApi que vous avez générées. Spécifiez `True` pour la propagation des balises dans vos ressources générées.

Type : valeur booléenne

Obligatoire : non

Par défaut : `False`

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`RouteSettings`

Les paramètres d'acheminement par acheminement pour cette API HTTP. Pour plus d'informations, consultez la section Travailler avec des routes pour le protocole HTTP APIs dans le guide du développeur d'API Gateway.

Type : RouteSettings

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `RouteSettings` propriété d'une `AWS::ApiGatewayV2::Stage` ressource.

`StageName`

Le nom de l'étape d'API. Si aucun nom n'est spécifié, AWS SAM utilise le `$default` stage d'API Gateway.

Type : chaîne

Obligatoire : non

Par défaut : $default

CloudFormation compatibilité : cette propriété est transmise directement à la `StageName` propriété d'une `AWS::ApiGatewayV2::Stage` ressource.

`StageVariables`

Mappage qui définit les variables de l'étape. Les noms de variables peuvent comporter des caractères alphanumériques et des caractères de soulignement. Les valeurs doivent correspondre à [A-Za-Z0-9-._~ :/ ? #&=,] +.

Type : Json

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `StageVariables` propriété d'une `AWS::ApiGatewayV2::Stage` ressource.

`Tags`

Un mappage (chaîne à chaîne) qui spécifie les balises à ajouter à cette étape API Gateway. Les clés peuvent comporter de 1 à 128 caractères Unicode de longueur et ne peuvent pas être précédée du préfixe `aws:`. Les caractères suivants sont acceptés : l'ensemble des lettres Unicode, les chiffres, les espaces, `_`, `.`, `/`, =, + et `-`. Les valeurs de balise doivent comporter de 1 à 256 caractères Unicode en longueur.

Type: carte (map)

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Remarques supplémentaires : La `Tags` propriété nécessite de AWS SAM modifier votre définition OpenAPI, de sorte que les balises ne sont ajoutées que si la `DefinitionBody` propriété

est spécifiée ; aucune balise n'est ajoutée si la propriété est spécifiée. `DefinitionUri` AWS SAM ajoute automatiquement un `httpapi:createdBy:SAM` tag. Les balises sont également ajoutées à la ressource `AWS::ApiGatewayV2::Stage` et à la ressource `AWS::ApiGatewayV2::DomainName` (si le `DomainName` est spécifié).

Valeurs renvoyées

Réf

Lorsque vous transmettez l'ID logique de cette ressource à la fonction intrinsèque `Ref`, `Ref` renvoie l'ID API de la ressource sous-jacente `AWS::ApiGatewayV2::Api`, par exemple, `a1bcdef2gh`.

Pour plus d'informations sur l'utilisation de la fonction `Ref`, consultez `Ref` dans le Guide de l'utilisateur AWS CloudFormation .

Exemples

Simple HttpApi

L'exemple suivant montre le minimum nécessaire pour configurer un point de terminaison d'API HTTP soutenu par une fonction Lambda. Cet exemple utilise l'API HTTP par défaut qui AWS SAM crée.

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 avec Auth

L'exemple suivant montre comment configurer l'autorisation sur les points de terminaison d'API HTTP.

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 avec définition OpenAPI

L'exemple suivant montre comment ajouter une définition OpenAPI au modèle.

Notez que toutes AWS SAM les intégrations Lambda manquantes sont renseignées pour les HttpApi événements faisant référence à cette API HTTP. AWS SAM ajoute également tous les chemins manquants auxquels les HttpApi événements font référence.

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 avec paramètres de configuration

L'exemple suivant montre comment ajouter des configurations d'API HTTP et des configurations d'étape au modèle.

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

Configurez l'autorisation pour contrôler l'accès à votre API HTTP Amazon API Gateway.

Pour plus d'informations sur la configuration de l'accès au protocole HTTP APIs, consultez la section Contrôle et gestion de l'accès à une API HTTP dans API Gateway dans le guide du développeur d'API Gateway.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
Authorizers: OAuth2Authorizer | LambdaAuthorizer
DefaultAuthorizer: String
EnableIamAuthorizer: Boolean
```

Propriétés

`Authorizers`

Le mécanisme d'autorisation utilisé pour contrôler l'accès à votre API API Gateway.

Type : OAuth2Autorisateur | LambdaAuthorizer

Obligatoire : non

Par défaut : aucun

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Remarques supplémentaires : AWS SAM ajoute les autorisateurs à la définition d'OpenAPI.

`DefaultAuthorizer`

Spécifiez l'autorisation par défaut à utiliser pour autoriser les appels d'API vers votre API API Gateway. Vous pouvez spécifier `AWS_IAM` en tant que mécanisme d'autorisation par défaut si `EnableIamAuthorizer` est défini sur `true`. Sinon, spécifiez un mécanisme d'autorisation que vous avez défini dans `Authorizers`.

Type : chaîne

Obligatoire : non

Par défaut : aucun

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`EnableIamAuthorizer`

Spécifiez si l'autorisation IAM doit être utilisée pour la route API.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

OAuth Autorisateur 2.0

OAuth Exemple d'autorisateur 2.0

YAML

```
Auth:
  Authorizers:
    OAuth2Authorizer:
      AuthorizationScopes:
        - scope1
        - scope2
      JwtConfiguration:
        issuer: "https://www.example.com/v1/connect/oauth2"
        audience:
          - MyApi
      IdentitySource: "$request.querystring.param"
  DefaultAuthorizer: OAuth2Authorizer
```

Mécanisme d'autorisation IAM

Exemple de mécanisme d'autorisation IAM

YAML

```
Auth:
  EnableIamAuthorizer: true
  DefaultAuthorizer: AWS_IAM
```

LambdaAuthorizer

Configurez un autorisateur Lambda pour contrôler l'accès à votre API HTTP Amazon API Gateway à l'aide d'une fonction. AWS Lambda

Pour plus d'informations et des exemples, consultez la section Working with AWS Lambda authorizers for HTTP APIs dans le guide du développeur d'API Gateway.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
  AuthorizerPayloadFormatVersion: String
  EnableFunctionDefaultPermissions: Boolean
  EnableSimpleResponses: Boolean
  FunctionArn: String
  FunctionInvokeRole: String
  Identity: LambdaAuthorizationIdentity
```

Propriétés

`AuthorizerPayloadFormatVersion`

Spécifie le format de la charge utile envoyée à un autorisateur Lambda API HTTP. Requis pour les autorisations Lambda API HTTP.

Cela est transmis à la section `authorizerPayloadFormatVersion`Section d'un `x-amazon-apigateway-authorizer` dans la section `securitySchemes` d'une définition OpenAPI.

Valeurs valides : `1.0` ou `2.0`

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`EnableFunctionDefaultPermissions`

Par défaut, la ressource d'API HTTP n'est pas autorisée à appeler le mécanisme d'autorisation Lambda. Spécifiez cette propriété comme `true` pour créer automatiquement des autorisations entre votre ressource d'API HTTP et votre mécanisme d'autorisation Lambda.

Type : valeur booléenne

Obligatoire : non

Valeur par défaut : `false`

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`EnableSimpleResponses`

Spécifie si un autorisateur Lambda renvoie une réponse dans un format simple. Par défaut, un autorisateur Lambda doit renvoyer une politique Gestion des identités et des accès AWS (IAM). Si cette option est activée, l'autorisateur Lambda peut renvoyer une valeur booléenne au lieu d'une politique IAM.

Cela est transmis à la section `enableSimpleResponses` d'un `x-amazon-apigateway-authorizer` dans la section `securitySchemes` d'une définition OpenAPI.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`FunctionArn`

L’Amazon Resource Name (ARN) de la fonction Lambda qui procure l’autorisation pour l’API.

Cela est transmis à la section `authorizerUri` d'un `x-amazon-apigateway-authorizer` dans la section `securitySchemes` d'une définition OpenAPI.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`FunctionInvokeRole`

L'ARN du rôle IAM qui possède les informations d'identification requises pour que la passerelle API Gateway appelle la fonction de mécanisme d'autorisation. Spécifiez ce paramètre si la stratégie basée sur les ressources de votre fonction n'accorde pas l'autorisation `lambda:InvokeFunction` API Gateway.

Cela est transmis à la section `authorizerCredentials` d'un `x-amazon-apigateway-authorizer` dans la section `securitySchemes` d'une définition OpenAPI.

Pour plus d'informations, consultez Création d'un mécanisme d'autorisation Lambda dans le Guide du développeur API Gateway.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Identity`

Spécifie une `IdentitySource` dans une demande de mécanisme d'autorisation entrante.

Cela est transmis à la section `identitySource` d'un `x-amazon-apigateway-authorizer` dans la section `securitySchemes` d'une définition OpenAPI.

Type : LambdaAuthorizationIdentity

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

LambdaAuthorizer

LambdaAuthorizer exemple

YAML

```
Auth:
  Authorizers:
    MyLambdaAuthorizer:
      AuthorizerPayloadFormatVersion: 2.0
      FunctionArn:
        Fn::GetAtt:
          - MyAuthFunction
          - Arn
      FunctionInvokeRole:
        Fn::GetAtt:
          - LambdaAuthInvokeRole
          - Arn
      Identity:
        Headers:
          - Authorization
```

LambdaAuthorizationIdentity

La propriété Use peut être utilisée pour spécifier un IdentitySource dans une demande entrante pour un autorisateur Lambda. Pour plus d'informations sur les sources d'identité, consultez Sources d'identité dans le Guide du développeur API Gateway.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

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

Propriétés

`Context`

Convertit les chaînes de contexte données en une liste d'expressions de mappage au format `$context.contextString`.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Headers`

Convertit les en-têtes en une liste d'expressions de mappage au format `$request.header.name`.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`QueryStrings`

Convertit les chaînes de demandes données en une liste d'expressions de mappage au format `$request.querystring.queryString`.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`ReauthorizeEvery`

Période time-to-live (TTL), en secondes, qui indique la durée pendant laquelle API Gateway met en cache les résultats de l'autorisateur. Si vous définissez une valeur supérieure à 0, API Gateway met en cache les réponses du mécanisme d'autorisation. La valeur maximale est 3 600 ou 1 heure.

Type : entier

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`StageVariables`

Convertit les variables d'étapes données en une liste d'expressions de mappage au format `$stageVariables.stageVariable`.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

LambdaRequestIdentity

Exemple d'identité de demande Lambda

YAML

```
Identity:
  QueryStrings:
    - auth
  Headers:
    - Authorization
  StageVariables:
    - VARIABLE
  Context:
    - authcontext
  ReauthorizeEvery: 100
```

OAuth2Authorizer

Définition d'un autorisateur OAuth 2.0, également connu sous le nom d'autorisateur JSON Web Token (JWT).

Pour plus d'informations, consultez la section Contrôle de l'accès au protocole HTTP APIs avec les autorisateurs JWT dans le guide du développeur d'API Gateway.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```yaml
AuthorizationScopes: List
IdentitySource: String
JwtConfiguration: Map
```

Propriétés

`AuthorizationScopes`

Liste des étendues d'autorisation pour cet autorisateur.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`IdentitySource`

Expression de source d'identité pour ce mécanisme d'autorisation.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`JwtConfiguration`

Configuration JWT pour ce mécanisme d'autorisation.

Cela est transmis à la section `jwtConfiguration` d'un `x-amazon-apigateway-authorizer` dans la section `securitySchemes` d'une définition OpenAPI.

> **Note**
>
> Les propriétés `audience` sont insensibles aux majuscules `issuer` et minuscules et peuvent être utilisées soit en minuscules comme dans OpenAPI, soit en majuscules et comme dans. `Issuer` `Audience` AWS::ApiGatewayV2::Authorizer

Type: carte (map)

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

OAuth Autorisateur 2.0

OAuth Exemple d'autorisateur 2.0

YAML

```
Auth:
  Authorizers:
    OAuth2Authorizer:
      AuthorizationScopes:
        - scope1
      JwtConfiguration:
        issuer: "https://www.example.com/v1/connect/oauth2"
        audience:
          - MyApi
      IdentitySource: "$request.querystring.param"
  DefaultAuthorizer: OAuth2Authorizer
```

HttpApiCorsConfiguration

Gérez le partage de ressources entre origines (CORS) pour votre protocole HTTP. APIs Spécifiez le domaine à autoriser en tant que chaîne ou spécifiez un dictionnaire avec une configuration Cors supplémentaire. REMARQUE : Cors a besoin de SAM pour modifier votre définition d'OpenAPI. Cela ne fonctionne donc qu'avec les éléments en OpenApi ligne définis dans la propriété. `DefinitionBody`

Pour plus d'informations sur CORS, consultez la Configuration CORS pour une API HTTP dans le Guide du développeur API Gateway.

Remarque : S'il HttpApiCorsConfiguration est défini à la fois dans OpenAPI et au niveau des propriétés, il les AWS SAM fusionne avec les propriétés prioritaires.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```yaml
AllowCredentials: Boolean
AllowHeaders: List
AllowMethods: List
AllowOrigins: List
ExposeHeaders: List
MaxAge: Integer
```

Propriétés

`AllowCredentials`

Spécifie si les informations d'identification sont incluses dans la demande CORS.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`AllowHeaders`

Représente une collection d'en-têtes autorisés.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`AllowMethods`

Représente une collection de méthodes HTTP autorisées.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`AllowOrigins`

Représente une collection d'origines autorisées.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`ExposeHeaders`

Représente une collection d'en-têtes exposés.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`MaxAge`

Nombre de secondes pendant lesquelles le navigateur doit mettre en cache les résultats de la demande de contrôle en amont.

Type : entier

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

HttpApiCorsConfiguration

Exemple de configuration Cors API HTTP.

YAML

```
CorsConfiguration:
  AllowOrigins:
    - "https://example.com"
  AllowHeaders:
    - x-apigateway-header
  AllowMethods:
    - GET
  MaxAge: 600
  AllowCredentials: true
```

HttpApiDefinition

Un document OpenAPI définissant l'API.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
  Bucket: String
  Key: String
  Version: String
```

Propriétés

`Bucket`

Nom du compartiment Amazon S3 dans lequel le fichier OpenAPI est stocké.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `Bucket` propriété du type de `AWS::ApiGatewayV2::Api` `BodyS3Location` données.

`Key`

La clé Amazon S3 du fichier OpenAPI.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `Key` propriété du type de `AWS::ApiGatewayV2::Api` `BodyS3Location` données.

`Version`

Pour les objets versionnés, la version du fichier OpenAPI.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Version` propriété du type de `AWS::ApiGatewayV2::Api` `BodyS3Location` données.

Exemples

Exemple de définition d'Uri

Exemple de définition d'API

YAML

```
DefinitionUri:
  Bucket: sam-s3-demo-bucket-name
  Key: mykey-name
  Version: 121212
```

HttpApiDomainConfiguration

Configure un domaine personnalisé pour une API.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
  BasePath: List
```

```
  CertificateArn: String
  DomainName: String
  EndpointConfiguration: String
  MutualTlsAuthentication: MutualTlsAuthentication
  OwnershipVerificationCertificateArn: String
  Route53: Route53Configuration
  SecurityPolicy: String
```

Propriétés

`BasePath`

Une liste des chemins de base à configurer avec le nom de domaine Amazon API Gateway.

Type : liste

Obligatoire : non

Par défaut : /

CloudFormation compatibilité : cette propriété est similaire à celle `ApiMappingKey` d'une `AWS::ApiGatewayV2::ApiMapping` ressource. AWS SAM crée plusieurs `AWS::ApiGatewayV2::ApiMapping` ressources, une par valeur spécifiée dans cette propriété.

`CertificateArn`

Le nom de ressource Amazon (ARN) d'un certificat AWS géré pour le point de terminaison de ce nom de domaine. AWS Certificate Manager est la seule source prise en charge.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `CertificateArn` propriété d'une `AWS::ApiGateway2::DomainName DomainNameConfiguration` ressource.

`DomainName`

Le nom de domaine personnalisé pour votre API dans Amazon API Gateway. Les majuscules ne sont pas prises en charge.

AWS SAM génère une `AWS::ApiGatewayV2::DomainName` ressource lorsque cette propriété est définie. Pour plus d'informations sur ce scénario, consultez DomainName la

propriété est spécifiée. Pour plus d'informations sur CloudFormation les ressources générées, consultezCloudFormation Ressources générées pour AWS SAM.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `DomainName` propriété d'une `AWS::ApiGateway2::DomainName` ressource.

`EndpointConfiguration`

Définit le type de point de terminaison API Gateway à mapper au domaine personnalisé. La valeur de cette propriété détermine la manière dont la `CertificateArn` propriété est mappée CloudFormation.

La seule valeur valide pour HTTP APIs est`REGIONAL`.

Type : chaîne

Obligatoire : non

Par défaut : `REGIONAL`

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`MutualTlsAuthentication`

La configuration d'authentification de protocole TLS (Transport Layer Security) mutuelle pour un nom de domaine personnalisé.

Type : MutualTlsAuthentication

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `MutualTlsAuthentication` propriété d'une `AWS::ApiGatewayV2::DomainName` ressource.

`OwnershipVerificationCertificateArn`

ARN du certificat public que vous avez généré dans pour valider la propriété de votre domaine personnalisé. Obligatoire uniquement lorsque vous configurez le protocole TLS mutuel et que vous précisez un ARN de certificat d'autorité de certification privé ou importé ACM pour le `CertificateArn`.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `OwnershipVerificationCertificateArn` propriété du type de `AWS::ApiGatewayV2::DomainName` `DomainNameConfiguration` données.

`Route53`

Définit une configuration Amazon Route 53.

Type : Route53Configuration

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`SecurityPolicy`

La version TLS de la stratégie de sécurité pour ce nom de domaine.

La seule valeur valide pour HTTP APIs est`TLS_1_2`.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `SecurityPolicy` propriété du type de `AWS::ApiGatewayV2::DomainName` `DomainNameConfiguration` données.

Exemples

DomainName

DomainName exemple

YAML

```
Domain:
  DomainName: www.example.com
```

```
  CertificateArn: arn-example
  EndpointConfiguration: REGIONAL
  Route53:
    HostedZoneId: Z1PA6795UKMFR9
  BasePath:
    - foo
    - bar
```

Route53Configuration

Configure les ensembles d'enregistrements Route53 pour une API.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
  DistributionDomainName: String
  EvaluateTargetHealth: Boolean
  HostedZoneId: String
  HostedZoneName: String
  IpV6: Boolean
  Region: String
  SetIdentifier: String
```

Propriétés

`DistributionDomainName`

Configure une distribution personnalisée du nom de domaine personnalisé de l'API.

Type : chaîne

Obligatoire : non

Par défaut : utilisez la distribution API Gateway.

CloudFormation compatibilité : cette propriété est transmise directement à la `DNSName` propriété d'une `AWS::Route53::RecordSetGroup AliasTarget` ressource.

Remarques supplémentaires : nom de domaine d'une CloudFrontdistribution.

`EvaluateTargetHealth`

Lorsque cela EvaluateTargetHealth est vrai, un enregistrement d'alias hérite de l'état de la AWS ressource référencée, comme un équilibreur de charge Elastic Load Balancing ou un autre enregistrement de la zone hébergée.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `EvaluateTargetHealth` propriété d'une `AWS::Route53::RecordSetGroup AliasTarget` ressource.

Remarques supplémentaires : vous ne pouvez pas EvaluateTargetHealth définir la valeur true lorsque l'alias cible est une CloudFront distribution.

`HostedZoneId`

ID de la zone hébergée dans laquelle vous souhaitez créer des enregistrements.

Spécifiez `HostedZoneName` ou `HostedZoneId`, mais pas les deux. Si plusieurs zones hébergées portent le même nom de domaine, vous devez spécifier la zone hébergée à l'aide de la valeur `HostedZoneId`.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `HostedZoneId` propriété d'une `AWS::Route53::RecordSetGroup RecordSet` ressource.

`HostedZoneName`

Nom de la zone hébergée dans laquelle vous souhaitez créer des enregistrements. Vous devez inclure un point de fin (par exemple, `www.example.com.`) dans le `HostedZoneName`.

Spécifiez `HostedZoneName` ou `HostedZoneId`, mais pas les deux. Si plusieurs zones hébergées portent le même nom de domaine, vous devez spécifier la zone hébergée à l'aide de la valeur `HostedZoneId`.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `HostedZoneName` propriété d'une `AWS::Route53::RecordSetGroup RecordSet` ressource.

`IpV6`

Lorsque cette propriété est définie, elle AWS SAM crée une `AWS::Route53::RecordSet` ressource et attribue à Type `AAAA` la valeur fournie HostedZone.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Region`

Ensembles d'enregistrements de ressources basés sur la latence uniquement : EC2 région Amazon dans laquelle vous avez créé la ressource à laquelle ce jeu d'enregistrements de ressources fait référence. La ressource est généralement une AWS ressource, telle qu'une EC2 instance ou un équilibreur de charge ELB, à laquelle il est fait référence par une adresse IP ou un nom de domaine DNS, selon le type d'enregistrement.

Lorsqu'Amazon Route 53 reçoit une requête DNS pour un nom et un type de domaine pour lesquels vous avez créé des ensembles d'enregistrements de ressources de latence, Route 53 sélectionne le jeu d'enregistrements de ressources de latence présentant la latence la plus faible entre l'utilisateur final et la EC2 région Amazon associée. Route 53 renvoie ensuite la valeur associée à l'ensemble d'enregistrements de ressource sélectionné.

Notez ce qui suit :

- Vous pouvez uniquement spécifier un `ResourceRecord` par ensemble d'enregistrements de ressource de latence.
- Vous ne pouvez créer qu'un seul ensemble d'enregistrements de ressources de latence pour chaque EC2 région Amazon.
- Vous n'êtes pas obligé de créer des ensembles d'enregistrements de ressources de latence pour toutes les EC2 régions Amazon. Route 53 choisit la région dotée de la meilleure latence parmi les régions pour lesquelles vous créez des ensembles d'enregistrements de ressource de latence.

- Vous ne pouvez pas créer des ensembles d'enregistrements de ressource sans latence ayant les mêmes valeurs pour les éléments `Name` et `Type` que celles des ensembles d'enregistrements de ressource de latence.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Region` propriété d'un type de `AWS::Route53::RecordSetGroup` `RecordSet` données.

`SetIdentifier`

Ensembles d'enregistrements de ressource ayant une politique de routage autre que simple : un identificateur qui se différencie parmi plusieurs ensembles d'enregistrements de ressource ayant la même combinaison de nom et de type, comme plusieurs ensembles d'enregistrements de ressource pondérés nommés acme.example.com ayant un type A. Dans un groupe d'ensembles d'enregistrements de ressource ayant les mêmes nom et type, la valeur `SetIdentifier` doit être unique pour chaque ensemble d'enregistrements de ressource.

Pour de plus amples informations sur les politiques de routage, veuillez consulter Choix d'une politique de routage dans le Guide du développeur Amazon Route 53.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `SetIdentifier` propriété d'un type de `AWS::Route53::RecordSetGroup` `RecordSet` données.

Exemples

Exemple de configuration Route 53

Cet exemple montre comment configurer Route 53.

YAML

```
Domain:
  DomainName: www.example.com
  CertificateArn: arn-example
```

```
  EndpointConfiguration: EDGE
  Route53:
    HostedZoneId: Z1PA6795UKMFR9
    EvaluateTargetHealth: true
    DistributionDomainName: xyz
```

AWS::Serverless::LayerVersion

Crée un Lambda LayerVersion qui contient la bibliothèque ou le code d'exécution nécessaire à une fonction Lambda.

La AWS::Serverless::LayerVersion ressource prend également en charge l'attribut `Metadata` resource, ce qui vous permet de demander AWS SAM de créer des couches incluses dans votre application. Pour plus d'informations sur les couches de construction, consultez Création de couches Lambda dans AWS SAM.

Remarque importante : depuis la publication de l'attribut de UpdateReplacePolicyressource dans CloudFormation, AWS::Lambda::LayerVersion(recommandé) offre les mêmes avantages queAWS::Serverless::LayerVersion.

Lorsqu'un Serverless LayerVersion est transformé, SAM transforme également l'identifiant logique de la ressource afin que les anciens ne LayerVersions soient pas automatiquement supprimés CloudFormation lors de la mise à jour de la ressource.

> **Note**
>
> Lorsque vous déployez vers AWS CloudFormation, vos AWS SAM ressources sont AWS SAM transformées en CloudFormation ressources. Pour de plus amples informations, veuillez consulter CloudFormation Ressources générées pour AWS SAM.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
Type: AWS::Serverless::LayerVersion
Properties:
```

```
  CompatibleArchitectures: List
  CompatibleRuntimes: List
  ContentUri: String | LayerContent
  Description: String
  LayerName: String
  LicenseInfo: String
  PublishLambdaVersion: Boolean
  RetentionPolicy: String
```

Propriétés

`CompatibleArchitectures`

Précise les architectures de jeux d'instructions prises en charge pour la version de couche.

Pour en savoir plus sur cette propriété, consultez Architectures du jeu d'instructions Lambda dans le guide du développeur AWS Lambda .

Valeurs valides : `x86_64`, `arm64`

Type : liste

Obligatoire : non

Par défaut : `x86_64`

CloudFormation compatibilité : cette propriété est transmise directement à la `CompatibleArchitectures` propriété d'une `AWS::Lambda::LayerVersion` ressource.

`CompatibleRuntimes`

Liste des runtimes compatibles avec cela LayerVersion.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `CompatibleRuntimes` propriété d'une `AWS::Lambda::LayerVersion` ressource.

`ContentUri`

Uri Amazon S3, chemin d'accès au dossier local ou LayerContent objet du code de couche.

Si un URI ou un LayerContent objet Amazon S3 est fourni, l'objet Amazon S3 référencé doit être une archive ZIP valide contenant le contenu d'une couche Lambda.

Si le chemin d'accès vers un fichier local est fourni, le modèle doit passer par le flux comprenant sam build suivie par sam deploy ou sam package, pour que le contenu soit correctement transformé. Par défaut, les chemins relatifs sont résolus en fonction de l'emplacement du AWS SAM modèle.

Type : Chaîne | LayerContent

Obligatoire : oui

CloudFormation compatibilité : cette propriété est similaire à celle `Content` d'une `AWS::Lambda::LayerVersion` ressource. Les propriétés imbriquées d'Amazon S3 sont nommées différemment.

`Description`

Description de cette couche.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Description` propriété d'une `AWS::Lambda::LayerVersion` ressource.

`LayerName`

Nom ou Amazon Resource Name (ARN) de la couche.

Type : chaîne

Obligatoire : non

Par défaut : ID logique de ressource

CloudFormation compatibilité : cette propriété est similaire à celle `LayerName` d'une `AWS::Lambda::LayerVersion` ressource. Si vous ne spécifiez pas de nom, l'ID logique de la ressource sera utilisée comme nom.

`LicenseInfo`

Informations sur la licence correspondante LayerVersion.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `LicenseInfo` propriété d'une `AWS::Lambda::LayerVersion` ressource.

`PublishLambdaVersion`

Une propriété opt-in qui crée une nouvelle version de Lambda chaque fois qu'une modification est apportée à la `LayerVersion` ressource référencée. Lorsqu'elle est activée avec `AutoPublishAlias` et `AutoPublishAliasAllProperties` dans la fonction Lambda connectée, une nouvelle version Lambda sera créée pour chaque modification apportée à la ressource. `LayerVersion`

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`RetentionPolicy`

Cette propriété indique si les anciennes versions de votre ressource `LayerVersion` sont conservées ou supprimées lorsque vous supprimez une ressource. Si vous devez conserver les anciennes versions de votre ressource `LayerVersion` lors de la mise à jour ou du remplacement d'une ressource, l'`UpdateReplacePolicy`attribut doit être activé. Pour plus d'informations à ce sujet, reportez-vous à la section `UpdateReplacePolicy`attribut du guide de AWS CloudFormation l'utilisateur.

Valeurs valides : `Retain` ou `Delete`

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Remarques supplémentaires : Lorsque vous spécifiez`Retain`, AWS SAM ajoute un Attributs de ressources pris en charge par AWS SAM de `DeletionPolicy: Retain` à la `AWS::Lambda::LayerVersion` ressource transformée.

Valeurs renvoyées

Réf

Lorsque l'ID logique de cette ressource est fourni à la fonction `Ref` intrinsèque, elle renvoie l'ARN de la ressource Lambda LayerVersion sous-jacente.

Pour plus d'informations sur l'utilisation de la fonction `Ref`, consultez `Ref` dans le Guide de l'utilisateur AWS CloudFormation .

Exemples

LayerVersionExample

Exemple de 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

Une archive ZIP hébergeant le contenu d'une couche Lambda.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
  Bucket: String
  Key: String
  Version: String
```

Propriétés

`Bucket`

Compartiment Amazon S3 de l'archive de couche.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `S3Bucket` propriété du type de `AWS::Lambda::LayerVersion` `Content` données.

`Key`

Clé Amazon S3 de l’archive de couche.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `S3Key` propriété du type de `AWS::Lambda::LayerVersion` `Content` données.

`Version`

Pour les objets versionnés, version de l’objet d’archive de couche à utiliser.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `S3ObjectVersion` propriété du type de `AWS::Lambda::LayerVersion` `Content` données.

Exemples

LayerContent

Exemple de contenu de couche

YAML

```
LayerContent:
  Bucket: amzn-s3-demo-bucket-name
```

```
  Key: mykey-name
  Version: 121212
```

AWS::Serverless::SimpleTable

Crée une table DynamoDB avec une clé primaire d'attribut unique. Cela est utile lorsque les données doivent seulement être accessibles via une clé primaire.

Pour des fonctionnalités plus avancées, utilisez une AWS::DynamoDB::Tableressource dans CloudFormation. Ces ressources peuvent être utilisées dans. AWS SAM Ils sont complets et offrent une personnalisation supplémentaire, y compris key schemaune resource policypersonnalisation.

> **Note**
>
> Lorsque vous déployez vers AWS CloudFormation, vos AWS SAM ressources sont AWS SAM transformées en CloudFormation ressources. Pour de plus amples informations, veuillez consulter CloudFormation Ressources générées pour AWS SAM.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
Type: AWS::Serverless::SimpleTable
Properties:
  PointInTimeRecoverySpecification: PointInTimeRecoverySpecification
  PrimaryKey: PrimaryKeyObject
  ProvisionedThroughput: ProvisionedThroughputObject
  SSESpecification: SSESpecification
  TableName: String
  Tags: Map
```

Propriétés

`PointInTimeRecoverySpecification`

Paramètres utilisés pour permettre une restauration à un instant dans le passé.

Type : PointInTimeRecoverySpecification

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `PointInTimeRecoverySpecification` propriété d'une `AWS::DynamoDB::Table` ressource.

`PrimaryKey`

Nom et type d'attribut à utiliser comme clé primaire de la table. Si elle n'est pas fournie, la clé primaire sera une `String` avec une valeur de `id`.

> **Note**
>
> La valeur de cette propriété ne peut pas être modifiée après la création de cette ressource.

Type : PrimaryKeyObject

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`ProvisionedThroughput`

Lire et écrire des informations d'approvisionnement du débit.

Si `ProvisionedThroughput` n'est pas spécifié `BillingMode` sera spécifié en tant que `PAY_PER_REQUEST`.

Type : ProvisionedThroughputObject

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `ProvisionedThroughput` propriété d'une `AWS::DynamoDB::Table` ressource.

`SSESpecification`

Spécifie les paramètres visant à activer le chiffrement côté serveur.

Type : SSESpecification

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `SSESpecification` propriété d'une `AWS::DynamoDB::Table` ressource.

`TableName`

Nom de la table DynamoDB.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `TableName` propriété d'une `AWS::DynamoDB::Table` ressource.

`Tags`

Une carte (chaîne à chaîne) qui indique les balises à y ajouter SimpleTable. Pour plus de détails sur les clés et les valeurs valides pour les étiquettes, voir l'étiquette Ressource dans le Guide de l'utilisateur AWS CloudFormation .

Type: carte (map)

Obligatoire : non

CloudFormation compatibilité : cette propriété est similaire à celle `Tags` d'une `AWS::DynamoDB::Table` ressource. La propriété Tags dans SAM se compose de paires Key:Value ; CloudFormation elle consiste en une liste d'objets Tag.

Valeurs renvoyées

Réf

Lorsque l'ID logique de cette ressource est fournie à la fonction intrinsèque Réf, elle renvoie le nom de la ressource de la table DynamoDB sous-jacente.

Pour plus d'informations sur l'utilisation de la fonction `Ref`, consultez `Ref` dans le Guide de l'utilisateur AWS CloudFormation .

Exemples

SimpleTableExample

Exemple de SimpleTable

YAML

```
Properties:
  TableName: my-table
  PrimaryKey:
    Name: MyPrimaryKey
    Type: String
  ProvisionedThroughput:
    ReadCapacityUnits: 5
    WriteCapacityUnits: 5
  Tags:
    Department: Engineering
    AppType: Serverless
```

PrimaryKeyObject

L'objet décrivant les propriétés d'une clé primaire.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
  Name: String
  Type: String
```

Propriétés

`Name`

Nom d'attribut de la clé primaire.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `AttributeName` propriété du type de `AWS::DynamoDB::Table` `AttributeDefinition` données.

Remarques supplémentaires : Cette propriété est également transmise à la AttributeNamepropriété d'un type de `AWS::DynamoDB::Table` `KeySchema` données.

`Type`

Le type de données pour la clé primaire.

Valeurs valides : `String`, `Number`, `Binary`

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `AttributeType` propriété du type de `AWS::DynamoDB::Table` `AttributeDefinition` données.

Exemples

PrimaryKey

Exemple de clé primaire.

YAML

```
Properties:
  PrimaryKey:
    Name: MyPrimaryKey
    Type: String
```

ProvisionedThroughputObject

Objet décrivant les propriétés d'un débit provisionné.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
  ReadCapacityUnits: Integer
```

```
  WriteCapacityUnits: Integer
```

Propriétés

`ReadCapacityUnits`

Nombre maximal de lectures à cohérence forte consommées par seconde avant que DynamoDB ne renvoie une exception `ThrottlingException`.

Type : entier

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `ReadCapacityUnits` propriété du type de `AWS::DynamoDB::Table` `ProvisionedThroughput` données.

`WriteCapacityUnits`

Nombre maximal d'écritures consommées par seconde avant que DynamoDB ne renvoie une exception `ThrottlingException`.

Type : entier

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `WriteCapacityUnits` propriété du type de `AWS::DynamoDB::Table` `ProvisionedThroughput` données.

Exemples

ProvisionedThroughput

Exemple de débit provisionné.

YAML

```
Properties:
   ProvisionedThroughput:
     ReadCapacityUnits: 5
     WriteCapacityUnits: 5
```

AWS::Serverless::StateMachine

Crée une machine à AWS Step Functions états, que vous pouvez utiliser pour orchestrer des AWS Lambda fonctions et d'autres AWS ressources afin de créer des flux de travail complexes et robustes.

Pour plus d'informations sur Step Functions, consultez le AWS Step Functions Guide du développeur .

> **Note**
>
> Lorsque vous déployez vers AWS CloudFormation, vos AWS SAM ressources sont AWS SAM transformées en CloudFormation ressources. Pour de plus amples informations, veuillez consulter CloudFormation Ressources générées pour AWS SAM.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

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
```

Propriétés

`AutoPublishAlias`

Nom de l'alias de la machine d'état. Pour en savoir plus sur l'utilisation des alias de machine d'état Step Functions, consultez la section Gérer les déploiements continus avec des versions et des alias dans le Guide du développeur AWS Step Functions .

Utiliser `DeploymentPreference` pour configurer les préférences de déploiement pour votre alias. Si vous ne le spécifiez pas`DeploymentPreference`, AWS SAM configurera le trafic pour qu'il passe immédiatement à la nouvelle version de State Machine.

AWS SAM définit les versions `DeletionPolicy` et `UpdateReplacePolicy` to `Retain` par défaut. Les versions précédentes ne seront pas supprimées automatiquement.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Name` propriété d'une `AWS::StepFunctions::StateMachineAlias` ressource.

`UseAliasAsEventTarget`

Indiquez s'il faut transmettre l'alias, créé à l'aide de la `AutoPublishAlias` propriété, à la cible de la source d'événements définie avec Events.

Spécifiez `True` d'utiliser l'alias comme cible des événements.

Type : valeur booléenne

Obligatoire : non

Par défaut : `False`

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Definition`

La définition de la machine à états est un objet dont le format correspond au format de votre fichier AWS SAM modèle, par exemple JSON ou YAML. Les définitions de machine d'état respectent le langage des états Amazon.

Pour un exemple de définition de machine d'état en ligne, consultez Exemples.

Vous devez fournir une `Definition` ou une `DefinitionUri`.

Type: carte (map)

Obligatoire : Conditionnelle

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`DefinitionSubstitutions`

string-to-stringCarte qui spécifie les mappages pour les variables d'espace réservé dans la définition de la machine à états. Cela permet au client d'injecter des valeurs obtenues au moment de l'exécution, par exemple à partir de fonctions intrinsèques, dans la définition de la machine d'état.

Type: carte (map)

Obligatoire : non

CloudFormation compatibilité : cette propriété est similaire à celle `DefinitionSubstitutions` d'une `AWS::StepFunctions::StateMachine` ressource. Si des fonctions intrinsèques sont spécifiées dans une définition de machine à états en ligne, AWS SAM ajoute des entrées à cette propriété pour les injecter dans la définition de machine à états.

`DefinitionUri`

L'URI Amazon Simple Storage Service (Amazon S3) ou le chemin d'accès au fichier local de la définition de la machine d'état écrits dans la Langue des états d'Amazon.

Si vous fournissez un chemin d'accès au fichier local, le modèle doit passer par le flux de travail qui inclut le fichier `sam deploy` ou la commande `sam package` pour que la définition soit correctement transformée. Pour ce faire, vous devez utiliser la version 0.52.0 ou ultérieure de la CLI AWS SAM .

Vous devez fournir une `Definition` ou une `DefinitionUri`.

Type : chaîne |Emplacement S3

Obligatoire : Conditionnelle

CloudFormation compatibilité : cette propriété est transmise directement à la `DefinitionS3Location` propriété d'une `AWS::StepFunctions::StateMachine` ressource.

`DeploymentPreference`

Les paramètres qui permettent et configurent les déploiements de machines à états progressifs. Pour en savoir plus sur les déploiements progressifs de Step Functions, consultez la section Gérer les déploiements continus avec des versions et des alias dans le Guide du développeur AWS Step Functions .

Spécifiez `AutoPublishAlias` avant de configurer cette propriété. Vos paramètres `DeploymentPreference` seront appliqués à l'alias spécifié avec `AutoPublishAlias`.

Lorsque vous spécifiez`DeploymentPreference`, AWS SAM génère automatiquement la valeur de la `StateMachineVersionArn` sous-propriété.

Type : DeploymentPreference

Obligatoire : non

CloudFormation compatibilité : AWS SAM génère et attache la valeur de `StateMachineVersionArn` propriété à la propriété `DeploymentPreference` d'une `AWS::StepFunctions::StateMachineAlias` ressource `DeploymentPreference` et la `DeploymentPreference` transmet à celle-ci.

`Events`

Spécifie les événements qui déclenchent cette machine d'état. Les événements sont constitués d'un type et d'un ensemble de propriétés qui dépendent du type.

Type : EventSource

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Logging`

Définit quels événements de l'historique d'exécution sont journalisés et à quel emplacement.

Type : LoggingConfiguration

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `LoggingConfiguration` propriété d'une `AWS::StepFunctions::StateMachine` ressource.

`Name`

Nom de la machine d'état.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `StateMachineName` propriété d'une `AWS::StepFunctions::StateMachine` ressource.

`PermissionsBoundary`

L'ARN d'une limite d'autorisations à utiliser pour le rôle d'exécution de cette machine d'état. Cette propriété ne fonctionne que si le rôle est généré pour vous.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `PermissionsBoundary` propriété d'une `AWS::IAM::Role` ressource.

`Policies`

Politiques d'autorisation pour cette machine d'état. Les politiques seront ajoutées au rôle d'exécution par défaut Gestion des identités et des accès AWS (IAM) de la machine à états.

Cette propriété accepte une valeur unique ou une liste de valeurs. Les valeurs autorisées sont les suivantes :

- Modèles de politique AWS SAM.
- L'ARN d'une politique gérée par AWS ou d'une politique gérée par le client.
- Le nom d'une politique AWS gérée dans la liste suivante.
- Une politique IAM en ligne formatée dans YAML sous forme de mappage.

> **Note**
>
> Si vous définissez la propriété `Role`, cette propriété est ignorée.

Type : chaîne | liste | carte

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`PropagateTags`

Indiquez s'il faut ou non transmettre les balises de la propriété `Tags` aux ressources AWS::Serverless::StateMachine que vous avez générées. Spécifiez `True` pour la propagation des balises dans vos ressources générées.

Type : valeur booléenne

Obligatoire : non

Par défaut : `False`

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Role`

L'ARN d'un rôle IAM à utiliser comme rôle d'exécution de cette machine d'état.

Type : chaîne

Obligatoire : Conditionnelle

CloudFormation compatibilité : cette propriété est transmise directement à la `RoleArn` propriété d'une `AWS::StepFunctions::StateMachine` ressource.

`RolePath`

Le chemin d'accès au rôle d'exécution IAM de la machine d'état.

Utilisez cette propriété lorsque le rôle est généré pour vous. Ne l'utilisez pas lorsque le rôle est spécifié avec la propriété `Role`.

Type : chaîne

Obligatoire : Conditionnelle

CloudFormation compatibilité : cette propriété est transmise directement à la `Path` propriété d'une `AWS::IAM::Role` ressource.

`Tags`

Une string-to-string carte qui spécifie les balises ajoutées à la machine d'état et le rôle d'exécution correspondant. Pour en savoir plus sur les clés et les valeurs valides pour les étiquettes, consultez la propriété Étiquettes d'une ressource AWS::StepFunctions::StateMachine.

Type: carte (map)

Obligatoire : non

CloudFormation compatibilité : cette propriété est similaire à celle `Tags` d'une `AWS::StepFunctions::StateMachine` ressource. AWS SAM ajoute automatiquement une `stateMachine:createdBy:SAM` balise à cette ressource et au rôle par défaut généré pour celle-ci.

`Tracing`

Indique si la machine à états AWS X-Ray est activée ou non. Pour plus d'informations sur l'utilisation de X-Ray avec Step Functions, consultez AWS X-Ray et Step Functions dans le Guide du développeur AWS Step Functions .

Type : TracingConfiguration

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `TracingConfiguration` propriété d'une `AWS::StepFunctions::StateMachine` ressource.

`Type`

Le type de la machine d'état.

Valeurs valides : `STANDARD` ou `EXPRESS`

Type : chaîne

Obligatoire : non

Par défaut : `STANDARD`

CloudFormation compatibilité : cette propriété est transmise directement à la `StateMachineType` propriété d'une `AWS::StepFunctions::StateMachine` ressource.

Valeurs renvoyées

Réf

Lorsque vous fournissez l'ID logique de cette ressource à la fonction intrinsèque Réf, Réf renvoie l'Amazon Resource Name (ARN) de la ressource sous-jacente `AWS::StepFunctions::StateMachine`.

Pour plus d'informations sur l'utilisation de la fonction `Ref`, consultez `Ref` dans le Guide de l'utilisateur AWS CloudFormation .

Ventilateur : GetAtt

`Fn::GetAtt` renvoie une valeur pour un attribut de ce type indiqué. Voici les attributs disponibles et des exemples de valeurs de retour.

Pour plus d'informations sur l'utilisation de `Fn::GetAtt`, consultez `Fn::GetAtt` dans le Guide de l'utilisateur AWS CloudFormation .

`Name`

Renvoie le nom de la machine d'état, comme `HelloWorld-StateMachine`.

Exemples

Fichier de définition de la machine d'état

Voici un exemple de définition de machine à états intégrée qui permet à une fonction lambda d'invoquer une machine à états. Notez que cet exemple attend de la `Role` propriété qu'elle configure une politique appropriée pour autoriser l'invocation. Le fichier `my_state_machine.asl.json` doit être écrit dans la Langue des états Amazon.

Dans cet exemple, les `DefinitionSubstitution` entrées permettent à la machine d'état d'inclure les ressources déclarées dans le fichier AWS SAM modèle.

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
```

Définition de la machine d'état en ligne

Voici un exemple de définition de machine d'état en ligne.

Dans cet exemple, le fichier AWS SAM modèle est écrit en YAML, de sorte que la définition de la machine d'état est également en YAML. Pour déclarer une définition de machine à états intégrée en JSON, écrivez votre fichier AWS SAM modèle en 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

L'objet décrivant la source des événements qui déclenchent la machine d'état. Chaque événement se compose d'un type et d'un ensemble de propriétés qui dépendent de ce type. Pour plus d'informations sur les propriétés de chaque source d'événement, consultez la sous-rubrique correspondant à ce type.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
Properties: Schedule | ScheduleV2 | CloudWatchEvent | EventBridgeRule | Api
Type: String
```

Propriétés

`Properties`

Un objet décrivant les propriétés de ce mappage d'événements. L'ensemble de propriétés doit être conforme au `Type` défini.

Type : Calendrier | ScheduleV2 | | | CloudWatchEventApi EventBridgeRule

Obligatoire : oui

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Type`

Type d'événement.

Valeurs valides : `Api`, `Schedule`, `ScheduleV2`, `CloudWatchEvent`, `EventBridgeRule`

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

API

Voici un exemple d'un événement du type `API`.

YAML

```
ApiEvent:
  Type: Api
  Properties:
    Method: get
```

```
    Path: /group/{user}
    RestApiId:
      Ref: MyApi
```

Api

L'objet décrivant un type de source d'événement `Api`. Si une ressource AWS::Serverless::Api est définie, les valeurs de chemin d'accès et de méthode doivent correspondre à une opération dans la définition OpenAPI de l'API.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
Auth: ApiStateMachineAuth
Method: String
Path: String
RestApiId: String
UnescapeMappingTemplate: Boolean
```

Propriétés

`Auth`

La configuration d'autorisation pour cette API, ce chemin d'accès et cette méthode.

Utilisez cette propriété pour remplacer la propriété `DefaultAuthorizer` pour un chemin d'accès individuel, lorsqu'aucun `DefaultAuthorizer` n'est spécifié, ou pour remplacer le réglage `ApiKeyRequired` par défaut.

Type : ApiStateMachineAuth

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Method`

La méthode HTTP pour laquelle cette fonction est appelée.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Path`

Le chemin d'accès d'URI pour lequel cette fonction est appelée. La valeur doit commencer par `/`.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`RestApiId`

L'identificateur d'une ressource `RestApi`, qui doit contenir une opération avec le chemin et la méthode donnés. Cela est généralement défini pour faire référence à une ressource AWS::Serverless::Api qui est définie dans ce modèle.

Si vous ne définissez pas cette propriété, AWS SAM crée une AWS::Serverless::Api ressource par défaut à l'aide d'un `OpenApi` document généré. Cette ressource contient une union de tous les chemins et méthodes définis par `Api` dans le même modèle qui ne spécifient pas un `RestApiId`.

Cette propriété ne peut pas faire référence à une ressource AWS::Serverless::Api qui est définie dans un autre modèle.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`UnescapeMappingTemplate`

Élimine les guillemets simples, en remplaçant `\'` par `'`, sur l'entrée transmise à la machine d'état. À utiliser lorsque votre entrée contient des guillemets simples.

> **Note**
>
> Si ce paramètre est défini sur `False` et que votre entrée contient des guillemets simples, une erreur se produira.

Type : valeur booléenne

Obligatoire : non

Par défaut : False

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

ApiEvent

Voici un exemple d'un événement du type `Api`.

YAML

```
Events:
  ApiEvent:
    Type: Api
    Properties:
      Path: /path
      Method: get
```

ApiStateMachineAuth

Configure l'autorisation au niveau de l'événement, pour une API, un chemin et une méthode spécifiques.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```yaml
ApiKeyRequired: Boolean
AuthorizationScopes: List
Authorizer: String
ResourcePolicy: ResourcePolicyStatement
```

Propriétés

`ApiKeyRequired`

Nécessite une clé API pour cette API, ce chemin d'accès et cette méthode.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`AuthorizationScopes`

Les étendues d'autorisation à appliquer à cette API, ce chemin d'accès et cette méthode.

Les étendues que vous spécifiez remplaceront toutes les étendues appliquées par la propriété `DefaultAuthorizer` si vous l'avez spécifiée.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Authorizer`

Le `Authorizer` pour une machine d'état spécifique.

Si vous avez spécifié un autorisateur global pour l'API et que vous souhaitez rendre cette machine d'état publique, remplacez le mécanisme d'autorisation global en définissant `Authorizer` sur `NONE`.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`ResourcePolicy`

Configurez la stratégie de ressources pour cette API et ce chemin d'accès.

Type : ResourcePolicyStatement

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

StateMachine-Authentification

L'exemple suivant spécifie l'autorisation au niveau de la machine d'état.

YAML

```
Auth:
  ApiKeyRequired: true
  Authorizer: NONE
```

ResourcePolicyStatement

Configure une stratégie de ressource pour toutes les méthodes et chemins d'accès d'une API. Pour plus d'informations sur les stratégies de ressources, consultez Contrôle de l'accès à une API avec des stratégies de ressources API Gateway dans le Guide du développeur API.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
  AwsAccountBlacklist: List
  AwsAccountWhitelist: List
  CustomStatements: List
  IntrinsicVpcBlacklist: List
```

```
  IntrinsicVpcWhitelist: List
  IntrinsicVpceBlacklist: List
  IntrinsicVpceWhitelist: List
  IpRangeBlacklist: List
  IpRangeWhitelist: List
  SourceVpcBlacklist: List
  SourceVpcWhitelist: List
```

Propriétés

`AwsAccountBlacklist`

Les AWS comptes à bloquer.

Type : liste de chaînes

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`AwsAccountWhitelist`

Les AWS comptes à autoriser. Pour obtenir un exemple d'utilisation de cette propriété, consultez la section Exemples en bas de cette page.

Type : liste de chaînes

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`CustomStatements`

Une liste des instructions de stratégie de ressource personnalisées à appliquer à cette API. Pour obtenir un exemple d'utilisation de cette propriété, consultez la section Exemples en bas de cette page.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`IntrinsicVpcBlacklist`

La liste des clouds privés virtuels (VPCs) à bloquer, où chaque VPC est spécifié comme référence, telle qu'une référence dynamique ou une fonction `Ref` intrinsèque. Pour obtenir un exemple d'utilisation de cette propriété, consultez la section Exemples en bas de cette page.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`IntrinsicVpcWhitelist`

La liste des VPCs à autoriser, où chaque VPC est spécifié comme référence, telle qu'une référence dynamique ou une fonction `Ref` intrinsèque.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`IntrinsicVpceBlacklist`

Liste des points de terminaison VPC à bloquer, où chaque point de terminaison VPC est spécifié comme une référence telle qu'une Référence dynamique ou la fonction intrinsèque `Ref`.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`IntrinsicVpceWhitelist`

Liste des points de terminaison VPC à autoriser, où chaque point de terminaison VPC est spécifié en tant que référence telle qu'une Référence dynamique ou la fonction intrinsèque `Ref`. Pour obtenir un exemple d'utilisation de cette propriété, consultez la section Exemples en bas de cette page.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`IpRangeBlacklist`

L'adresse IP ou les plages d'adresses à bloquer. Pour obtenir un exemple d'utilisation de cette propriété, consultez la section Exemples en bas de cette page.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`IpRangeWhitelist`

L'adresse IP ou les plages d'adresses à autoriser.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`SourceVpcBlacklist`

Les points de terminaison VPC ou VPC source à bloquer. Les noms de VPC source doivent commencer par `"vpc-"` et les noms de point de terminaison VPC source doivent commencer par `"vpce-"`. Pour obtenir un exemple d'utilisation de cette propriété, consultez la section Exemples en bas de cette page.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`SourceVpcWhitelist`

Les points de terminaison VPC ou VPC source à autoriser. Les noms de VPC source doivent commencer par `"vpc-"` et les noms de point de terminaison VPC source doivent commencer par `"vpce-"`.

Type : liste

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

Exemples de stratégie basée sur les ressources

L'exemple suivant bloque deux adresses IP et un VPC source, et autorise un AWS compte.

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

L'objet décrivant un type de source d'événement `CloudWatchEvent`.

AWS Serverless Application Model (AWS SAM) génère une AWS::Events::Ruleressource lorsque ce type d'événement est défini.

Remarque importante : EventBridgeRule c'est le type de source d'événements préféré à utiliser, à la place de`CloudWatchEvent`. `EventBridgeRule`et `CloudWatchEvent` utilisez le même service, la même API et les mêmes CloudFormation ressources sous-jacents. Cependant, AWS SAM ajoutera la prise en charge des nouvelles fonctionnalités uniquement à`EventBridgeRule`.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
EventBusName: String
Input: String
InputPath: String
Pattern: EventPattern
```

Propriétés

`EventBusName`

Le bus d'événements à associer à cette règle. Si vous omettez cette propriété, AWS SAM utilise le bus d'événements par défaut.

Type : chaîne

Obligatoire : non

Par défaut : bus d'événement par défaut

CloudFormation compatibilité : cette propriété est transmise directement à la `EventBusName` propriété d'une `AWS::Events::Rule` ressource.

`Input`

Texte JSON valide transmis à la cible. Si vous utilisez cette propriété, aucun élément du texte de l'événement lui-même n'est transmis à la cible.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Input` propriété d'une `AWS::Events::Rule Target` ressource.

`InputPath`

Lorsque vous ne voulez pas transmettre l'événement correspondant complet, utilisez la propriété `InputPath` pour décrire quelles parties de l'événement transmettre à la cible.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `InputPath` propriété d'une `AWS::Events::Rule Target` ressource.

`Pattern`

Décrit les événements qui sont acheminés vers la cible spécifiée. Pour plus d'informations, consultez la section Événements et modèles d'événements EventBridge dans le guide de EventBridge l'utilisateur Amazon.

Type : EventPattern

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `EventPattern` propriété d'une `AWS::Events::Rule` ressource.

Exemples

CloudWatchEvent

Voici un exemple de type de source d'événement `CloudWatchEvent`.

YAML

```
CWEvent:
  Type: CloudWatchEvent
  Properties:
    Input: '{"Key": "Value"}'
    Pattern:
      detail:
        state:
          - running
```

EventBridgeRule

L'objet décrivant un type de source d'`EventBridgeRule`événement, qui définit votre machine d'état comme cible pour une EventBridge règle Amazon. Pour plus d'informations, consultez Qu'est-ce qu'Amazon EventBridge ? dans le guide de EventBridge l'utilisateur Amazon.

AWS SAM génère une AWS::Events::Ruleressource lorsque ce type d'événement est défini.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
  DeadLetterConfig: DeadLetterConfig
  EventBusName: String
  Input: String
  InputPath: String
  InputTransformer: InputTransformer
  Pattern: EventPattern
  RetryPolicy: RetryPolicy
  RuleName: String
  State: String
  Target: Target
```

Propriétés

`DeadLetterConfig`

Configurez la file d'attente Amazon Simple Queue Service (Amazon SQS) dans EventBridge laquelle les événements sont envoyés après l'échec d'un appel cible. L'invocation peut échouer,

par exemple, lors de l'envoi d'un événement à une fonction Lambda qui n'existe pas ou EventBridge lorsque les autorisations sont insuffisantes pour appeler la fonction Lambda. Pour plus d'informations, consultez la politique relative aux nouvelles tentatives relatives aux événements et l'utilisation des files d'attente contenant des lettres mortes dans le guide de l'utilisateur Amazon. EventBridge

Type : DeadLetterConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est similaire à celle du type de `AWS::Events::Rule Target` données. `DeadLetterConfig` La AWS SAM version de cette propriété inclut des sous-propriétés supplémentaires, au cas où vous souhaiteriez AWS SAM créer la file d'attente de lettres mortes pour vous.

`EventBusName`

Le bus d'événements à associer à cette règle. Si vous omettez cette propriété, AWS SAM utilise le bus d'événements par défaut.

Type : chaîne

Obligatoire : non

Par défaut : bus d'événement par défaut

CloudFormation compatibilité : cette propriété est transmise directement à la `EventBusName` propriété d'une `AWS::Events::Rule` ressource.

`Input`

Texte JSON valide transmis à la cible. Si vous utilisez cette propriété, aucun élément du texte de l'événement lui-même n'est transmis à la cible.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Input` propriété d'une `AWS::Events::Rule Target` ressource.

`InputPath`

Lorsque vous ne voulez pas transmettre l'événement correspondant complet, utilisez la propriété `InputPath` pour décrire quelles parties de l'événement transmettre à la cible.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `InputPath` propriété d'une `AWS::Events::Rule Target` ressource.

`InputTransformer`

Paramètres qui vous permettent de fournir une entrée personnalisée à une cible en fonction de certaines données d'événement. Vous pouvez extraire une ou plusieurs paires clé-valeur à partir de l'événement, puis utiliser ces données pour envoyer l'entrée personnalisée à la cible. Pour plus d'informations, consultez la section Transformation des EventBridge entrées Amazon dans le guide de EventBridge l'utilisateur Amazon.

Type : InputTransformer

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `InputTransformer` propriété d'un type de `AWS::Events::Rule Target` données.

`Pattern`

Décrit les événements qui sont acheminés vers la cible spécifiée. Pour plus d'informations, consultez la section Événements et modèles d'événements EventBridge dans le guide de EventBridge l'utilisateur Amazon.

Type : EventPattern

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `EventPattern` propriété d'une `AWS::Events::Rule` ressource.

`RetryPolicy`

Objet `RetryPolicy` qui inclut des informations sur les paramètres de politique de nouvelle tentative. Pour plus d'informations, consultez la politique relative aux nouvelles tentatives relatives aux événements et l'utilisation des files d'attente contenant des lettres mortes dans le guide de l'utilisateur Amazon. EventBridge

Type : RetryPolicy

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `RetryPolicy` propriété du type de `AWS::Events::Rule` `Target` données.

`RuleName`

Le nom de la règle .

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Name` propriété d'une `AWS::Events::Rule` ressource.

`State`

État de la règle.

Valeurs valides : `[ DISABLED | ENABLED ]`

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `State` propriété d'une `AWS::Events::Rule` ressource.

`Target`

La AWS ressource qui est EventBridge invoquée lorsqu'une règle est déclenchée. Vous pouvez utiliser cette propriété pour spécifier l'ID logique de la cible. Si cette propriété n'est pas spécifiée, AWS SAM génère l'ID logique de la cible.

Type : cible

Obligatoire : non

CloudFormation compatibilité : cette propriété est similaire à celle `Targets` d'une `AWS::Events::Rule` ressource. La version AWS SAM de cette propriété vous permet uniquement de spécifier l'ID logique d'une seule cible.

Exemples

EventBridgeRule

Voici un exemple de type de source d'événement `EventBridgeRule`.

YAML

```
EBRule:
  Type: EventBridgeRule
  Properties:
    Input: '{"Key": "Value"}'
    Pattern:
      detail:
        state:
          - terminated
```

DeadLetterConfig

L'objet utilisé pour spécifier la file d'attente Amazon Simple Queue Service (Amazon SQS) EventBridge où envoie les événements après l'échec d'un appel cible. L'appel peut échouer, par exemple, lors de l'envoi d'un événement à une machine d'état qui n'existe pas, ou des autorisations insuffisantes pour appeler la machine d'état. Pour plus d'informations, consultez la politique relative aux nouvelles tentatives relatives aux événements et l'utilisation des files d'attente contenant des lettres mortes dans le guide de l'utilisateur Amazon. EventBridge

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
  Arn: String
  QueueLogicalId: String
  Type: String
```

Propriétés

`Arn`

L'Amazon Resource Name (ARN) de la file d'attente Amazon SQS spécifiée comme cible pour la file d'attente de lettres mortes.

> **Note**
>
> Spécifiez la propriété `Type` ou la propriété `Arn`, mais pas les deux.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Arn` propriété du type de `AWS::Events::Rule` `DeadLetterConfig` données.

`QueueLogicalId`

Le nom personnalisé de la file d'attente de lettres mortes qui la AWS SAM crée `Type` est spécifié.

> **Note**
>
> Si la propriété `Type` n'est pas définie, cette propriété est ignorée.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Type`

Type de la file d'attente. Lorsque cette propriété est définie, crée AWS SAM automatiquement une file d'attente de lettres mortes et y joint la politique basée sur les ressources nécessaire pour autoriser la ressource à envoyer des événements à la file d'attente.

> **Note**
>
> Spécifiez la propriété `Type` ou la propriété `Arn`, mais pas les deux.

Valeurs valides : `SQS`

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

DeadLetterConfig

DeadLetterConfig

YAML

```
DeadLetterConfig:
  Type: SQS
  QueueLogicalId: MyDLQ
```

Target

Configure la AWS ressource qui est EventBridge invoquée lorsqu'une règle est déclenchée.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
  Id: String
```

Propriétés

`Id`

L'ID logique de la cible.

La valeur de `Id` peut inclure des caractères alphanumériques, des points (`.`), des tirets (`-`) et des tirets du bas (`_`).

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `Id` propriété du type de `AWS::Events::Rule` `Target` données.

Exemples

Cible

YAML

```
EBRule:
  Type: EventBridgeRule
  Properties:
    Target:
      Id: MyTarget
```

Schedule

L'objet décrivant un type de source d'`Schedule`événement, qui définit votre machine à états comme cible d'une EventBridge règle qui se déclenche selon un calendrier. Pour plus d'informations, consultez Qu'est-ce qu'Amazon EventBridge ? dans le guide de EventBridge l'utilisateur Amazon.

AWS Serverless Application Model (AWS SAM) génère une AWS::Events::Ruleressource lorsque ce type d'événement est défini.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
  DeadLetterConfig: DeadLetterConfig
  Description: String
  Enabled: Boolean
  Input: String
  Name: String
  RetryPolicy: RetryPolicy
```

```
  RoleArn: String
  Schedule: String
  State: String
  Target: Target
```

Propriétés

`DeadLetterConfig`

Configurez la file d'attente Amazon Simple Queue Service (Amazon SQS) dans EventBridge laquelle les événements sont envoyés après l'échec d'un appel cible. L'invocation peut échouer, par exemple, lors de l'envoi d'un événement à une fonction Lambda qui n'existe pas ou EventBridge lorsque les autorisations sont insuffisantes pour appeler la fonction Lambda. Pour plus d'informations, consultez la politique relative aux nouvelles tentatives relatives aux événements et l'utilisation des files d'attente contenant des lettres mortes dans le guide de l'utilisateur Amazon. EventBridge

Type : DeadLetterConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est similaire à celle du type de `AWS::Events::Rule` `Target` données. `DeadLetterConfig` La AWS SAM version de cette propriété inclut des sous-propriétés supplémentaires, au cas où vous souhaiteriez AWS SAM créer la file d'attente de lettres mortes pour vous.

`Description`

Description de la règle.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Description` propriété d'une `AWS::Events::Rule` ressource.

`Enabled`

Indique si la règle est activée.

Pour désactiver la règle, définissez cette propriété sur `false`.

> **Note**
>
> Spécifiez la propriété `Enabled` ou la propriété `State`, mais pas les deux.

Type : valeur booléenne

Obligatoire : non

CloudFormation compatibilité : cette propriété est similaire à celle `State` d'une `AWS::Events::Rule` ressource. Si cette propriété est définie sur `true` then`ENABLED`, elle est AWS SAM transmise dans le cas contraire`DISABLED`.

`Input`

Texte JSON valide transmis à la cible. Si vous utilisez cette propriété, aucun élément du texte de l'événement lui-même n'est transmis à la cible.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Input` propriété d'une `AWS::Events::Rule Target` ressource.

`Name`

Le nom de la règle . Si vous ne spécifiez pas de nom, CloudFormation génère un identifiant physique unique et utilise cet identifiant comme nom de règle.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Name` propriété d'une `AWS::Events::Rule` ressource.

`RetryPolicy`

Objet `RetryPolicy` qui inclut des informations sur les paramètres de politique de nouvelle tentative. Pour plus d'informations, consultez la politique relative aux nouvelles tentatives relatives aux événements et l'utilisation des files d'attente contenant des lettres mortes dans le guide de l'utilisateur Amazon. EventBridge

Type : RetryPolicy

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `RetryPolicy` propriété du type de `AWS::Events::Rule` `Target` données.

`RoleArn`

L'ARN du rôle IAM que le EventBridge planificateur utilisera pour la cible lorsque le calendrier est invoqué.

Type : RoleArn

Nécessaire : Non S'il n'est pas fourni, un nouveau rôle sera créé et utilisé.

CloudFormation compatibilité : cette propriété est transmise directement à la `RoleArn` propriété du type de `AWS::Scheduler::Schedule` `Target` données.

`Schedule`

Expression de planification qui détermine quand et à quelle fréquence la règle s'exécute. Pour plus d'informations, consultez Expression de planification des règles.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `ScheduleExpression` propriété d'une `AWS::Events::Rule` ressource.

`State`

État de la règle.

Valeurs acceptées : `DISABLED | ENABLED`

> **Note**
>
> Spécifiez la propriété `Enabled` ou la propriété `State`, mais pas les deux.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `State` propriété d'une `AWS::Events::Rule` ressource.

`Target`

La AWS ressource qui est EventBridge invoquée lorsqu'une règle est déclenchée. Vous pouvez utiliser cette propriété pour spécifier l'ID logique de la cible. Si cette propriété n'est pas spécifiée, AWS SAM génère l'ID logique de la cible.

Type : cible

Obligatoire : non

CloudFormation compatibilité : cette propriété est similaire à celle `Targets` d'une `AWS::Events::Rule` ressource. La AWS SAM version de cette propriété vous permet uniquement de spécifier l'ID logique d'une seule cible.

Exemples

CloudWatch Planifier un événement

CloudWatch Exemple de planification d'un événement

YAML

```
CWSchedule:
  Type: Schedule
  Properties:
    Schedule: 'rate(1 minute)'
    Name: TestSchedule
    Description: test schedule
    Enabled: false
```

DeadLetterConfig

L'objet utilisé pour spécifier la file d'attente Amazon Simple Queue Service (Amazon SQS) EventBridge où envoie les événements après l'échec d'un appel cible. L'appel peut échouer, par exemple, lors de l'envoi d'un événement à une machine d'état qui n'existe pas, ou des autorisations insuffisantes pour appeler la machine d'état. Pour plus d'informations, consultez la politique relative aux nouvelles tentatives relatives aux événements et l'utilisation des files d'attente contenant des lettres mortes dans le guide de l'utilisateur Amazon. EventBridge

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
Arn: String
QueueLogicalId: String
Type: String
```

Propriétés

`Arn`

L'Amazon Resource Name (ARN) de la file d'attente Amazon SQS spécifiée comme cible pour la file d'attente de lettres mortes.

> **Note**
>
> Spécifiez la propriété `Type` ou la propriété `Arn`, mais pas les deux.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Arn` propriété du type de `AWS::Events::Rule` `DeadLetterConfig` données.

`QueueLogicalId`

Le nom personnalisé de la file d'attente de lettres mortes qui la AWS SAM crée `Type` est spécifié.

> **Note**
>
> Si la propriété `Type` n'est pas définie, cette propriété est ignorée.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`Type`

Type de la file d'attente. Lorsque cette propriété est définie, crée AWS SAM automatiquement une file d'attente de lettres mortes et y joint la politique basée sur les ressources nécessaire pour autoriser la ressource à envoyer des événements à la file d'attente.

> **Note**
>
> Spécifiez la propriété `Type` ou la propriété `Arn`, mais pas les deux.

Valeurs valides : `SQS`

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

Exemples

DeadLetterConfig

DeadLetterConfig

YAML

```
DeadLetterConfig:
  Type: SQS
  QueueLogicalId: MyDLQ
```

Target

Configure la AWS ressource qui est EventBridge invoquée lorsqu'une règle est déclenchée.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

YAML

```
Id: String
```

Propriétés

`Id`

L'ID logique de la cible.

La valeur de `Id` peut inclure des caractères alphanumériques, des points (`.`), des tirets (`-`) et des tirets du bas (`_`).

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `Id` propriété du type de `AWS::Events::Rule` `Target` données.

Exemples

Cible

YAML

```
EBRule:
  Type: Schedule
  Properties:
    Target:
      Id: MyTarget
```

ScheduleV2

L'objet décrivant un type de source d'`ScheduleV2`événement, qui définit votre machine d'état comme cible d'un événement Amazon EventBridge Scheduler qui se déclenche selon un calendrier. Pour plus d'informations, consultez Qu'est-ce qu'Amazon EventBridge Scheduler ? dans le guide de l'utilisateur EventBridge du planificateur.

AWS Serverless Application Model (AWS SAM) génère une AWS::Scheduler::Scheduleressource lorsque ce type d'événement est défini.

Syntaxe

Pour déclarer cette entité dans votre modèle AWS Serverless Application Model (AWS SAM), utilisez la syntaxe suivante.

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
```

Propriétés

`DeadLetterConfig`

Configurez la file d'attente Amazon Simple Queue Service (Amazon SQS) dans EventBridge laquelle les événements sont envoyés après l'échec d'un appel cible. L'invocation peut échouer, par exemple, lors de l'envoi d'un événement à une fonction Lambda qui n'existe pas ou EventBridge lorsque les autorisations sont insuffisantes pour appeler la fonction Lambda. Pour plus d'informations, consultez la section Configuration d'une file d'attente de lettres mortes pour le EventBridge planificateur dans le guide de l'utilisateur du EventBridge planificateur.

Type : DeadLetterConfig

Obligatoire : non

CloudFormation compatibilité : cette propriété est similaire à celle du type de `AWS::Scheduler::Schedule` `Target` données. `DeadLetterConfig` La AWS SAM version

de cette propriété inclut des sous-propriétés supplémentaires, au cas où vous souhaiteriez AWS SAM créer la file d'attente de lettres mortes pour vous.

`Description`

Description de la planification.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Description` propriété d'une `AWS::Scheduler::Schedule` ressource.

`EndDate`

Date, au format UTC, avant laquelle la planification peut invoquer sa cible. En fonction de l'expression de récurrence de la planification, les appels peuvent s'arrêter avant, ou au moment de, la EndDate spécifiée.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `EndDate` propriété d'une `AWS::Scheduler::Schedule` ressource.

`FlexibleTimeWindow`

Permet de configurer une fenêtre dans laquelle une planification peut être appelée.

Type : FlexibleTimeWindow

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `FlexibleTimeWindow` propriété d'une `AWS::Scheduler::Schedule` ressource.

`GroupName`

Nom du groupe de planifications à associer à cette planification. S'il n'est pas défini, le groupe par défaut est utilisé.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `GroupName` propriété d'une `AWS::Scheduler::Schedule` ressource.

`Input`

Texte JSON valide transmis à la cible. Si vous utilisez cette propriété, aucun élément du texte de l'événement lui-même n'est transmis à la cible.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Input` propriété d'une `AWS::Scheduler::Schedule Target` ressource.

`KmsKeyArn`

ARN d'une clé KMS utilisée pour chiffrer les données client.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `KmsKeyArn` propriété d'une `AWS::Scheduler::Schedule` ressource.

`Name`

Nom du calendrier. Si vous ne spécifiez pas de nom, AWS SAM génère un nom au format *`StateMachine-Logical-IDEvent-Source-Name`* et utilise cet ID pour le nom du planning.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `Name` propriété d'une `AWS::Scheduler::Schedule` ressource.

`OmitName`

Par défaut, AWS SAM génère et utilise un nom de calendrier au format de*`<State-machine-logical-ID><event-source-name>`*. Définissez cette propriété `true` pour CloudFormation générer un identifiant physique unique et utilisez-le plutôt pour le nom du calendrier.

Type : valeur booléenne

Obligatoire : non

Par défaut : `false`

CloudFormation compatibilité : cette propriété est unique AWS SAM et n'a pas d' CloudFormation équivalent.

`PermissionsBoundary`

ARN de la politique utilisée pour définir la limite d'autorisations du rôle.

> **Note**
>
> S'il `PermissionsBoundary` est défini, les mêmes limites AWS SAM seront appliquées au rôle IAM cible du calendrier du planificateur.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `PermissionsBoundary` propriété d'une `AWS::IAM::Role` ressource.

`RetryPolicy`

Objet `RetryPolicy` qui inclut des informations sur les paramètres de politique de nouvelle tentative.

Type : RetryPolicy

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `RetryPolicy` propriété du type de `AWS::Scheduler::Schedule` `Target` données.

`RoleArn`

L'ARN du rôle IAM que le EventBridge planificateur utilisera pour la cible lorsque le calendrier est invoqué.

Type : RoleArn

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `RoleArn` propriété du type de `AWS::Scheduler::Schedule` `Target` données.

`ScheduleExpression`

Expression de planification qui détermine quand et à quelle fréquence la planification s'exécute.

Type : chaîne

Obligatoire : oui

CloudFormation compatibilité : cette propriété est transmise directement à la `ScheduleExpression` propriété d'une `AWS::Scheduler::Schedule` ressource.

`ScheduleExpressionTimezone`

Fuseau horaire dans lequel l'expression de planification est évaluée.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `ScheduleExpressionTimezone` propriété d'une `AWS::Scheduler::Schedule` ressource.

`StartDate`

Date, au format UTC, après laquelle la planification peut commencer à appeler une cible. En fonction de l'expression de récurrence de la planification, les appels peuvent se dérouler après, ou au moment de, la StartDate spécifiée.

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `StartDate` propriété d'une `AWS::Scheduler::Schedule` ressource.

`State`

État de la planification.

Valeurs acceptées : `DISABLED | ENABLED`

Type : chaîne

Obligatoire : non

CloudFormation compatibilité : cette propriété est transmise directement à la `State` propriété d'une `AWS::Scheduler::Schedule` ressource.

Exemples

Exemple de base de définition d'une ressource ScheduleV2

```
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
```

CloudFormation Ressources générées pour AWS SAM

Cette section fournit des informations détaillées sur les CloudFormation ressources créées lors du traitement AWS SAM de votre AWS modèle. L'ensemble de CloudFormation ressources AWS SAM généré varie en fonction des scénarios que vous spécifiez. Un scénario est la combinaison des ressources et propriétés AWS SAM spécifiées dans votre fichier de modèle. Vous pouvez référencer les ressources CloudFormation générées ailleurs dans votre fichier de modèle, de la même manière que vous référencez les ressources que vous déclarez explicitement dans votre fichier de modèle.

Par exemple, si vous spécifiez une ressource `AWS::Serverless::Function` dans votre fichier de modèle AWS SAM , AWS SAM génère toujours une ressource de base `AWS::Lambda::Function`. Si vous spécifiez également la `AutoPublishAlias` propriété facultative, des AWS SAM sources `AWS::Lambda::Alias` et `AWS::Lambda::Version` des ressources supplémentaires.

Cette section répertorie les scénarios et les CloudFormation ressources qu'ils génèrent, et montre comment référencer les CloudFormation ressources générées dans votre fichier AWS SAM modèle.

Référencement des ressources générées CloudFormation

Vous avez deux options pour référencer les CloudFormation ressources générées dans votre fichier AWS SAM modèle, par `LogicalId` ou par propriété référençable.

Référencement des CloudFormation ressources générées par LogicalId

Les CloudFormation ressources AWS SAM générées possèdent chacune un `LogicalId` identifiant alphanumérique (A-Z, a-z, 0-9) unique dans un fichier modèle. AWS SAM utilise les AWS SAM ressources `LogicalIds` de votre fichier modèle pour créer `LogicalIds` les CloudFormation ressources qu'il génère. Vous pouvez utiliser une CloudFormation ressource générée pour accéder aux propriétés de cette ressource dans votre fichier modèle, comme vous le feriez pour une CloudFormation ressource que vous avez explicitement déclarée. `LogicalId` Pour plus d'informations sur `LogicalIds` les modèles intégrés CloudFormation et les AWS SAM modèles, consultez la section Ressources du guide de AWS CloudFormation l'utilisateur.

> **Note**
>
> Les `LogicalIds` de certaines ressources générées incluent une valeur de hachage unique pour éviter les conflits d'espace de noms. Les `LogicalIds` de ces ressources sont dérivées lorsque la pile est créée. Vous ne pouvez les récupérer qu'une fois que la pile a été créée

à l'aide du AWS Management Console AWS CLI, ou de l'un des AWS SDKs. Nous vous déconseillons de référencer ces ressources par `LogicalId` car les valeurs de hachage peuvent être modifiées.

Référencement des CloudFormation ressources générées par une propriété référençable

AWS SAM Fournit une propriété référençable de la ressource pour certaines ressources générées. AWS SAM Vous pouvez utiliser cette propriété pour référencer une CloudFormation ressource générée et ses propriétés dans votre fichier AWS SAM modèle.

> **Note**
>
> Les CloudFormation ressources générées ne possèdent pas toutes des propriétés référençables. Pour ces ressources, vous devez utiliser le `LogicalId`.

Scénarios CloudFormation de ressources générés

Le tableau suivant récapitule les AWS SAM ressources et les propriétés qui constituent les scénarios qui génèrent des CloudFormation ressources. Les rubriques de la colonne Scénarios fournissent des détails sur les CloudFormation ressources supplémentaires AWS SAM générées pour ce scénario.

AWS SAM ressource	CloudFormation Ressource de base	Scénarios
`AWS::Serverless::Api`	`AWS::ApiGateway::RestApi`	• DomainName la propriété est spécifiée • UsagePlan la propriété est spécifiée
`AWS::Serverless::Application`	`AWS::CloudFormation::Stack`	• Hormis la génération de la CloudFormation ressource de base, il n'existe aucun autre scénario pour cette ressource sans serveur.

AWS SAM ressource	CloudFormation Ressource de base	Scénarios
AWS::Serverless::CapacityProvider	AWS::Lambda::CapacityProvider	• OperatorRole la propriété n'est pas spécifiée
AWS::Serverless::Function	AWS::Lambda::Function	• AutoPublishAlias la propriété est spécifiée • La propriété Role n'est pas spécifiée • DeploymentPreference la propriété est spécifiée • Une source d'événement Api est spécifiée • Une source d' HttpApi événement est spécifiée • Une source d'événements de streaming en continu est spécifiée • Une source d'événement pont d'événements (ou bus d'événements) est spécifiée • Une source d' IotRule événement est spécifiée • OnSuccess (ou OnFailure) la propriété est spécifiée pour les événements Amazon SNS • OnSuccess la propriété (ou OnFailure) est spécifiée pour les événements Amazon SQS

AWS SAM ressource	CloudFormation Ressource de base	Scénarios
AWS::Serverless::HttpApi	AWS::ApiGatewayV2::Api	• StageName la propriété est spécifiée • StageName la propriété n'est pas spécifiée • DomainName la propriété est spécifiée
AWS::Serverless::LayerVersion	AWS::Lambda::LayerVersion	• Hormis la génération de la CloudFormation ressource de base, il n'existe aucun autre scénario pour cette ressource sans serveur.
AWS::Serverless::SimpleTable	AWS::DynamoDB::Table	• Hormis la génération de la CloudFormation ressource de base, il n'existe aucun autre scénario pour cette ressource sans serveur.
AWS::Serverless::StateMachine	AWS::StepFunctions::StateMachine	• La propriété Role n'est pas spécifiée • Une source d'événement d'API est spécifiée • Une source d'événement pont d'événements (ou bus d'événements) est spécifiée

Rubriques

- CloudFormation ressources générées lorsque cela AWS::Serverless::Api est spécifié
- CloudFormation ressources générées lorsque cela AWS::Serverless::Application est spécifié
- CloudFormation ressources générées lorsque cela AWS::Serverless::CapacityProvider est spécifié
- CloudFormation ressources générées lorsque vous spécifiez AWS::Serverless::Connector
- CloudFormation ressources générées lorsque cela AWS::Serverless::Function est spécifié
- CloudFormation ressources générées lorsque cela AWS::Serverless::GraphQLApi est spécifié

- CloudFormation ressources générées lorsque cela AWS::Serverless::HttpApi est spécifié
- CloudFormation ressources générées lorsque cela AWS::Serverless::LayerVersion est spécifié
- CloudFormation ressources générées lorsque cela AWS::Serverless::SimpleTable est spécifié
- CloudFormation ressources générées lorsque cela AWS::Serverless::StateMachine est spécifié

CloudFormation ressources générées lorsque cela AWS::Serverless::Api est spécifié

Lorsqu'un `AWS::Serverless::Api` est spécifié, AWS Serverless Application Model (AWS SAM) génère toujours une CloudFormation ressource `AWS::ApiGateway::RestApi` de base. En outre, il génère toujours une ressource `AWS::ApiGateway::Stage` et `AWS::ApiGateway::Deployment`.

`AWS::ApiGateway::RestApi`

`LogicalId`: *`<api-LogicalId>`*

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

`AWS::ApiGateway::Stage`

`LogicalId`: *`<api-LogicalId><stage-name>`*`Stage`

`<stage-name>` est la chaîne sur laquelle la propriété `StageName` est définie. Par exemple, si vous définissez `StageName` sur `Gamma`, le `LogicalId` est *`MyRestApiGamma`*`Stage`.

Propriété référençable :*`<api-LogicalId>`*`.Stage`

`AWS::ApiGateway::Deployment`

`LogicalId`: *`<api-LogicalId>`*`Deployment`*`<sha>`*

`<sha>` est une valeur de hachage unique qui est générée lors de la création de la pile. Par exemple, *`MyRestApi`*`Deployment`*`926eeb5ff1`*.

Propriété référençable :*`<api-LogicalId>`*`.Deployment`

En plus de ces CloudFormation ressources, lorsqu'elles `AWS::Serverless::Api` sont spécifiées, cela AWS SAM génère des CloudFormation ressources supplémentaires pour les scénarios suivants.

Scénarios

- DomainName la propriété est spécifiée
- UsagePlan la propriété est spécifiée

DomainName la propriété est spécifiée

Lorsque la `DomainName` propriété de la `Domain` propriété de an `AWS::Serverless::Api` est spécifiée, AWS SAM génère la `AWS::ApiGateway::DomainName` CloudFormation ressource.

`AWS::ApiGateway::DomainName`

`LogicalId`: `ApiGatewayDomainName`*`<sha>`*

`<sha>` est une valeur de hachage unique qui est générée lors de la création de la pile. Par exemple : `ApiGatewayDomainName`*`926eeb5ff1`*.

Propriété référençable :*`<api-LogicalId>`*`.DomainName`

UsagePlan la propriété est spécifiée

Lorsque la `UsagePlan` propriété de la `Auth` propriété de an `AWS::Serverless::Api` est spécifiée, AWS SAM génère les CloudFormation ressources suivantes : `AWS::ApiGateway::UsagePlan``AWS::ApiGateway::UsagePlanKey`, et`AWS::ApiGateway::ApiKey`.

`AWS::ApiGateway::UsagePlan`

`LogicalId`: *`<api-LogicalId>`*`UsagePlan`

Propriété référençable :*`<api-LogicalId>`*`.UsagePlan`

`AWS::ApiGateway::UsagePlanKey`

`LogicalId`: *`<api-LogicalId>`*`UsagePlanKey`

Propriété référençable :*`<api-LogicalId>`*`.UsagePlanKey`

`AWS::ApiGateway::ApiKey`

`LogicalId`: *`<api-LogicalId>`*`ApiKey`

Propriété référençable :`<api-LogicalId>.ApiKey`

CloudFormation ressources générées lorsque cela AWS::Serverless::Application est spécifié

Lorsqu'un `AWS::Serverless::Application` est spécifié, AWS Serverless Application Model (AWS SAM) génère une CloudFormation ressource `AWS::CloudFormation::Stack` de base.

`AWS::CloudFormation::Stack`

`LogicalId`: *`<application-LogicalId>`*

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

CloudFormation ressources générées lorsque cela AWS::Serverless::CapacityProvider est spécifié

Lorsqu'une `AWS::Serverless::CapacityProvider` est spécifiée, AWS Serverless Application Model (AWS SAM) génère une ressource CloudFormation de base `AWS::Lambda::CapacityProvider`.

`AWS::Lambda::CapacityProvider`

`LogicalId`: *`<capacityprovider-LogicalId>`*

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

Outre cette CloudFormation ressource, lorsqu'elle `AWS::Serverless::CapacityProvider` est spécifiée, elle génère AWS SAM également CloudFormation des ressources pour les scénarios suivants :

Scénarios

- OperatorRole la propriété n'est pas spécifiée

OperatorRole la propriété n'est pas spécifiée

Lorsque la `OperatorRole` propriété de an n'*`AWS::Serverless::CapacityProvider`*est pas spécifiée, AWS SAM génère une `AWS::IAM::Role` CloudFormation ressource à laquelle est attachée la politique `AWSLambdaManagedEC2ResourceOperator` gérée.

`AWS::IAM::Role`

`LogicalId`: *`<capacityprovider-LogicalId>`*`OperatorRole`

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

CloudFormation ressources générées lorsque vous spécifiez AWS::Serverless::Connector

> **Note**
>
> Lorsque vous définissez des connecteurs via la propriété `Connectors` propriété, celle-ci est d'abord transformée en ressource `AWS::Serverless::Connector` avant de générer ces ressources.

Lorsque vous spécifiez une `AWS::Serverless::Connector` ressource dans un AWS SAM modèle, AWS SAM génère les AWS CloudFormation ressources suivantes selon les besoins.

`AWS::IAM::ManagedPolicy`

`LogicalId`:*`<connector-LogicalId>`*`Policy`

Propriété référençable : N/A (Pour référencer cette CloudFormation ressource, vous devez utiliser le`LogicalId`.)

`AWS::SNS::TopicPolicy`

`LogicalId`:*`<connector-LogicalId>`*`TopicPolicy`

Propriété référençable : N/A (Pour référencer cette CloudFormation ressource, vous devez utiliser le`LogicalId`.)

`AWS::SQS::QueuePolicy`

`LogicalId`:*`<connector-LogicalId>`*`QueuePolicy`

Propriété référençable : N/A (Pour référencer cette CloudFormation ressource, vous devez utiliser le`LogicalId`.)

`AWS::Lambda::Permission`

`LogicalId`:*`<connector-LogicalId><permission>`*`LambdaPermission`

`<permission>` est une autorisation spécifiée par la propriété `Permissions`. Par exemple, `Write`.

Propriété référençable : N/A (Pour référencer cette CloudFormation ressource, vous devez utiliser le`LogicalId`.)

CloudFormation ressources générées lorsque cela AWS::Serverless::Function est spécifié

Lorsqu'un `AWS::Serverless::Function` est spécifié, AWS Serverless Application Model (AWS SAM) crée toujours une CloudFormation ressource `AWS::Lambda::Function` de base.

`AWS::Lambda::Function`

`LogicalId`: *`<function-LogicalId>`*

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

En plus de cette CloudFormation ressource, lorsqu'elle `AWS::Serverless::Function` est spécifiée, elle génère AWS SAM également CloudFormation des ressources pour les scénarios suivants.

Scénarios

- Propriétés des fonctions de base
- Sources des évènements
- Configuration de l'événement

Propriétés des fonctions de base

Les scénarios suivants génèrent des CloudFormation ressources en fonction des propriétés des fonctions principales :

La propriété Role n'est pas spécifiée

Lorsque la `Role` propriété de an n'*`AWS::Serverless::Function`*est pas spécifiée, AWS SAM génère une `AWS::IAM::Role` CloudFormation ressource.

`AWS::IAM::Role`

`LogicalId`: *`<function-LogicalId>`*`Role`

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

AutoPublishAlias la propriété est spécifiée

Lorsque la `AutoPublishAlias` propriété de an `AWS::Serverless::Function` est spécifiée, AWS SAM génère les CloudFormation ressources suivantes : `AWS::Lambda::Alias` et`AWS::Lambda::Version`.

`AWS::Lambda::Alias`

`LogicalId`: *`<function-LogicalId>`*`Alias`*`<alias-name>`*

`<alias-name>` est la chaîne sur laquelle `AutoPublishAlias` est défini. Par exemple, si vous définissez sur `AutoPublishAliaslive`, `LogicalId` c'est : *`MyFunction`* Alias*`live`*.

Propriété référençable :*`<function-LogicalId>`*`.Alias`

`AWS::Lambda::Version`

`LogicalId`: *`<function-LogicalId>`*`Version`*`<sha>`*

`<sha>` est une valeur de hachage unique qui est générée lors de la création de la pile. Par exemple, *`MyFunction`* Version*`926eeb5ff1`*.

Propriété référençable :*`<function-LogicalId>`*`.Version`

Pour plus d'informations sur la `AutoPublishAlias` propriété, consultez la section Propriétés de AWS::Serverless::Function.

DeploymentPreference la propriété est spécifiée

Lorsque la `DeploymentPreference` propriété de an `AWS::Serverless::Function` est spécifiée, AWS SAM génère les CloudFormation ressources suivantes : `AWS::CodeDeploy::Application` et`AWS::CodeDeploy::DeploymentGroup`. En outre, si la `Role` propriété de l'`DeploymentPreference`objet n'est pas spécifiée, génère AWS SAM également une `AWS::IAM::Role` CloudFormation ressource.

`AWS::CodeDeploy::Application`

`LogicalId`: `ServerlessDeploymentApplication`

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

`AWS::CodeDeploy::DeploymentGroup`

`LogicalId`: *`<function-LogicalId>`*`DeploymentGroup`

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

`AWS::IAM::Role`

`LogicalId`: `CodeDeployServiceRole`

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

FunctionUrlConfig la propriété est spécifiée

Lorsque la `FunctionUrlConfig` propriété est spécifiée, AWS SAM génère différentes CloudFormation ressources en fonction de`AuthType`.

Lorsque `AuthType: NONE` cela est spécifié, AWS SAM génère les CloudFormation ressources suivantes :

`AWS::Lambda::Permission`(Invoquer l'accès)

`LogicalId`: *`<function-LogicalId>`*`URLInvokeAllowPublicAccess`

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

`AWS::Lambda::Permission`(Accès public)

`LogicalId`: *`<function-LogicalId>`*`UrlPublicPermissions`

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

`AWS::Lambda::Url`

`LogicalId`: *`<function-LogicalId>`*`Url`

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

Lorsqu'il `AuthType: AWS_IAM` est spécifié, AWS SAM génère uniquement :

`AWS::Lambda::Url`

`LogicalId`: *`<function-LogicalId>`*`Url`

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

Pour plus d'informations sur la `FunctionUrlConfig` propriété, voir FunctionUrlConfig.

Sources des évènements

Les scénarios suivants génèrent des CloudFormation ressources en fonction des sources d'événements :

Une source d'événement Api est spécifiée

Lorsque la `Event` propriété de an `AWS::Serverless::Function` est définie sur`Api`, mais que la `RestApiId` propriété n'est pas spécifiée, AWS SAM la `AWS::ApiGateway::RestApi` CloudFormation ressource est générée.

`AWS::ApiGateway::RestApi`

`LogicalId`: `ServerlessRestApi`

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

Une source d' HttpApi événement est spécifiée

Lorsque la `Event` propriété de an `AWS::Serverless::Function` est définie sur`HttpApi`, mais que la `ApiId` propriété n'est pas spécifiée, AWS SAM la `AWS::ApiGatewayV2::Api` CloudFormation ressource est générée.

`AWS::ApiGatewayV2::Api`

`LogicalId`: `ServerlessHttpApi`

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

Une source d'événements de streaming en continu est spécifiée

Lorsque la `Event` propriété de an `AWS::Serverless::Function` est définie sur l'un des types de streaming, AWS SAM génère la `AWS::Lambda::EventSourceMapping` CloudFormation ressource. Cela s'applique aux types suivants : `DynamoDB`, `Kinesis`, `MQ`, `MSK`, et `SQS`.

`AWS::Lambda::EventSourceMapping`

`LogicalId`: *`<function-LogicalId><event-LogicalId>`*

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

Une source d'événement pont d'événements (ou bus d'événements) est spécifiée

Lorsque la `Event` propriété de an `AWS::Serverless::Function` est définie sur l'un des types de pont d'événements (ou bus d'événements), AWS SAM génère la `AWS::Events::Rule` CloudFormation ressource. Cela s'applique aux types suivants : `EventBridgeRule`, `Schedule`, et `CloudWatchEvents`.

`AWS::Events::Rule`

`LogicalId`: *`<function-LogicalId><event-LogicalId>`*

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

Une source d' IotRule événement est spécifiée

Lorsque la `Event` propriété de an `AWS::Serverless::Function` est définie sur IoTRule, AWS SAM génère la `AWS::IoT::TopicRule` CloudFormation ressource.

`AWS::IoT::TopicRule`

`LogicalId`: *`<function-LogicalId><event-LogicalId>`*

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

Configuration de l'événement

Les scénarios suivants génèrent des CloudFormation ressources en fonction de la configuration des événements :

OnSuccess (ou OnFailure) la propriété est spécifiée pour les événements Amazon SNS

Lorsque la propriété `OnSuccess` (ou`OnFailure`) de la `DestinationConfig` propriété de la `EventInvokeConfig` propriété d'un `AWS::Serverless::Function` est spécifiée, et que le type de destination est spécifié `SNS` mais que l'ARN de destination n'est pas spécifié, AWS SAM génère les CloudFormation ressources suivantes : `AWS::Lambda::EventInvokeConfig` et`AWS::SNS::Topic`.

`AWS::Lambda::EventInvokeConfig`

`LogicalId`: *`<function-LogicalId>`*`EventInvokeConfig`

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

`AWS::SNS::Topic`

`LogicalId`: *`<function-LogicalId>`*`OnSuccessTopic` (ou*`<function-LogicalId>`*`OnFailureTopic`)

Propriété référençable :*`<function-LogicalId>`*`.DestinationTopic`

Si les deux `OnSuccess` et `OnFailure` sont spécifiés pour un événement Amazon SNS, pour faire la distinction entre les ressources générées, vous devez utiliser la `LogicalId`.

OnSuccess la propriété (ou OnFailure) est spécifiée pour les événements Amazon SQS

Lorsque la propriété `OnSuccess` (ou`OnFailure`) de la `DestinationConfig` propriété de la `EventInvokeConfig` propriété d'un `AWS::Serverless::Function` est spécifiée, et que le type de destination est spécifié `SQS` mais que l'ARN de destination n'est pas spécifié, AWS SAM génère les CloudFormation ressources suivantes : `AWS::Lambda::EventInvokeConfig` et`AWS::SQS::Queue`.

`AWS::Lambda::EventInvokeConfig`

`LogicalId`: *`<function-LogicalId>`*`EventInvokeConfig`

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

`AWS::SQS::Queue`

`LogicalId`: *`<function-LogicalId>`*`OnSuccessQueue` (ou*`<function-LogicalId>`*`OnFailureQueue`)

Propriété référençable :*`<function-LogicalId>`*`.DestinationQueue`

Si les deux `OnSuccess` et `OnFailure` sont spécifiés pour un événement Amazon SQS, pour faire la distinction entre les ressources générées, vous devez utiliser la `LogicalId`.

CloudFormation ressources générées lorsque cela AWS::Serverless::GraphQLApi est spécifié

Lorsque vous spécifiez une `AWS::Serverless::GraphQLApi` ressource dans un modèle AWS Serverless Application Model (AWS SAM), crée AWS SAM toujours les AWS CloudFormation ressources de base suivantes.

`AWS::AppSync::DataSource`

`LogicalId`: *`<graphqlapi-LogicalId><datasource-RelativeId><datasource-Type>`*`DataSource`

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

`AWS::AppSync::FunctionConfiguration`

`LogicalId`: *`<graphqlapi-LogicalId><function-RelativeId>`*

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

`AWS::AppSync::GraphQLApi`

`LogicalId`: *`<graphqlapi-LogicalId>`*

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

`AWS::AppSync::GraphQLSchema`

`LogicalId`: *`<graphqlapi-LogicalId>`*`Schema`

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

`AWS::AppSync::Resolver`

`LogicalId`: *`<graphqlapi-LogicalId><OperationType><resolver-RelativeId>`*

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

En plus de ces CloudFormation ressources, `AWS::Serverless::GraphQLApi` le moment spécifié AWS SAM peut également générer les CloudFormation ressources suivantes.

`AWS::AppSync::ApiCache`

`LogicalId`: *`<graphqlapi-LogicalId>`*`ApiCache`

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

`AWS::AppSync::ApiKey`

`LogicalId`: *`<graphqlapi-LogicalId><apikey-RelativeId>`*

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

`AWS::AppSync::DomainName`

`LogicalId`: *`<graphqlapi-LogicalId>`*`DomainName`

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

`AWS::AppSync::DomainNameApiAssociation`

`LogicalId`: *`<graphqlapi-LogicalId>`*`DomainNameApiAssociation`

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

AWS SAM peut également utiliser la `AWS::Serverless::Connector` ressource pour octroyer des autorisations. Pour de plus amples informations, veuillez consulter CloudFormation ressources générées lorsque vous spécifiez AWS::Serverless::Connector.

CloudFormation ressources générées lorsque cela AWS::Serverless::HttpApi est spécifié

Lorsqu'un `AWS::Serverless::HttpApi` est spécifié, AWS Serverless Application Model (AWS SAM) génère une CloudFormation ressource `AWS::ApiGatewayV2::Api` de base.

`AWS::ApiGatewayV2::Api`

`LogicalId`: *`<httpapi-LogicalId>`*

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

En plus de cette CloudFormation ressource, lorsqu'elle `AWS::Serverless::HttpApi` est spécifiée, elle génère AWS SAM également CloudFormation des ressources pour les scénarios suivants :

Scénarios

- StageName la propriété est spécifiée

- StageName la propriété n'est pas spécifiée
- DomainName la propriété est spécifiée

StageName la propriété est spécifiée

Lorsque la `StageName` propriété de an `AWS::Serverless::HttpApi` est spécifiée, AWS SAM génère la `AWS::ApiGatewayV2::Stage` CloudFormation ressource.

`AWS::ApiGatewayV2::Stage`

`LogicalId`: *`<httpapi-LogicalId><stage-name>`*`Stage`

`<stage-name>` est la chaîne sur laquelle la propriété `StageName` est définie. Par exemple, si vous définissez sur `StageNameGamma`, `LogicalId` c'est : *`MyHttpApiGamma`* Stage.

Propriété référençable :*`<httpapi-LogicalId>`*`.Stage`

StageName la propriété n'est pas spécifiée

Lorsque la `StageName` propriété de an n'*`AWS::Serverless::HttpApi`*est pas spécifiée, AWS SAM génère la `AWS::ApiGatewayV2::Stage` CloudFormation ressource.

`AWS::ApiGatewayV2::Stage`

`LogicalId`: *`<httpapi-LogicalId>`*`ApiGatewayDefaultStage`

Propriété référençable :*`<httpapi-LogicalId>`*`.Stage`

DomainName la propriété est spécifiée

Lorsque la `DomainName` propriété de la `Domain` propriété de an `AWS::Serverless::HttpApi` est spécifiée, AWS SAM génère la `AWS::ApiGatewayV2::DomainName` CloudFormation ressource.

`AWS::ApiGatewayV2::DomainName`

`LogicalId`: `ApiGatewayDomainNameV2`*`<sha>`*

`<sha>` est une valeur de hachage unique qui est générée lors de la création de la pile. Par exemple, `ApiGatewayDomainNameV2`*`926eeb5ff1`*.

Propriété référençable :*`<httpapi-LogicalId>`*`.DomainName`

CloudFormation ressources générées lorsque cela AWS::Serverless::LayerVersion est spécifié

Lorsqu'un `AWS::Serverless::LayerVersion` est spécifié, AWS Serverless Application Model (AWS SAM) génère une CloudFormation ressource `AWS::Lambda::LayerVersion` de base.

`AWS::Lambda::LayerVersion`

`LogicalId`: *`<layerversion-LogicalId>`*

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

CloudFormation ressources générées lorsque cela AWS::Serverless::SimpleTable est spécifié

Lorsqu'un `AWS::Serverless::SimpleTable` est spécifié, AWS Serverless Application Model (AWS SAM) génère une CloudFormation ressource `AWS::DynamoDB::Table` de base.

`AWS::DynamoDB::Table`

`LogicalId`: *`<simpletable-LogicalId>`*

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

CloudFormation ressources générées lorsque cela AWS::Serverless::StateMachine est spécifié

Lorsqu'une `AWS::Serverless::StateMachine` est spécifiée, AWS Serverless Application Model (AWS SAM) génère une ressource CloudFormation de base `AWS::StepFunctions::StateMachine`.

`AWS::StepFunctions::StateMachine`

`LogicalId`: *`<statemachine-LogicalId>`*

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

Outre cette CloudFormation ressource, lorsqu'elle `AWS::Serverless::StateMachine` est spécifiée, elle génère AWS SAM également CloudFormation des ressources pour les scénarios suivants :

Scénarios

- La propriété Role n'est pas spécifiée
- Une source d'événement d'API est spécifiée
- Une source d'événement pont d'événements (ou bus d'événements) est spécifiée

La propriété Role n'est pas spécifiée

Lorsque la `Role` propriété de an n'*`AWS::Serverless::StateMachine`*est pas spécifiée, AWS SAM génère une `AWS::IAM::Role` CloudFormation ressource.

`AWS::IAM::Role`

`LogicalId`: *`<statemachine-LogicalId>`*`Role`

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

Une source d'événement d'API est spécifiée

Lorsque la `Event` propriété de an `AWS::Serverless::StateMachine` est définie sur`Api`, mais que la `RestApiId` propriété n'est pas spécifiée, AWS SAM la `AWS::ApiGateway::RestApi` CloudFormation ressource est générée.

`AWS::ApiGateway::RestApi`

`LogicalId`: `ServerlessRestApi`

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

Une source d'événement pont d'événements (ou bus d'événements) est spécifiée

Lorsque la `Event` propriété de an `AWS::Serverless::StateMachine` est définie sur l'un des types de pont d'événements (ou bus d'événements), AWS SAM génère la `AWS::Events::Rule`

CloudFormation ressource. Cela s'applique aux types suivants : `EventBridgeRule`, `Schedule`, et `CloudWatchEvents`.

`AWS::Events::Rule`

`LogicalId`: *`<statemachine-LogicalId><event-LogicalId>`*

Propriété référençable : N/A (vous devez utiliser le `LogicalId` pour référencer cette CloudFormation ressource)

Attributs de ressources pris en charge par AWS SAM

Les attributs de ressources sont des attributs auxquels vous pouvez ajouter AWS SAM CloudFormation des ressources pour contrôler des comportements et des relations supplémentaires. Pour plus d'informations sur les attributs des ressources, consultez Référence d'attribut de ressource dans le Guide de l'utilisateur AWS CloudFormation .

AWS SAM prennent en charge un sous-ensemble d'attributs de ressources définis par CloudFormation. Parmi les attributs de ressources pris en charge, certains sont copiés uniquement dans la CloudFormation ressource générée de base de la AWS SAM ressource correspondante, tandis que d'autres sont copiés dans toutes les CloudFormation ressources générées à partir de la AWS SAM ressource correspondante. Pour plus d'informations sur CloudFormation les ressources générées à partir AWS SAM des ressources correspondantes, consultezCloudFormation Ressources générées pour AWS SAM.

Le tableau suivant récapitule la prise en charge des attributs de ressources par AWS SAM, sous réserve de ce qui est Exceptions indiqué ci-dessous.

Attributs de ressource	Ressource(s) Destination générée(s)
DependsOn Métadonnées [1, 2]	Ressource CloudFormation générée par la base uniquement. Pour plus d'informations sur le mappage entre les AWS SAM ressources et les CloudFormation ressources de base, consultezScénarios CloudFormation de ressources générés.
Condition DeletionPolicy	Toutes les CloudFormation ressources générées à partir de la AWS SAM ressource correspondante. Pour plus d'informations sur les scénarios

Attributs de ressource	Ressource(s) Destination générée(s)
UpdateReplacePolicy	relatifs CloudFormation aux ressources générées, consultezScénarios CloudFormation de ressources générés.

Remarques :

1. Pour plus d'informations sur l'utilisation de l'attribut de ressource `Metadata` avec le type de ressource `AWS::Serverless::Function`, consultez Création de fonctions Lambda avec des environnements d'exécution personnalisés dans AWS SAM.
2. Pour plus d'informations sur l'utilisation de l'attribut de ressource `Metadata` avec le type de ressource `AWS::Serverless::LayerVersion`, consultez Création de couches Lambda dans AWS SAM.

Exceptions

Il existe un certain nombre d'exceptions aux règles d'attribut de ressource décrites précédemment :

- Car`AWS::Lambda::LayerVersion`, le champ personnalisé AWS SAM-only `RetentionPolicy` définit `DeletionPolicy` les CloudFormation ressources générées. La priorité est plus élevée que pour `DeletionPolicy`. Si aucun n'est défini, `DeletionPolicy` est alors défini par défaut sur `Retain`.
- Pour `AWS::Lambda::Version`, si `DeletionPolicy` n'est pas spécifié, la valeur par défaut est `Retain`.
- Dans le cas où une fonction sans serveur `DeploymentPreferences` est spécifiée, les attributs de ressources ne sont pas copiés dans les CloudFormation ressources générées suivantes :
  - `AWS::CodeDeploy::Application`
  - `AWS::CodeDeploy::DeploymentGroup`
  - Le `AWS::IAM::Role` nommé `CodeDeployServiceRole`, qui est créé pour ce scénario
- Si votre AWS SAM modèle contient plusieurs fonctions dont les sources d'événements d'API sont créées implicitement, les fonctions partageront la `AWS::ApiGateway::RestApi` ressource générée. Dans ce scénario, si les fonctions ont des attributs de ressource différents, pour la `AWS::ApiGateway::RestApi` ressource générée, AWS SAM copie les attributs de ressource conformément aux listes de priorité suivantes :

- `UpdateReplacePolicy`:
  1. `Retain`
  2. `Snapshot`
  3. `Delete`
- `DeletionPolicy`:
  1. `Retain`
  2. `Delete`

Extensions API Gateway pour AWS SAM

Les extensions API Gateway, spécialement conçues pour AWS, fournissent des personnalisations et des fonctionnalités supplémentaires pour la conception et la gestion APIs. Les extensions API Gateway sont des extensions de la spécification OpenAPI qui prennent en charge l'autorisation AWS spécifique et les intégrations d'API spécifiques à API Gateway. Pour plus d'informations sur les extensions API Gateway, reportez-vous à la section Extensions API Gateway pour OpenAPI.

AWS SAM prend en charge un sous-ensemble d'extensions API Gateway. Pour savoir quelles extensions API Gateway sont prises en charge AWS SAM, reportez-vous au tableau suivant.

Extension API Gateway	Soutenu par AWS SAM
x-amazon-apigateway-anyObjet -method	Oui
x-amazon-apigateway-apiPropriété -key-source	Non
x-amazon-apigateway-auth Objet	Oui
x-amazon-apigateway-authorizer Objet	Oui
x-amazon-apigateway-authtype Propriété	Oui
x-amazon-apigateway-binaryPropriété -media-types	Oui
x-amazon-apigateway-documentation Objet	Non
x-amazon-apigateway-endpointObjet -configuration	Non

x-amazon-apigateway-gatewayObjet -responses	Oui
x-amazon-apigateway-gateway-Responses.GatewayResponse Objet	Oui
x-amazon-apigateway-gatewayObjet -ResponseParameters	Oui
x-amazon-apigateway-gatewayObjet -ResponseTemplates	Oui
x-amazon-apigateway-integration Objet	Oui
x-amazon-apigateway-integrationObjet .RequestTemplates	Oui
x-amazon-apigateway-integrationObjet .RequestParameters	Non
x-amazon-apigateway-integrationObjet .responses	Oui
x-amazon-apigateway-integrationObjet .response	Oui
x-amazon-apigateway-integrationObjet .ResponseTemplates	Oui
x-amazon-apigateway-integrationObjet .ResponseParameters	Oui
x-amazon-apigateway-requestPropriété -validator	Non
x-amazon-apigateway-requestObjet -validators	Non
x-amazon-apigateway-requestObjet -Validators.RequestValidator	Non

Fonctions intrinsèques pour AWS SAM

Les fonctions intrinsèques sont des fonctions intégrées qui vous permettent d'attribuer des valeurs à des propriétés qui ne sont disponibles qu'au moment de l'exécution. AWS SAM a un support limité pour certaines propriétés intrinsèques des fonctions, il est donc incapable de résoudre certaines fonctions intrinsèques. Par conséquent, nous vous recommandons d'ajouter la `AWS::LanguageExtensions` transformation pour résoudre ce problème. `AWS::LanguageExtensions`Il s'agit d'une macro hébergée par CloudFormation qui vous permet d'utiliser des fonctions intrinsèques et d'autres fonctionnalités qui ne sont pas incluses par défaut CloudFormation.

```
Transform:
```

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

> **Note**
>
> Si vous utilisez des fonctions intrinsèques dans les CodeUri propriétés, vous ne AWS SAM pourrez pas analyser correctement les valeurs. Envisagez plutôt d'utiliser la `AWS::LanguageExtensions` transformation.
> Pour plus d'informations, reportez-vous à la section Propriétés de AWS::Serverless::Function.

Pour plus d'informations sur les fonctions intrinsèques, consultez Référence des fonctions intrinsèques dans le AWS CloudFormation Guide de l'utilisateur.

Exemples d'applications sans serveur pour AWS SAM

Cette section inclut deux exemples d'applications : l'une qui traite les événements DynamoDB et l'autre qui traite les événements Amazon S3. Chaque exemple décrit le step-by-step processus de création d'une application. En outre, les deux incluent des détails sur la configuration des sources d'événements et AWS des ressources. Les deux commencent par identifier ce qui doit être fait avant de commencer et suivent les étapes d'initialisation, de test, d'empaquetage et de déploiement de votre application.

Rubriques

- Traitez les événements DynamoDB avec AWS SAM
- Traitez les événements Amazon S3 avec AWS SAM

Traitez les événements DynamoDB avec AWS SAM

Avec cet exemple d'application, vous construisez en utilisant ce que vous avez appris dans la présentation et le Guide de démarrage rapide, et installez un autre exemple d'application. Cette application se compose d'une fonction Lambda appelée par une source d'événement de table DynamoDB. La fonction Lambda est très simple : elle journalise les données qui ont été transmises par le message source de l'événement.

Cet exercice vous montre comment imiter les messages source d'événements qui sont transmis aux fonctions Lambda lorsqu'elles sont appelées.

Avant de commencer

Assurez-vous que vous avez terminé la configuration requise dans Installer la CLI AWS SAM.

Étape 1 : initialiser l'application

Dans cette section, vous allez télécharger le package de candidature, qui se compose d'un AWS SAM modèle et d'un code d'application.

Pour initialiser l'application

1. Exécutez la commande suivante à l'invite de commande de la CLI AWS SAM.

```
sam init \
--location gh:aws-samples/cookiecutter-aws-sam-dynamodb-python \
--no-input
```

Notez que `gh:` dans la commande ci-dessus, elle est étendue à l' GitHub URL`https://github.com/`.

2. Vérifiez le contenu du répertoire créé par la commande (`dynamodb_event_reader/`) :

   - `template.yaml`— Définit deux AWS ressources dont l'application Read DynamoDB a besoin : une fonction Lambda et une table DynamoDB. Le modèle définit également le mappage entre les deux ressources.
   - Répertoire `read_dynamodb_event/` – Contient le code d'application DynamoDB.

Étape 2 : tester l'application localement

Pour les tests locaux, utilisez la CLI AWS SAM pour générer un exemple d'événement DynamoDB et invoquer la fonction Lambda :

```
sam local generate-event dynamodb update | sam local invoke --event - ReadDynamoDBEvent
```

La `generate-event` commande crée un message source d'événement de test, comme les messages créés lorsque tous les composants sont déployés dans le AWS cloud. Ce message de source d'événement est redirigé vers la fonction ReadDynamo DBEvent Lambda.

Vérifiez que les messages attendus soient imprimés sur la console, en fonction du code source dans `app.py`.

Étape 3 : créer le package de l'application

Après avoir testé votre application localement, vous utilisez le AWS SAMCLI pour créer un package de déploiement, que vous utilisez pour déployer l'application AWS dans le cloud.

Pour créer un package de déploiement Lambda

1. Créez un compartiment S3 à l'emplacement où vous souhaitez enregistrer le code empaqueté. Si vous souhaitez utiliser un compartiment S3 existant, ignorez cette étape.

```
aws s3 mb s3://bucketname
```

2. Créez le package de déploiement en exécutant la commande CLI `package` suivante.

```
sam package \
    --template-file template.yaml \
    --output-template-file packaged.yaml \
    --s3-bucket bucketname
```

Vous spécifiez le nouveau fichier de modèle, `packaged.yaml`, lorsque vous déployez l'application dans l'étape suivante.

Étape 4 : déployer l'application

Maintenant que vous avez créé le package de déploiement, vous l'utilisez pour déployer l'application AWS dans le cloud. Vous testez ensuite l'application.

Pour déployer l'application sans serveur dans le cloud AWS

- Dans le AWS SAMCLI, utilisez la commande `deploy` CLI pour déployer toutes les ressources que vous avez définies dans le modèle.

```
sam deploy \
    --template-file packaged.yaml \
    --stack-name sam-app \
    --capabilities CAPABILITY_IAM \
    --region us-east-1
```

Dans la commande, le paramètre `--capabilities` permet à AWS CloudFormation de créer un rôle IAM.

CloudFormation crée les AWS ressources définies dans le modèle. Vous pouvez accéder aux noms de ces ressources dans la CloudFormation console.

Pour tester l'application sans serveur dans le cloud AWS

1. Ouvrez la console DynamoDB.

2. Insérez un enregistrement dans la table que vous venez de créer.
3. Accédez à l'onglet Mesures du tableau, puis choisissez Afficher toutes les CloudWatch mesures. Dans la CloudWatch console, choisissez Logs pour voir le résultat du journal.

Étapes suivantes

Pour en savoir plus AWS SAM, consultez les ressources suivantes :

- AWS SAM exemples d'applications : le AWS SAM GitHub référentiel contient des exemples d'applications supplémentaires que vous pouvez télécharger et tester.
- Serverless Land : site qui rassemble des modèles sans serveur, des exemples de code et des ressources d'apprentissage spécifiques aux applications AWS SAM.

Traitez les événements Amazon S3 avec AWS SAM

Avec cet exemple d'application, vous mettez à profit ce que vous avez appris dans les exemples précédents et installez une application plus complexe. Cette application se compose d'une fonction Lambda appelée par une source d'événement de téléchargement d'objet Amazon S3. Cet exercice explique comment accéder aux AWS ressources et effectuer des appels de AWS service par le biais d'une fonction Lambda.

Cet exemple d'application sans serveur traite les événements de création d'objets dans Amazon S3. Pour chaque image téléchargée vers un compartiment, Amazon S3 détecte l'événement créé par l'objet et appelle une fonction Lambda. La fonction Lambda appelle Amazon Rekognition pour détecter le texte qui se trouve dans l'image. Elle stocke ensuite les résultats renvoyés par Amazon Rekognition dans une table DynamoDB.

> **Note**
>
> Avec cet exemple d'application, vous effectuez des étapes dans un ordre légèrement différent de celui des exemples précédents. La raison en est que cet exemple nécessite que AWS des ressources soient créées et que les autorisations IAM soient configurées avant de pouvoir tester la fonction Lambda localement. Nous allons en tirer parti CloudFormation pour créer les ressources et configurer les autorisations pour vous. Sinon, vous devrez le faire manuellement avant de pouvoir tester la fonction Lambda localement.

Étant donné que cet exemple est plus compliqué, assurez-vous d'être familiarisé avec l'installation des exemples d'applications précédents avant d'exécuter celui-ci.

Avant de commencer

Assurez-vous que vous avez terminé la configuration requise dans Installer la CLI AWS SAM.

Étape 1 : initialiser l'application

Dans cette section, vous allez télécharger l'exemple d'application, qui se compose d'un AWS SAM modèle et d'un code d'application.

Pour initialiser l'application

1. Exécutez la commande suivante à l'invite de commande de la CLI AWS SAM.

```
sam init \
--location https://github.com/aws-samples/cookiecutter-aws-sam-s3-rekognition-
dynamodb-python \
--no-input
```

2. Vérifiez le contenu du répertoire créé par la commande (`aws_sam_ocr/`) :

   - `template.yaml`— Définit trois AWS ressources dont l'application Amazon S3 a besoin : une fonction Lambda, un compartiment Amazon S3 et une table DynamoDB. Le modèle définit également les mappages et les autorisations entre ces ressources.
   - Répertoire `src/` – Contient le code d'application Amazon S3.
   - `SampleEvent.json` – L'exemple de source d'événement, qui est utilisé pour les tests locaux.

Étape 2 : empaqueter l'application

Avant de pouvoir tester cette application localement, vous devez utiliser le AWS SAMCLI pour créer un package de déploiement, que vous utiliserez pour déployer l'application AWS dans le cloud. Ce déploiement crée les AWS ressources et les autorisations nécessaires pour tester l'application localement.

Pour créer un package de déploiement Lambda

1. Créez un compartiment S3 à l'emplacement où vous souhaitez enregistrer le code empaqueté. Si vous souhaitez utiliser un compartiment S3 existant, ignorez cette étape.

```
aws s3 mb s3://bucketname
```

2. Créez le package de déploiement en exécutant la commande CLI `package` suivante.

```
sam package \
    --template-file template.yaml \
    --output-template-file packaged.yaml \
    --s3-bucket bucketname
```

   Vous spécifiez le nouveau fichier de modèle, `packaged.yaml`, lorsque vous déployez l'application dans l'étape suivante.

Étape 3 : déployer l'application

Maintenant que vous avez créé le package de déploiement, vous l'utilisez pour déployer l'application AWS dans le cloud. Vous testez ensuite l'application en l'invoquant dans le AWS Cloud.

Pour déployer l'application sans serveur dans le cloud AWS

- Dans le AWS SAMCLI, utilisez la `deploy` commande pour déployer toutes les ressources que vous avez définies dans le modèle.

```
sam deploy \
    --template-file packaged.yaml \
    --stack-name aws-sam-ocr \
    --capabilities CAPABILITY_IAM \
    --region us-east-1
```

  Dans la commande, le `--capabilities` paramètre permet CloudFormation de créer un rôle IAM.

  CloudFormation crée les AWS ressources définies dans le modèle. Vous pouvez accéder aux noms de ces ressources dans la CloudFormation console.

Pour tester l'application sans serveur dans le cloud AWS

1. Téléchargez une image dans le compartiment Amazon S3 que vous avez créé pour cet exemple d'application.
2. Ouvrez la console DynamoDB et recherchez la table qui a été créée. Voir la table pour les résultats renvoyés par Amazon Rekognition.
3. Vérifiez que la table DynamoDB contient de nouveaux enregistrements contenant du texte trouvé par Amazon Rekognition dans l'image téléchargée.

Étape 4 : tester l'application localement

Avant de pouvoir tester l'application localement, vous devez d'abord récupérer les noms des AWS ressources créées par CloudFormation.

- Récupérez le nom de la clé et le nom du compartiment Amazon S3 auprès de CloudFormation. Modifiez le fichier `SampleEvent.json` en remplaçant les valeurs de la clé d'objet, du nom du compartiment et de l'ARN du compartiment.
- Récupérez le nom de la table DynamoDB. Ce nom est utilisé pour la commande `sam local invoke` suivante.

Utilisez la CLI AWS SAM pour générer un exemple d'événement Amazon S3 et invoquer la fonction Lambda :

```
TABLE_NAME=Table name obtained from CloudFormation console sam local invoke --event
 SampleEvent.json
```

La portion `TABLE_NAME=` définit le nom de la table DynamoDB. Le paramètre `--event` spécifie le fichier contenant le message d'événement test à transmettre à la fonction Lambda.

Vous pouvez maintenant vérifier que les enregistrements DynamoDB attendus ont été créés, en fonction des résultats renvoyés par Amazon Rekognition.

Étapes suivantes

Pour en savoir plus AWS SAM, consultez les ressources suivantes :

- AWS SAM exemples d'applications : le AWS SAM GitHub référentiel contient des exemples d'applications supplémentaires que vous pouvez télécharger et tester.

- Serverless Land : site qui rassemble des modèles sans serveur, des exemples de code et des ressources d'apprentissage spécifiques aux applications AWS SAM.

Développez votre application sans serveur avec AWS SAM

Cette section contient des rubriques relatives à la validation de votre AWS SAM modèle et à la création de votre application avec des dépendances. Il contient également des rubriques relatives à l'utilisation AWS SAM dans certains cas d'utilisation tels que l'utilisation de couches Lambda, l'utilisation d'applications imbriquées, le contrôle de l'accès à API Gateway APIs, l'orchestration des ressources avec AWS Step Functions et la signature de code de vos applications. Les trois étapes principales que vous devez franchir pour développer votre application sont répertoriées ci-dessous.

Rubriques

- Créez votre application dans AWS SAM
- Définissez votre infrastructure avec AWS SAM
- Créez votre application avec AWS SAM

Créez votre application dans AWS SAM

Après avoir terminé Getting started et luComment utiliser AWS Serverless Application Model (AWS SAM), vous serez prêt à créer un AWS SAM projet dans votre environnement de développement. Votre AWS SAM projet servira de point de départ pour l'écriture de votre application sans serveur. Pour obtenir la liste des options de AWS SAMCLI `sam init` commande, consultezsam init.

La AWS Serverless Application Model commande Command Line Interface (AWS SAMCLI) `sam init` fournit des options permettant d'initialiser une nouvelle application sans serveur composée des éléments suivants :

- Un AWS SAM modèle pour définir votre code d'infrastructure.
- Une structure de dossiers qui organise votre application.
- Configuration de vos AWS Lambda fonctions.

Pour créer un AWS SAM projet, reportez-vous aux rubriques de ces sections.

> **Note**
>
> `sam init`inclut des modèles de projet pour les fonctions durables dans les environnements d'exécution pris en charge (PythonTypeScript, Java). Ces modèles fournissent le code

de démarrage et la configuration nécessaires à la création d'applications sans serveur dynamiques.

Rubriques

- Initialiser une nouvelle application sans serveur
- Options pour sam init
- Résolution des problèmes
- Exemples
- En savoir plus
- Étapes suivantes

Initialiser une nouvelle application sans serveur

Pour initialiser une nouvelle application sans serveur à l'aide de la CLI AWS SAM

1. `cd` vers un répertoire de départ.
2. Dans la ligne de commande, exécutez la commande suivante :

```
$ sam init
```

3. La CLI AWS SAM vous guidera à travers un flux interactif pour créer une nouvelle application sans serveur.

> **Note**
>
> Comme indiqué dansTutoriel : Déployer une application Hello World avec AWS SAM, cette commande initialise votre application sans serveur et crée le répertoire de votre projet. Ce répertoire contiendra plusieurs fichiers et dossiers. Le fichier le plus important est`template.yaml`. Il s'agit de votre AWS SAM modèle. Votre version de python doit correspondre à la version de python répertoriée dans le `template.yaml` fichier créé par la sam init commande.

Choisir un modèle de départ

Un modèle se compose des éléments suivants :

1. Un AWS SAM modèle pour votre code d'infrastructure.
2. Un répertoire de départ qui organise les fichiers de votre projet. Par exemple, cela peut inclure :
   a. Une structure pour votre code de fonction Lambda et leurs dépendances.
   b. Un dossier `events` qui contient les événements de test pour les tests locaux.
   c. Un dossier `tests` destiné à faciliter les tests d'unités.
   d. Un fichier `samconfig.toml` pour configurer les paramètres du projet.
   e. Un fichier `ReadMe` et d'autres fichiers de base du projet de départ.

   Voici un exemple de répertoire de projet de départ :

```
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
```

Vous pouvez choisir parmi une liste de modèles de démarrage rapide AWS disponibles ou fournir votre propre emplacement de modèle personnalisé.

Pour choisir un modèle de démarrage AWS rapide

1. Lorsque vous y êtes invité, sélectionnez modèles de démarrage rapide AWS .
2. Sélectionnez un modèle de démarrage AWS rapide pour commencer. Voici un exemple :

```
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
```

Pour choisir l'emplacement de votre modèle personnalisé

1. Lorsque vous y êtes invité, sélectionnez l'emplacement du modèle personnalisé.

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

2. La CLI AWS SAM vous invitera à indiquer l'emplacement du modèle.

```
Template location (git, mercurial, http(s), zip, path):
```

Indiquez l'un des emplacements suivants pour l'archive du fichier .zip de votre modèle :

- Référentiel GitHub : chemin d'accès au fichier .zip de votre référentiel GitHub. Le fichier doit se trouver à la racine de votre référentiel.
- Référentiel Mercurial : chemin d'accès au fichier .zip de votre référentiel Mercurial. Le fichier doit se trouver à la racine de votre référentiel.
- Chemin .zip : chemin HTTPS ou local vers votre fichier .zip.

3. La CLI AWS SAM initialisera votre application sans serveur à l'aide de votre modèle personnalisé.

Choisir une exécution

Lorsque vous choisissez un modèle de démarrage rapide AWS , la CLI AWS SAM vous invite à sélectionner une exécution pour vos fonctions Lambda. La liste des options affichées par la CLI AWS SAM correspond aux exécutions prises en charge nativement par Lambda.

- Le runtime fournit un environnement spécifique au langage qui s'exécute dans un environnement d'exécution.
- Lorsqu'il est déployé sur le AWS Cloud, le service Lambda appelle votre fonction dans un environnement d'exécution.

Vous pouvez utiliser n'importe quel autre langage de programmation doté d'une exécution personnalisée. Pour ce faire, vous devez créer manuellement la structure de votre application de départ. Vous pouvez ensuite utiliser `sam init` pour initialiser rapidement votre application en configurant un emplacement de modèle personnalisé.

À partir de votre sélection, la CLI AWS SAM crée le répertoire de départ pour votre code de fonction Lambda et vos dépendances.

Si Lambda prend en charge plusieurs gestionnaires de dépendances pour votre exécution, vous serez invité à choisir votre gestionnaire de dépendances préféré.

Choisir un type de package

Lorsque vous choisissez un modèle de démarrage rapide AWS , et une exécution laCLI AWS SAM vous invite à sélectionner un type de package. Le type de package détermine la manière dont vos fonctions Lambda sont déployées pour être utilisées avec le service Lambda. Les deux types de packages pris en charge sont les suivants :

1. Image du conteneur : contient le système d'exploitation de base, l'exécution, les extensions Lambda, le code de votre application et ses dépendances.
2. .zip file archive : contient le code de votre application et ses dépendances.

Pour en savoir plus sur le package de déploiement, consultez Packages de déploiement Lambda dans le Guide du développeur AWS Lambda .

Voici un exemple de structure de répertoires d'une application avec une fonction Lambda packagée sous forme d'image de conteneur. AWS SAMCLITélécharge l'image et en crée une `Dockerfile` dans le répertoire de la fonction pour spécifier l'image.

```
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
```

Voici un exemple de structure de répertoires d'une application avec une fonction packagée sous forme d'archive de fichier .zip.

```
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
```

Configuration du AWS X-Ray suivi

Vous pouvez choisir d'activer le AWS X-Ray suivi. Pour en savoir plus, consultez Qu'est-ce que c'est AWS X-Ray ? dans le Guide AWS X-Ray du développeur.

Si vous l'activez, votre AWS SAM modèle AWS SAMCLI est configuré. Voici un exemple :

```
Globals:
  Function:
    ...
    Tracing: Active
  Api:
    TracingEnabled: True
```

Configurer la surveillance avec Amazon CloudWatch Application Insights

Vous pouvez choisir d'activer la surveillance à l'aide d'Amazon CloudWatch Application Insights. Pour en savoir plus, consultez Amazon CloudWatch Application Insights dans le guide de CloudWatch l'utilisateur Amazon.

Si vous l'activez, votre AWS SAM modèle AWS SAMCLI est configuré. Voici un exemple :

```
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
```

Nommer votre application

Saisissez un nom pour votre application. La CLI AWS SAM crée un dossier de niveau supérieur pour votre application sous ce nom.

Options pour sam init

Voici quelques-unes des principales options que vous pouvez utiliser avec la commande `sam init`. Pour obtenir la liste de toutes les options, consultez sam init.

Initialiser une application à l'aide d'un emplacement de modèle personnalisé

Utilisez l'option `--location` et indiquez un emplacement de modèle personnalisé pris en charge. Voici un exemple :

```
$ sam init --location https://github.com/aws-samples/sessions-with-aws-sam/raw/master/
starter-templates/web-app.zip
```

Initialiser une application sans le flux interactif

Utilisez l'option `--no-interactive` et indiquez vos choix de configuration sur la ligne de commande pour ignorer le flux interactif. Voici un exemple :

```
$ sam init --no-interactive --runtime go1.x --name go-demo --dependency-manager mod --
app-template hello-world
```

Résolution des problèmes

Pour résoudre le problème AWS SAMCLI, voirRésolution des problèmes de la CLI AWS SAM.

Exemples

Initialiser une nouvelle application sans serveur à l'aide du modèle de démarrage Hello World AWS

Pour cet exemple, consultez Étape 1 : initialisation de l'exemple d'application Hello World dans le didacticiel : déploiement de l'application Hello World.

Initialiser une nouvelle application sans serveur avec un emplacement de modèle personnalisé

Vous trouverez ci-dessous des exemples d'ajout d'un emplacement GitHub à votre modèle personnalisé :

```
$ 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
```

Voici un exemple de chemin d'accès à un fichier local :

```
$ sam init --location /path/to/template.zip
```

Voici un exemple de chemin accessible par HTTPS :

```
$ sam init --location https://github.com/aws-samples/sessions-with-aws-sam/raw/master/
starter-templates/web-app.zip
```

En savoir plus

Pour en savoir plus sur l'utilisation de la commande `sam init`, reportez-vous à ce qui suit :

- Apprentissage AWS SAM : sam init — Série Serverless Land « Learning AWS SAM » surYouTube.
- Structurer des applications sans serveur pour les utiliser avec la CLI AWS SAM(sessions avec SAM S2E7) – sessions avec les séries AWS SAM sur YouTube.

Étapes suivantes

Maintenant que vous avez créé votre AWS SAM projet, vous êtes prêt à créer votre application. Consultez Définissez votre infrastructure avec AWS SAM les instructions détaillées sur les tâches que vous devez effectuer pour ce faire.

Définissez votre infrastructure avec AWS SAM

Maintenant que vous avez créé votre projet, vous êtes prêt à définir votre infrastructure d'applications avec AWS SAM. Pour ce faire, configurez votre AWS SAM modèle pour définir les ressources et les propriétés de votre application, c'est-à-dire le `template.yaml` fichier de votre AWS SAM projet.

Les rubriques de cette section fournissent du contenu sur la définition de votre infrastructure dans votre AWS SAM modèle (votre `template.yaml` fichier). Il contient également des rubriques sur la définition de ressources pour des cas d'utilisation spécifiques, tels que l'utilisation de couches Lambda, l'utilisation d'applications imbriquées, le contrôle de l'accès à API Gateway APIs, l'orchestration des ressources avec AWS Step Functions, la signature de code de vos applications et la validation de votre modèle. AWS SAM

Rubriques

- Définissez les ressources de l'application dans votre AWS SAM modèle
- Configuration et gestion de l'accès aux ressources dans votre AWS SAM modèle
- Contrôlez l'accès aux API avec votre AWS SAM modèle
- Améliorez l'efficacité en utilisant les couches Lambda avec AWS SAM
- Réutilisez le code et les ressources à l'aide d'applications imbriquées dans AWS SAM
- Gérez les événements temporels avec le EventBridge planificateur dans AWS SAM
- Orchestrer les AWS SAM ressources avec AWS Step Functions
- Configurer la signature de code pour votre AWS SAM application
- Valider les fichiers AWS SAM modèles

Définissez les ressources de l'application dans votre AWS SAM modèle

Vous définissez les AWS ressources utilisées par votre application sans serveur dans la `Resources` section de votre AWS SAM modèle. Lorsque vous définissez une ressource, vous identifiez ce qu'elle est, comment elle interagit avec les autres ressources et comment elle est accessible (c'est-à-dire les autorisations de la ressource).

La `Resources` section de votre AWS SAM modèle peut contenir une combinaison de CloudFormation ressources et de AWS SAM ressources. En outre, vous pouvez utiliser AWS SAM la syntaxe abrégée pour les ressources suivantes :

AWS SAM syntaxe abrégée	Ce qu'il fait avec une AWS ressource connexe
AWS::Serverless::Api	Crée une collection de ressources et de méthodes API Gateway qui peuvent être invoquées via des points de terminaison HTTPS.
AWS::Serverless::Application	Intègre une application sans serveur à partir du AWS Serverless Application Repository ou à partir d'un compartiment Amazon S3 comme application imbriquée.
AWS::Serverless::Connector	Configure les autorisations entre deux ressources. Pour obtenir une présentation des connecteurs, veuillez consulter Gestion des autorisations relatives aux ressources à l'aide de AWS SAM connecteurs.
AWS::Serverless::Function	Crée une AWS Lambda fonction, un rôle d'exécution Gestion des identités et des accès AWS (IAM) et des mappages de sources d'événements qui déclenchent la fonction.
AWS::Serverless::GraphQLApi	crée et configure une AWS AppSync GraphQL API pour votre application sans serveur.
AWS::Serverless::HttpApi	Crée une API HTTP Amazon API Gateway, qui vous permet de créer RESTful APIs avec une latence et des coûts inférieurs à ceux de REST APIs.
AWS::Serverless::LayerVersion	Crée un Lambda LayerVersion qui contient la bibliothèque ou le code d'exécution nécessaire à une fonction Lambda.

AWS SAM syntaxe abrégée	Ce qu'il fait avec une AWS ressource connexe
AWS::Serverless::SimpleTable	Crée une table DynamoDB avec une clé primaire d'attribut unique.
AWS::Serverless::StateMachine	Crée une machine à AWS Step Functions états, que vous pouvez utiliser pour orchestre r des AWS Lambda fonctions et d'autres AWS ressources afin de créer des flux de travail complexes et robustes.

Les ressources ci-dessus sont également répertoriées dansAWS SAM ressources et propriétés.

Pour obtenir des informations de référence sur tous les types de AWS ressources et de propriétés CloudFormation ainsi que sur le AWS SAM support, voir la référence aux types de AWS ressources et de propriétés dans le guide de AWS CloudFormation l'utilisateur.

Configuration et gestion de l'accès aux ressources dans votre AWS SAM modèle

Pour que vos AWS ressources puissent interagir les unes avec les autres, l'accès et les autorisations appropriés doivent être configurés entre vos ressources. Pour ce faire, vous devez configurer les utilisateurs, les rôles et les politiques Gestion des identités et des accès AWS (IAM) pour effectuer votre interaction de manière sécurisée.

Les rubriques de cette section concernent toutes la configuration de l'accès aux ressources définies dans votre modèle. Cette section commence par les meilleures pratiques générales. Les deux rubriques suivantes passent en revue deux options dont vous disposez pour configurer l'accès et les autorisations entre les ressources référencées dans votre application sans serveur : les AWS SAM connecteurs et les modèles de AWS SAM politique. La dernière rubrique fournit des détails sur la gestion de l'accès des utilisateurs à l'aide des mêmes mécanismes que CloudFormation ceux utilisés pour gérer les utilisateurs.

Pour en savoir plus, veuillez consulter la rubrique Contrôle de l'accès avec Gestion des identités et des accès AWS dans le Guide de l'utilisateur AWS CloudFormation .

Le AWS Serverless Application Model (AWS SAM) propose deux options qui simplifient la gestion des accès et des autorisations pour vos applications sans serveur.

1. AWS SAM connecteurs
2. Modèles de politique AWS SAM

AWS SAM connecteurs

Les connecteurs permettent de fournir des autorisations entre deux ressources. Pour ce faire, décrivez comment ils doivent interagir les uns avec les autres dans votre AWS SAM modèle. Ils peuvent être définis à l'aide de l'attribut de ressource `Connectors` ou du type de ressource `AWS::Serverless::Connector`. Les connecteurs prennent en charge le provisionnement `Read` et l'`Write`accès aux données et aux événements entre une combinaison de AWS ressources. Pour en savoir plus sur les AWS SAM connecteurs, voirGestion des autorisations relatives aux ressources à l'aide de AWS SAM connecteurs.

Modèles de politique AWS SAM

AWS SAM les modèles de politique sont des ensembles prédéfinis d'autorisations que vous pouvez ajouter à vos AWS SAM modèles pour gérer l'accès et les autorisations entre vos AWS Lambda fonctions, vos machines AWS Step Functions d'état et les ressources avec lesquelles elles interagissent. Pour en savoir plus sur les modèles de AWS SAM politiques, consultezModèles de politique AWS SAM.

AWS CloudFormation mécanismes

CloudFormation les mécanismes incluent la configuration des utilisateurs, des rôles et des politiques IAM pour gérer les autorisations entre vos AWS ressources. Pour en savoir plus, veuillez consulter la section Gestion des AWS SAM autorisations à l'aide de CloudFormation mécanismes.

Bonnes pratiques

Dans l'ensemble de vos applications sans serveur, vous pouvez utiliser plusieurs méthodes pour configurer les autorisations entre vos ressources. Vous pouvez donc sélectionner la meilleure option pour chaque scénario et utiliser plusieurs options ensemble dans vos applications. Voici quelques éléments à prendre en compte lors du choix de l'option la mieux adaptée à votre cas :

- AWS SAM les connecteurs et les modèles de politique réduisent tous deux l'expertise IAM requise pour faciliter les interactions sécurisées entre vos AWS ressources. Utilisez des connecteurs et des modèles de politique lorsque cela est possible.
- AWS SAM les connecteurs fournissent une syntaxe abrégée simple et intuitive pour définir les autorisations dans vos AWS SAM modèles et nécessitent le moins d'expertise en matière d'IAM.

Lorsque les AWS SAM connecteurs et les modèles de politique sont pris en charge, utilisez des AWS SAM connecteurs.

- AWS SAM les connecteurs peuvent fournir des données `Read` et des événements et `Write` y accéder entre les ressources AWS SAM source et de destination prises en charge. Pour afficher la liste des ressources prises en charge, consultez AWS SAM référence du connecteur. Lorsque cela est pris en charge, utilisez AWS SAM des connecteurs.
- Bien que les modèles de AWS SAM politique soient limités aux autorisations entre vos fonctions Lambda, les machines d'état Step Functions et les AWS ressources avec lesquelles elles interagissent, les modèles de politique prennent en charge toutes les opérations CRUD. Lorsque cela est pris en charge et qu'un modèle de AWS SAM stratégie adapté à votre scénario est disponible, utilisez des modèles AWS SAM de stratégie. Pour afficher la liste des modèles de politique disponibles, consultez Modèles de politique AWS SAM.
- Pour tous les autres scénarios, ou lorsque la granularité est requise, utilisez CloudFormation des mécanismes.

Gestion des autorisations relatives aux ressources à l'aide de AWS SAM connecteurs

Les connecteurs sont un AWS Serverless Application Model (AWS SAM) type de ressource abstrait, identifié comme tel`AWS::Serverless::Connector`, qui fournit des autorisations simples et bien définies entre les ressources de vos applications sans serveur.

Avantages des AWS SAM connecteurs

En composant automatiquement les politiques d'accès appropriées entre les ressources, les connecteurs vous permettent de créer vos applications sans serveur et de vous concentrer sur l'architecture de votre application sans avoir besoin d'expertise en matière de fonctionnalités d' AWS autorisation, de langage de politique et de paramètres de sécurité spécifiques aux services. Par conséquent, les connecteurs constituent un avantage considérable pour les développeurs novices en matière de développement sans serveur, mais également pour les développeurs chevronnés qui souhaitent augmenter leur vitesse de développement.

Utilisation de AWS SAM connecteurs

Utilisez l'attribut de ressource `Connectors` en l'incorporant dans une ressource source. Définissez ensuite votre ressource de destination et décrivez comment les données ou les événements doivent circuler entre ces ressources. AWS SAM compose ensuite les politiques d'accès nécessaires pour faciliter les interactions requises.

Voici un aperçu de la façon dont cet attribut de ressource est écrit :

```
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>
  ...
```

Fonctionnement des connecteurs

> **Note**
>
> Cette section explique comment les connecteurs fournissent les ressources nécessaires en arrière-plan. Cela se produit automatiquement lorsque vous utilisez des connecteurs.

Tout d'abord, l'attribut de ressource `Connectors` intégré est transformé en un type de ressource `AWS::Serverless::Connector`. Son identifiant logique est automatiquement créé en tant que*<source-resource-logical-id><embedded-connector-logical-id>*.

Par exemple, voici un connecteur intégré :

```
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
```

Cela générera la ressource `AWS::Serverless::Connector` suivante :

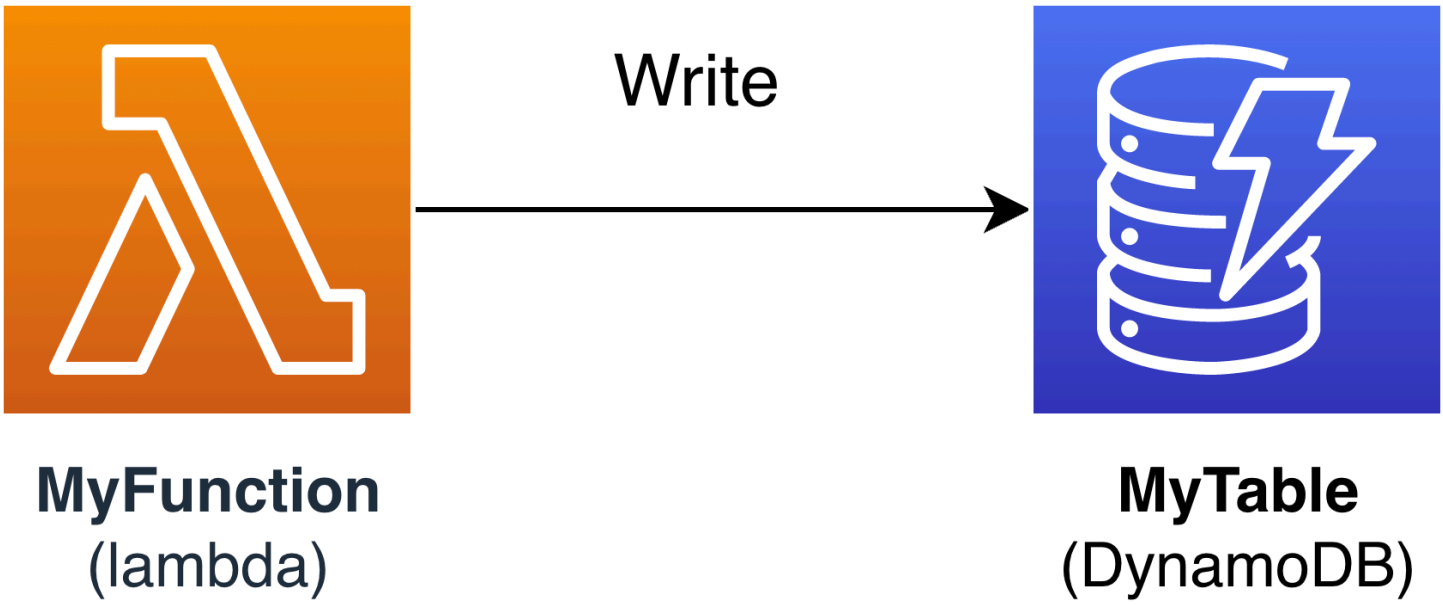
```
Transform: AWS::Serverless-2016-10-31
Resources:
  ...
  MyFunctionMyConn:
    Type: AWS::Serverless::Connector
    Properties:
      Source:
        Id: MyFunction
      Destination:
        Id: MyTable
      Permissions:
        - Read
        - Write
```

> **Note**
>
> Vous pouvez également définir des connecteurs dans votre AWS SAM modèle à l'aide de cette syntaxe. Cela est recommandé lorsque votre ressource source est définie sur un modèle distinct de votre connecteur.

Ensuite, les politiques d'accès nécessaires à cette connexion sont automatiquement élaborées. Pour plus d'informations sur les ressources générées par les connecteurs, consultez CloudFormation ressources générées lorsque vous spécifiez AWS::Serverless::Connector.

Exemple de connecteurs

L'exemple suivant montre comment utiliser des connecteurs pour écrire des données d'une AWS Lambda fonction dans une table Amazon DynamoDB.




```
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
```

L'attribut de ressource `Connectors` est intégré à la ressource source de la fonction Lambda. La table DynamoDB est définie comme la ressource de destination à l'aide de la propriété `Id`. Les connecteurs fourniront des autorisations `Write` entre ces deux ressources.

Lorsque vous déployez votre AWS SAM modèle sur CloudFormation, il compose AWS SAM automatiquement les politiques d'accès nécessaires au bon fonctionnement de cette connexion.

Connexions prises en charge entre les ressources source et de destination

Les connecteurs prennent en charge les types d'autorisations `Read` et `Write` pour les données et les événements entre une combinaison choisie de connexions de ressources source et destination. Par exemple, ils prennent en charge une connexion `Write` entre une ressource source `AWS::ApiGateway::RestApi` et une ressource destination `AWS::Lambda::Function`.

Les ressources source et de destination peuvent être définies à l'aide d'une combinaison de propriétés prises en charge. Les exigences en matière de propriété dépendent de la connexion que vous établissez et de l'endroit où les ressources sont définies.

> **Note**
>
> Les connecteurs peuvent attribuer des autorisations entre les types de ressources sans serveur et non sans serveur pris en charge.

Pour obtenir la liste des connexions aux ressources prises en charge et de leurs exigences en matière de propriétés, consultez Types de ressources source et de destination pris en charge pour les connecteurs.

Définissez les autorisations de lecture et d'écriture dans AWS SAM

Dans AWS SAM, `Read` et `Write` les autorisations peuvent être allouées au sein d'un seul connecteur :

```
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
```

Pour plus d'informations sur l'utilisation des connecteurs, reportez-vous àAWS SAM référence du connecteur.

Définissez les ressources à l'aide d'autres propriétés prises en charge dans AWS SAM

Pour les ressources source et de destination, lorsqu'elles sont définies dans le même modèle, utilisez la propriété `Id`. Sinon, un élément `Qualifier` peut être ajouté pour réduire la portée de la ressource que vous avez définie. Quand les ressources ne se trouvent pas dans le même modèle, utilisez une combinaison d'autres propriétés prises en charge.

- Pour obtenir la liste des combinaisons de propriétés prises en charge pour les ressources source et de destination, consultez Types de ressources source et de destination pris en charge pour les connecteurs.
- Pour obtenir une description des propriétés que vous pouvez utiliser avec les connecteurs, reportez-vous à la section AWS::Serverless::Connector.

Lorsque vous définissez une ressource source avec une propriété autre que la propriété `Id`, utilisez la propriété `SourceReference`.

```
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>
```

Voici un exemple d'utilisation d'un élément `Qualifier` pour réduire la portée d'une ressource Amazon API Gateway :

```
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
  ...
```

Voici un exemple utilisant une combinaison prise en charge d'éléments `Arn` et `Type` pour définir une ressource de destination à partir d'un autre modèle :

```
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
  ...
```

Pour plus d'informations sur l'utilisation des connecteurs, reportez-vous à[AWS SAM référence du connecteur](#).

Créez plusieurs connecteurs à partir d'une seule source dans AWS SAM

Au sein d'une ressource source, vous pouvez définir plusieurs connecteurs, chacun ayant une ressource de destination différente.

```
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
...
```

Pour plus d'informations sur l'utilisation des connecteurs, reportez-vous àAWS SAM référence du connecteur.

Créez des connecteurs à destinations multiples dans AWS SAM

Au sein d'une ressource source, vous pouvez définir un connecteur unique avec plusieurs ressources de destination. Voici un exemple de ressource source d'une fonction Lambda connectée à un compartiment Amazon Simple Storage Service (Amazon S3) et à une table DynamoDB :

```
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
```

Pour plus d'informations sur l'utilisation des connecteurs, reportez-vous àAWS SAM référence du connecteur.

Définissez les attributs des ressources à l'aide de connecteurs dans AWS SAM

Les attributs de ressource peuvent être définis pour les ressources afin de spécifier des comportements et des relations supplémentaires. Pour plus d'informations sur les attributs des ressources, consultez Référence d'attribut de ressource dans le Guide de l'utilisateur AWS CloudFormation .

Vous pouvez ajouter des attributs de ressource à votre connecteur intégré en les définissant au même niveau que les propriétés de votre connecteur. Lorsque votre AWS SAM modèle est transformé lors du déploiement, les attributs sont transmis aux ressources générées.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Connectors:
      MyConn:
        DeletionPolicy: Retain
        DependsOn: AnotherFunction
        Properties:
          ...
```

Pour plus d'informations sur l'utilisation des connecteurs, reportez-vous àAWS SAM référence du connecteur.

En savoir plus

Pour plus d'informations sur l'utilisation des AWS SAM connecteurs, consultez les rubriques suivantes :

- AWS::Serverless::Connector
- Définissez les autorisations de lecture et d'écriture dans AWS SAM
- Définissez les ressources à l'aide d'autres propriétés prises en charge dans AWS SAM
- Créez plusieurs connecteurs à partir d'une seule source dans AWS SAM
- Créez des connecteurs à destinations multiples dans AWS SAM
- Définissez les autorisations de lecture et d'écriture dans AWS SAM
- Définissez les attributs des ressources à l'aide de connecteurs dans AWS SAM

Faire un commentaire

Pour fournir des commentaires sur les connecteurs, soumettez un nouveau problème dans le serverless-application-model AWS GitHubréférentiel.

Modèles de politique AWS SAM

Le AWS Serverless Application Model (AWS SAM) vous permet de choisir parmi une liste de modèles de politiques pour étendre les autorisations de vos fonctions Lambda et de vos machines d' AWS Step Functions état aux ressources utilisées par votre application.

AWS SAM les applications du AWS Serverless Application Repository qui utilisent des modèles de politique ne nécessitent aucun accusé de réception spécial du client pour déployer l'application depuis le. AWS Serverless Application Repository

Si vous souhaitez demander un nouveau modèle de stratégie à ajouter, procédez comme suit :

1. Soumettez une pull request sur le fichier source policy_templates.json dans la branche du `develop` projet. AWS SAM GitHub Vous pouvez trouver le fichier source dans policy_templates.json sur le site Web. GitHub
2. Soumettez un problème dans le AWS SAM GitHub projet qui inclut les raisons de votre pull request et un lien vers la demande. Utilisez ce lien pour soumettre un nouveau problème : AWS Serverless Application Model : Problèmes.

Syntaxe

Pour chaque modèle de stratégie que vous spécifiez dans votre fichier de AWS SAM modèle, vous devez toujours spécifier un objet contenant les valeurs d'espace réservé du modèle de stratégie. Si un modèle de stratégie ne nécessite aucune valeur d'espace réservé, vous devez spécifier un objet vide.

YAML

```
MyFunction:
  Type: AWS::Serverless::Function
  Properties:
    Policies:
      - PolicyTemplateName1:        # Policy template with placeholder value
          Key1: Value1
      - PolicyTemplateName2: {}     # Policy template with no placeholder value
```

> **Note**
>
> Si vous avez configuré une politique IAM normale ou si vous avez géré des politiques via Lambda, le modèle de stratégie peut être défini sans utiliser d'objet vide.

Exemples

Exemple 1 : Modèle de stratégie avec des valeurs d'espace réservé

L'exemple suivant montre que le modèle de stratégie SQSPollerPolicy attend un `QueueName` comme ressource. Le AWS SAM modèle récupère le nom de la file « `MyQueue` » Amazon SQS, que vous pouvez créer dans la même application ou demander en tant que paramètre de l'application.

```
MyFunction:
  Type: 'AWS::Serverless::Function'
  Properties:
    CodeUri: ${codeuri}
    Handler: hello.handler
    Runtime: python2.7
    Policies:
      - SQSPollerPolicy:
          QueueName:
            !GetAtt MyQueue.QueueName
```

Exemple 2 : modèle de stratégie sans valeurs d'espace réservé

L'exemple suivant contient le modèle de stratégieCloudWatchPutMetricPolicy, qui n'a pas de valeurs d'espace réservé.

> **Note**
>
> Même s'il n'y a pas de valeurs d'espace réservé, vous devez spécifier un objet vide, sinon une erreur se produira.

```
MyFunction:
  Type: 'AWS::Serverless::Function'
  Properties:
    CodeUri: ${codeuri}
```

```
    Handler: hello.handler
    Runtime: python2.7
    Policies:
      - CloudWatchPutMetricPolicy: {}
```

Exemple 3 : modèle de politique avec des valeurs d'espace réservé et une politique IAM normale

L'exemple suivant contient la politique et DynamoDBCrudPolicy le modèle de politique Amazon SQSFull Access. La politique Amazon SQSFull Access est une politique IAM et non une AWS SAM politique. Il n'est donc pas nécessaire de spécifier un objet vide car la politique serait directement transmise au CloudFormation.

```
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
```

Tableau de modèle de stratégie

Voici un tableau des modèles de stratégie disponibles.

Modèle de stratégie	Description		
AcmGetCertificatePolicy	Donne l'autorisation de lire un certificat à partir de AWS Certificate Manager.		
AMIDescribePolicy	Donne l'autorisation de décrire Amazon Machine Images (AMIs).		
AthenaQueryPolicy	Donne des autorisations pour exécuter des requêtes Athena.		

Modèle de stratégie	Description		
AWSSecretsManagerGetSecretValuePolicy	Donne l'autorisation d'obtenir la valeur secrète pour le secret AWS Secrets Manager spécifié.		
AWSSecretsManagerRotationPolicy	Donne l'autorisation de faire pivoter un secret dans AWS Secrets Manager.		
CloudFormationDescribeStacksPolicy	Donne l'autorisation de décrire les CloudFormation piles.		
CloudWatchDashboardPolicy	Permet de placer des métriques à appliquer sur les CloudWatch tableaux de bord.		
CloudWatchDescribeAlarmHistoryPolicy	Permet de décrire l'historique des CloudWatch alarmes.		
CloudWatchPutMetricPolicy	Donne l'autorisation d'envoyer des métriques à CloudWatch.		
CodeCommitCrudPolicy	Donne des autorisations aux create/read/update/delete objets d'un CodeCommit référentiel spécifique.		
CodeCommitReadPolicy	Permet de lire des objets au sein d'un CodeCommit référentiel spécifique.		

Modèle de stratégie	Description
CodePipelineLambdaExecutionPolicy	Permet à une fonction Lambda invoquée par CodePipeline de signaler l'état de la tâche.
CodePipelineReadOnlyPolicy	Donne l'autorisation de lecture pour obtenir des informations sur un CodePipeline pipeline.
ComprehendBasicAccessPolicy	Donne l'autorisation de détecter des entités, des expressions clés, des langues et des sentiments.
CostExplorerReadOnlyPolicy	Donne une autorisation en lecture seule à Cost APIs Explorer en lecture seule pour l'historique de facturation.
DynamoDBBackupFullAccessPolicy	Donne l'autorisation de lecture/écriture aux sauvegardes DynamoDB à la demande pour une table.
DynamoDBCrudPolicy	Donne les autorisations de créer, lire, mettre à jour, supprimer à une table Amazon DynamoDB.
DynamoDBReadPolicy	Donne l'autorisation de lecture seule à une table DynamoDB.
DynamoDBReconfigurePolicy	Donne l'autorisation de reconfigurer une table DynamoDB.
DynamoDBRestoreFromBackupPolicy	Donne l'autorisation de restaurer une table DynamoDB à partir d'une sauvegarde.

Modèle de stratégie	Description		
DynamoDBStreamReadPolicy	Donne l'autorisation de décrire et de lire les flux DynamoDB Streams ainsi que les enregistrements.		
DynamoDBWritePolicy	Donne l'autorisation d'écriture seule à une table DynamoDB.		
EC2CopyImagePolicy	Donne l'autorisation de copier les images Amazon EC2.		
EC2DescribePolicy	Donne l'autorisation de décrire des instances Amazon Elastic Compute Cloud (Amazon EC2).		
EcsRunTaskPolicy	Donne l'autorisation de démarrer une nouvelle tâche pour une définition de tâche.		
EFSWriteAccessPolicy	Donne l'autorisation de monter un système de fichiers Amazon EFS avec accès en écriture.		
EKSDescribePolicy	Donne l'autorisation de décrire ou de répertorier les clusters Amazon EKS.		
ElasticMapReduceAddJobFlowStepsPolicy	Donne l'autorisation d'ajouter de nouvelles étapes à un cluster en cours d'exécution.		
ElasticMapReduceCancelStepsPolicy	Donne l'autorisation d'annuler une étape ou des étapes en attente dans un cluster en cours d'exécution.		

Modèle de stratégie	Description		
ElasticMapReduceModifyInstanceFleetPolicy	Donne l'autorisation de répertorier les détails et de modifier les capacités des flottes d'instances au sein d'un cluster.		
ElasticMapReduceModifyInstanceGroupsPolicy	Donne l'autorisation de répertorier les détails et de modifier les réglages des groupes d'instances au sein d'un cluster.		
ElasticMapReduceSetTerminationProtectionPolicy	Donne l'autorisation de définir une protection contre l'arrêt pour un cluster.		
ElasticMapReduceTerminateJobFlowsPolicy	Donne l'autorisation d'arrêter un cluster.		
ElasticsearchHttpPostPolicy	Donne l'autorisation POST à Amazon OpenSearch Service.		
EventBridgePutEventsPolicy	Donne l'autorisation d'envoyer des événements à EventBridge.		
FilterLogEventsPolicy	Donne l'autorisation de filtrer CloudWatch les événements des journaux à partir d'un groupe de journaux spécifié.		

Modèle de stratégie	Description		
FirehoseCrudPolicy	Donne l'autorisation de créer, d'écrire, de mettre à jour et de supprimer un flux de diffusion Firehose.		
FirehoseWritePolicy	Donne l'autorisation d'écrire dans un flux de diffusion Firehose.		
KinesisCrudPolicy	Donne l'autorisation de créer, publier et supprimer un flux Amazon Kinesis.		
KinesisStreamReadPolicy	Donne l'autorisation d'afficher et de lire un flux Amazon Kinesis.		
KMSDecryptPolicy	Donne l'autorisation de déchiffrer avec une clé AWS Key Management Service (AWS KMS).		
KMSEncryptPolicy	Donne l'autorisation de chiffrer avec une clé AWS Key Management Service (AWS KMS).		
LambdaInvokePolicy	Donne l'autorisation d'invoquer une AWS Lambda fonction, un alias ou une version.		
MobileAnalyticsWriteOnlyAccessPolicy	Donne uniquement l'autorisation d'écriture pour placer les données d'événements pour toutes les ressources d'application.		
OrganizationsListAccountsPolicy	Donne l'autorisation en lecture seule de répertorier les noms de comptes enfants et. IDs		

Modèle de stratégie	Description
PinpointEndpointAccessPolicy	Donne l'autorisation d'obtenir et de mettre à jour les points de terminaison pour une application Amazon Pinpoint.
PollyFullAccessPolicy	Donne l'autorisation d'accès complet aux ressources de lexique Amazon Polly.
RekognitionDetectOnlyPolicy	Donne l'autorisation de comparer et de détecter les visages et les étiquettes.
RekognitionFacesManagementPolicy	Donne l'autorisation d'ajouter, de supprimer et de rechercher des visages dans une collection Amazon Rekognition.
RekognitionFacesPolicy	Donne l'autorisation de comparer et de détecter les visages et les étiquettes.
RekognitionLabelsPolicy	Donne l'autorisation de détecter les étiquettes d'objet et de modération.
RekognitionNoDataAccessPolicy	Donne l'autorisation de comparer et de détecter les visages et les étiquettes.
RekognitionReadPolicy	Donne l'autorisation de lister et de rechercher des visages.
RekognitionWriteOnlyAccessPolicy	Donne l'autorisation de créer des collections et d'indexer des visages.

Modèle de stratégie	Description		
Route53ChangeResourceRecordSetsPolicy	Donne l'autorisation de modifier des jeux d'enregistrements de ressources dans Route 53.		
S3CrudPolicy	Donne l'autorisation de créer, lire, mettre à jour et supprimer pour agir sur les objets d'un compartiment Amazon S3.		
S3FullAccessPolicy	Donne l'autorisation d'accès complet pour agir sur les objets d'un compartiment Amazon S3.		
S3ReadPolicy	Donne l'autorisation de lire des objets en lecture seule dans un compartiment Amazon Simple Storage Service (Amazon S3).		
S3WritePolicy	Donne l'autorisation d'écrire des objets dans un compartiment Amazon S3.		
SageMakerCreateEndpointConfigPolicy	Donne l'autorisation de créer une configuration de point de terminaison dans SageMaker AI.		
SageMakerCreateEndpointPolicy	Donne l'autorisation de créer un point de terminaison dans SageMaker AI.		
ServerlessRepoReadWriteAccessPolicy	Donne l'autorisation de créer et de répertorier des applications dans le AWS Serverless Application Repository service.		

Modèle de stratégie	Description		
SESBulkTemplatedCrudPolicy	Donne l'autorisation d'envoyer des e-mails, des e-mails modélisés, des e-mails groupés et de vérifier l'identité.		
SESBulkTemplatedCrudPolicy_v2	Donne l'autorisation d'envoyer des e-mails Amazon SES, des e-mails modélisés, des e-mails groupés et de vérifier l'identité.		
SESCrudPolicy	Donne l'autorisation d'envoyer un e-mail et de vérifier l'identité.		
SESEmailTemplateCrudPolicy	Donne l'autorisation de créer, obtenir, répertorier, mettre à jour et supprimer les modèles d'e-mail Amazon SES.		
SESSendBouncePolicy	Donne SendBounce l'autorisation d'utiliser une identité Amazon Simple Email Service (Amazon SES).		
SNSCrudPolicy	Donne l'autorisation de créer et de publier des rubriques Amazon SNS, et de s'y abonner.		
SNSPublishMessagePolicy	Donne l'autorisation de publier un message dans une rubrique Amazon Simple Notification Service (Amazon SNS).		
SQSPollerPolicy	Donne l'autorisation d'interroger une file d'attente Amazon Simple Queue Service (Amazon SQS).		
SQSSendMessagePolicy	Donne l'autorisation d'envoyer un message à une file d'attente Amazon SQS.		

Modèle de stratégie	Description		
SSMParameterReadPolicy	Donne l'autorisation d'accéder à un paramètre à partir d'un magasin de paramètres Amazon EC2 Systems Manager (SSM) pour charger des secrets dans ce compte. À utiliser lorsque le nom du paramètre n'est pas précédé d'une barre oblique.		
SSMParameterWithSlashPrefixReadPolicy	Donne l'autorisation d'accéder à un paramètre à partir d'un magasin de paramètres Amazon EC2 Systems Manager (SSM) pour charger des secrets dans ce compte. À utiliser lorsque le nom du paramètre est précédé d'une barre oblique.		
StepFunctionsExecutionPolicy	Donne l'autorisation de lancer l'exécution d'une machine d'état Step Functions.		
TextractDetectAnalyzePolicy	Donne l'accès permettant de détecter et d'analyser des documents avec Amazon Textract.		
TextractGetResultPolicy	Donne l'accès permettant d'obtenir des documents détectés et analysés avec Amazon Textract.		
TextractPolicy	Donne l'accès complet à Amazon Textract.		
VPCAccessPolicy	Donne l'accès pour créer, supprimer, décrire et détacher des interfaces réseau élastiques.		

Résolution des problèmes

Erreur SAM CLI : « Vous devez spécifier des valeurs de paramètres valides pour le modèle de politique '< policy-template-name >' »

Lors de l'exécution de `sam build`, l'erreur suivante s'affiche :

```
"Must specify valid parameter values for policy template '<policy-template-name>'"
```

Cela signifie que vous n'avez pas transmis d'objet vide lors de la déclaration d'un modèle de stratégie qui ne contient aucune valeur d'espace réservé.

Pour résoudre ce problème, déclarez la stratégie comme l'exemple suivant pour CloudWatchPutMetricPolicy.

```
MyFunction:
  Policies:
    - CloudWatchPutMetricPolicy: {}
```

AWS SAM liste de modèles de politiques

Vous trouverez ci-dessous les modèles de politiques disponibles, ainsi que les autorisations appliquées à chacun d'entre eux. AWS Serverless Application Model (AWS SAM) renseigne automatiquement les éléments réservés (tels que AWS la région et le numéro de compte) avec les informations appropriées.

Rubriques

- 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

Donne l'autorisation de lire un certificat à partir de AWS Certificate Manager.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "acm:GetCertificate"
    ],
    "Resource": {
      "Fn::Sub": [
        "${certificateArn}",
        {
          "certificateArn": {
            "Ref": "CertificateArn"
          }
        }
      ]
    }
  }
]
```

AMIDescribePolicy

Donne l'autorisation de décrire Amazon Machine Images (AMIs).

```
"Statement": [
  {
```

```
    "Effect": "Allow",
    "Action": [
      "ec2:DescribeImages"
    ],
    "Resource": "*"
  }
]
```

AthenaQueryPolicy

Donne des autorisations pour exécuter des requêtes Athena.

```
"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

Donne l'autorisation d'obtenir la valeur secrète pour le AWS Secrets Manager secret spécifié.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "secretsmanager:GetSecretValue"
    ],
    "Resource": {
      "Fn::Sub": [
        "${secretArn}",
        {
          "secretArn": {
            "Ref": "SecretArn"
          }
        }
      ]
    }
  }
]
```

AWSSecretsManagerRotationPolicy

Donne l'autorisation de faire pivoter un secret dans 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

Donne l'autorisation de décrire les CloudFormation piles.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
```

```
      "cloudformation:DescribeStacks"
    ],
    "Resource": {
      "Fn::Sub": "arn:${AWS::Partition}:cloudformation:${AWS::Region}:
${AWS::AccountId}:stack/*"
    }
  }
]
```

CloudWatchDashboardPolicy

Permet de placer des métriques à appliquer sur les CloudWatch tableaux de bord.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "cloudwatch:GetDashboard",
      "cloudwatch:ListDashboards",
      "cloudwatch:PutDashboard",
      "cloudwatch:ListMetrics"
    ],
    "Resource": "*"
  }
]
```

CloudWatchDescribeAlarmHistoryPolicy

Permet de décrire l'historique des CloudWatch alarmes Amazon.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "cloudwatch:DescribeAlarmHistory"
    ],
    "Resource": "*"
  }
]
```

CloudWatchPutMetricPolicy

Donne l'autorisation d'envoyer des métriques à CloudWatch.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "cloudwatch:PutMetricData"
    ],
    "Resource": "*"
  }
]
```

CodePipelineLambdaExecutionPolicy

Permet à une fonction Lambda invoquée par AWS CodePipeline de signaler l'état de la tâche.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "codepipeline:PutJobSuccessResult",
      "codepipeline:PutJobFailureResult"
    ],
    "Resource": "*"
  }
]
```

CodePipelineReadOnlyPolicy

Donne l'autorisation de lecture pour obtenir des informations sur un CodePipeline pipeline.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "codepipeline:ListPipelineExecutions"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:codepipeline:${AWS::Region}:${AWS::AccountId}:
${pipelinename}",
        {
          "pipelinename": {
            "Ref": "PipelineName"
          }
```

```
        }
      ]
    }
  }
]
```

CodeCommitCrudPolicy

Permet de créer, de lire, de mettre à jour et de supprimer des objets dans un CodeCommit référentiel spécifique.

```
"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

Permet de lire des objets dans un CodeCommit référentiel spécifique.

```
"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

Donne l'autorisation de détecter des entités, des expressions clés, des langues et des sentiments.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "comprehend:BatchDetectKeyPhrases",
      "comprehend:DetectDominantLanguage",
      "comprehend:DetectEntities",
      "comprehend:BatchDetectEntities",
      "comprehend:DetectKeyPhrases",
      "comprehend:DetectSentiment",
      "comprehend:BatchDetectDominantLanguage",
      "comprehend:BatchDetectSentiment"
    ],
    "Resource": "*"
  }
]
```

CostExplorerReadOnlyPolicy

Donne une autorisation en lecture seule à l'utilisateur en lecture seule (Cost AWS Cost Explorer Explorer) APIs pour l'historique de facturation.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ce:GetCostAndUsage",
      "ce:GetDimensionValues",
      "ce:GetReservationCoverage",
      "ce:GetReservationPurchaseRecommendation",
      "ce:GetReservationUtilization",
      "ce:GetTags"
    ],
    "Resource": "*"
  }
]
```

DynamoDBBackupFullAccessPolicy

Donne l'autorisation de lecture/écriture aux sauvegardes DynamoDB à la demande pour une table.

```
"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

Donne les autorisations de créer, lire, mettre à jour, supprimer à une table Amazon DynamoDB.

```
"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

Donne l'autorisation de lecture seule à une table DynamoDB.

```
"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

Donne l'autorisation de reconfigurer une table DynamoDB.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "dynamodb:UpdateTable"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
${tableName}",
        {
          "tableName": {
            "Ref": "TableName"
          }
        }
      ]
    }
  }
]
```

DynamoDBRestoreFromBackupPolicy

Donne l'autorisation de restaurer une table DynamoDB à partir d'une sauvegarde.

```
"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

Donne l'autorisation de décrire et de lire les flux DynamoDB Streams ainsi que les enregistrements.

```
"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

Donne l'autorisation d'écriture seule à une table DynamoDB.

```
"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

Donne l'autorisation de copier EC2 des images Amazon.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ec2:CopyImage"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:ec2:${AWS::Region}:${AWS::AccountId}:image/${imageId}",
        {
          "imageId": {
            "Ref": "ImageId"
          }
        }
      ]
    }
  }
]
```

EC2DescribePolicy

Donne l'autorisation de décrire les instances Amazon Elastic Compute Cloud (Amazon EC2).

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ec2:DescribeRegions",
      "ec2:DescribeInstances"
    ],
    "Resource": "*"
  }
]
```

EcsRunTaskPolicy

Donne l'autorisation de démarrer une nouvelle tâche pour une définition de tâche.

```
"Statement": [
```

```
  {
    "Action": [
      "ecs:RunTask"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:ecs:${AWS::Region}:${AWS::AccountId}:task-definition/
${taskDefinition}",
        {
          "taskDefinition": {
            "Ref": "TaskDefinition"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

EFSWriteAccessPolicy

Donne l'autorisation de monter un système de fichiers Amazon EFS avec accès en écriture.

```
"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

Donne l'autorisation de décrire ou de répertorier des clusters Amazon Elastic Kubernetes Service (Amazon EKS).

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "eks:DescribeCluster",
      "eks:ListClusters"
    ],
    "Resource": "*"
  }
]
```

ElasticMapReduceAddJobFlowStepsPolicy

Donne l'autorisation d'ajouter de nouvelles étapes à un cluster en cours d'exécution.

```
"Statement": [
  {
    "Action": "elasticmapreduce:AddJobFlowSteps",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:
${AWS::AccountId}:cluster/${clusterId}",
```

```
        {
          "clusterId": {
            "Ref": "ClusterId"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

ElasticMapReduceCancelStepsPolicy

Donne l'autorisation d'annuler une étape ou des étapes en attente dans un cluster en cours d'exécution.

```
"Statement": [
  {
    "Action": "elasticmapreduce:CancelSteps",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:
${AWS::AccountId}:cluster/${clusterId}",
        {
          "clusterId": {
            "Ref": "ClusterId"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

ElasticMapReduceModifyInstanceFleetPolicy

Donne l'autorisation de répertorier les détails et de modifier les capacités des flottes d'instances au sein d'un cluster.

```
"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

Donne l'autorisation de répertorier les détails et de modifier les réglages des groupes d'instances au sein d'un cluster.

```
"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

Donne l'autorisation de définir une protection contre l'arrêt pour un cluster.

```
"Statement": [
  {
    "Action": "elasticmapreduce:SetTerminationProtection",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:
${AWS::AccountId}:cluster/${clusterId}",
        {
          "clusterId": {
            "Ref": "ClusterId"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

ElasticMapReduceTerminateJobFlowsPolicy

Donne l'autorisation d'arrêter un cluster.

```
"Statement": [
  {
    "Action": "elasticmapreduce:TerminateJobFlows",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:
${AWS::AccountId}:cluster/${clusterId}",
        {
          "clusterId": {
            "Ref": "ClusterId"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

ElasticsearchHttpPostPolicy

Donne les autorisations POST et PUT à Amazon OpenSearch Service.

```
"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

Donne l'autorisation d'envoyer des événements à Amazon 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

Donne l'autorisation de filtrer CloudWatch les événements des journaux à partir d'un groupe de journaux spécifié.

```
"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

Donne l'autorisation de créer, d'écrire, de mettre à jour et de supprimer un flux de diffusion Firehose.

```
"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

Donne l'autorisation d'écrire dans un flux de diffusion Firehose.

```
"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

Donne l'autorisation de créer, publier et supprimer un flux Amazon Kinesis.

```
"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

Donne l'autorisation d'afficher et de lire un flux Amazon Kinesis.

```
"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

Donne l'autorisation de déchiffrer avec une clé AWS Key Management Service (AWS KMS). Notez qu'`keyId`il doit s'agir d'un identifiant de AWS KMS clé et non d'un alias de clé.

```
"Statement": [
  {
    "Action": "kms:Decrypt",
    "Effect": "Allow",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:kms:${AWS::Region}:${AWS::AccountId}:key/${keyId}",
        {
          "keyId": {
            "Ref": "KeyId"
```

```
          }
        }
      ]
    }
  }
]
```

KMSEncryptPolicy

Donne l'autorisation de chiffrer avec une AWS KMS clé. Notez que KeyID doit être un identifiant de AWS KMS clé et non un alias de clé.

```
"Statement": [
  {
    "Action": "kms:Encrypt",
    "Effect": "Allow",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:kms:${AWS::Region}:${AWS::AccountId}:key/${keyId}",
        {
          "keyId": {
            "Ref": "KeyId"
          }
        }
      ]
    }
  }
]
```

LambdaInvokePolicy

Donne l'autorisation d'invoquer une AWS Lambda fonction, un alias ou une version.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "lambda:InvokeFunction"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:lambda:${AWS::Region}:${AWS::AccountId}:function:
${functionName}*",
```

```
        {
          "functionName": {
            "Ref": "FunctionName"
          }
        }
      ]
    }
  }
]
```

MobileAnalyticsWriteOnlyAccessPolicy

Donne uniquement l'autorisation d'écriture pour placer les données d'événements pour toutes les ressources d'application.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "mobileanalytics:PutEvents"
    ],
    "Resource": "*"
  }
]
```

OrganizationsListAccountsPolicy

Donne l'autorisation en lecture seule de répertorier les noms de comptes enfants et. IDs

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "organizations:ListAccounts"
    ],
    "Resource": "*"
  }
]
```

PinpointEndpointAccessPolicy

Donne l'autorisation d'obtenir et de mettre à jour les points de terminaison pour une application Amazon Pinpoint.

```
"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

Donne l'autorisation d'accès complet aux ressources de lexique Amazon Polly.

```
"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

Donne l'autorisation de comparer et de détecter les visages et les étiquettes.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "rekognition:DetectFaces",
      "rekognition:DetectLabels",
      "rekognition:DetectModerationLabels",
      "rekognition:DetectText"
    ],
    "Resource": "*"
  }
]
```

RekognitionFacesManagementPolicy

Donne l'autorisation d'ajouter, de supprimer et de rechercher des visages dans une collection Amazon Rekognition.

```
"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

Donne l'autorisation de comparer et de détecter les visages et les étiquettes.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "rekognition:CompareFaces",
      "rekognition:DetectFaces"
    ],
    "Resource": "*"
  }
]
```

RekognitionLabelsPolicy

Donne l'autorisation de détecter les étiquettes d'objet et de modération.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "rekognition:DetectLabels",
      "rekognition:DetectModerationLabels"
    ],
    "Resource": "*"
  }
]
```

RekognitionNoDataAccessPolicy

Donne l'autorisation de comparer et de détecter les visages et les étiquettes.

```
"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

Donne l'autorisation de lister et de rechercher des visages.

```
"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

Donne l'autorisation de créer des collections et d'indexer des visages.

```
"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

Donne l'autorisation de modifier des jeux d'enregistrements de ressources dans Route 53.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "route53:ChangeResourceRecordSets"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:route53:::hostedzone/${HostedZoneId}",
        {
          "HostedZoneId": {
            "Ref": "HostedZoneId"
          }
        }
      ]
    }
  }
]
```

S3CrudPolicy

Donne l'autorisation de créer, lire, mettre à jour et supprimer pour agir sur les objets d'un compartiment Amazon S3.

```
"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

Donne l'autorisation d'accès complet pour agir sur les objets d'un compartiment Amazon S3.

```
"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

Donne l'autorisation de lire des objets en lecture seule dans un compartiment Amazon Simple Storage Service (Amazon S3).

```
"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

Donne l'autorisation d'écrire des objets dans un compartiment Amazon S3.

```
"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

Donne l'autorisation de créer une configuration de point de terminaison dans SageMaker AI.

```
"Statement": [
  {
    "Action": [
      "sagemaker:CreateEndpointConfig"
    ],
    "Resource": {
```

```
      "Fn::Sub": [
        "arn:${AWS::Partition}:sagemaker:${AWS::Region}:${AWS::AccountId}:endpoint-
config/${endpointConfigName}",
        {
          "endpointConfigName": {
            "Ref": "EndpointConfigName"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

SageMakerCreateEndpointPolicy

Donne l'autorisation de créer un point de terminaison dans SageMaker AI.

```
"Statement": [
  {
    "Action": [
      "sagemaker:CreateEndpoint"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:sagemaker:${AWS::Region}:${AWS::AccountId}:endpoint/
${endpointName}",
        {
          "endpointName": {
            "Ref": "EndpointName"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

ServerlessRepoReadWriteAccessPolicy

Donne l'autorisation de créer et de répertorier des applications dans le service AWS Serverless Application Repository (AWS SAM).

```
"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

Donne l'autorisation d'envoyer des e-mails Amazon SES, des e-mails modélisés, des e-mails groupés et de vérifier l'identité.

> **Note**
>
> L'action `ses:SendTemplatedEmail` nécessite un modèle d'ARN. Utilisez `SESBulkTemplatedCrudPolicy_v2` à la place.

```
"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

Donne l'autorisation d'envoyer des e-mails Amazon SES, des e-mails modélisés, des e-mails groupés et de vérifier l'identité.

```
"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

Donne l'autorisation d'envoyer un e-mail et de vérifier l'identité.

```
"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

Donne l'autorisation de créer, obtenir, répertorier, mettre à jour et supprimer les modèles d'e-mail Amazon SES.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ses:CreateTemplate",
      "ses:GetTemplate",
      "ses:ListTemplates",
      "ses:UpdateTemplate",
      "ses:DeleteTemplate",
      "ses:TestRenderTemplate"
    ],
    "Resource": "*"
  }
]
```

SESSendBouncePolicy

Donne SendBounce l'autorisation d'utiliser une identité Amazon Simple Email Service (Amazon SES).

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ses:SendBounce"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:ses:${AWS::Region}:${AWS::AccountId}:identity/
${identityName}",
        {
          "identityName": {
            "Ref": "IdentityName"
          }
        }
      ]
    }
```

```
  }
]
```

SNSCrudPolicy

Donne l'autorisation de créer et de publier des rubriques Amazon SNS, et de s'y abonner.

```
"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

Donne l'autorisation de publier un message dans une rubrique Amazon Simple Notification Service (Amazon SNS).

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "sns:Publish"
    ],
    "Resource": {
```

```
      "Fn::Sub": [
        "arn:${AWS::Partition}:sns:${AWS::Region}:${AWS::AccountId}:${topicName}",
        {
          "topicName": {
            "Ref": "TopicName"
          }
        }
      ]
    }
  }
]
```

SQSPollerPolicy

Donne l'autorisation d'interroger une file d'attente Amazon Simple Queue Service (Amazon SQS).

```
"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

Donne l'autorisation d'envoyer un message à une file d'attente Amazon SQS.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "sqs:SendMessage*"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:sqs:${AWS::Region}:${AWS::AccountId}:${queueName}",
        {
          "queueName": {
            "Ref": "QueueName"
          }
        }
      ]
    }
  }
]
```

SSMParameterReadPolicy

Permet d'accéder à un paramètre depuis un magasin de paramètres Amazon EC2 Systems Manager (SSM) pour charger des secrets dans ce compte. À utiliser lorsque le nom du paramètre n'est pas précédé d'une barre oblique.

> **Note**
>
> Si vous n'utilisez pas la clé par défaut, vous aurez également besoin de la stratégie `KMSDecryptPolicy`.

```
"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

Permet d'accéder à un paramètre depuis un magasin de paramètres Amazon EC2 Systems Manager (SSM) pour charger des secrets dans ce compte. À utiliser lorsque le nom du paramètre est précédé d'une barre oblique.

> **Note**
>
> Si vous n'utilisez pas la clé par défaut, vous aurez également besoin de la stratégie `KMSDecryptPolicy`.

```
"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

Donne l'autorisation de lancer l'exécution d'une machine d'état Step Functions.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "states:StartExecution"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:states:${AWS::Region}:${AWS::AccountId}:stateMachine:
${stateMachineName}",
        {
          "stateMachineName": {
            "Ref": "StateMachineName"
          }
        }
      ]
    }
  }
]
```

TextractDetectAnalyzePolicy

Donne l'accès permettant de détecter et d'analyser des documents avec Amazon Textract.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "textract:DetectDocumentText",
      "textract:StartDocumentTextDetection",
      "textract:StartDocumentAnalysis",
      "textract:AnalyzeDocument"
    ],
    "Resource": "*"
  }
]
```

TextractGetResultPolicy

Donne l'accès permettant d'obtenir des documents détectés et analysés avec Amazon Textract.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "textract:GetDocumentTextDetection",
      "textract:GetDocumentAnalysis"
    ],
    "Resource": "*"
  }
]
```

TextractPolicy

Donne l'accès complet à Amazon Textract.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "textract:*"
    ],
```

```
    "Resource": "*"
  }
]
```

VPCAccessPolicy

Donne l'accès pour créer, supprimer, décrire et détacher des interfaces réseau élastiques.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ec2:CreateNetworkInterface",
      "ec2:DeleteNetworkInterface",
      "ec2:DescribeNetworkInterfaces",
      "ec2:DetachNetworkInterface"
    ],
    "Resource": "*"
  }
]
```

Gestion des AWS SAM autorisations à l'aide de CloudFormation mécanismes

Pour contrôler l'accès aux AWS ressources, le AWS Serverless Application Model (AWS SAM) peut utiliser les mêmes mécanismes que CloudFormation. Pour plus d'informations, consultez Contrôle de l'accès à l'aide de Gestion des identités et des accès AWS dans le Guide de l'utilisateur AWS CloudFormation .

Il existe trois options principales pour accorder à un utilisateur l'autorisation de gérer des applications sans serveur. Chaque option offre aux utilisateurs différents niveaux de contrôle d'accès.

- Accorder des autorisations d'administrateur
- Joignez les politiques AWS gérées nécessaires.
- Accordez des autorisations spécifiques Gestion des identités et des accès AWS (IAM).

Selon l'option choisie, les utilisateurs ne peuvent gérer que les applications sans serveur contenant des AWS ressources auxquelles ils sont autorisés à accéder.

Les sections suivantes décrivent chaque option plus en détail.

Accorder des autorisations d'administrateur

Si vous accordez des autorisations d'administrateur à un utilisateur, celui-ci peut gérer des applications sans serveur contenant n'importe quelle combinaison de AWS ressources. C'est l'option la plus simple, mais elle accorde également aux utilisateurs l'ensemble d'autorisations le plus étendu, ce qui leur permet de réaliser des actions d'un impact le plus élevé.

Pour plus d'informations sur l'attribution d'autorisations d'administrateur à un utilisateur, consultez Création de votre premier utilisateur administrateur et groupe IAM dans le Guide de l'utilisateur IAM

Joindre les politiques AWS gérées nécessaires

Vous pouvez accorder aux utilisateurs un sous-ensemble d'autorisations en utilisant des stratégies gérées par AWS, plutôt que d'accorder des autorisations d'administrateur complètes. Si vous utilisez cette option, assurez-vous que l'ensemble de politiques AWS gérées couvre toutes les actions et ressources requises pour les applications sans serveur gérées par les utilisateurs.

Par exemple, les politiques AWS gérées suivantes sont suffisantes pour déployer l'exemple d'application Hello World :

- AWSCloudFormationFullAccess
- IAMFullAccès
- AWSLambda_FullAccess
- APIGatewayAdministrateur Amazon
- Amazon S3 FullAccess
- Amazon EC2 ContainerRegistryFullAccess

Pour plus d'informations sur l'attachement de stratégies à un utilisateur IAM, consultez Modification des autorisations d'un utilisateur IAM dans le Guide de l'utilisateur IAM.

Accorder des autorisations (IAM) spécifiques.

Pour un niveau de contrôle d'accès plus détaillé, vous pouvez accorder des autorisations IAM spécifiques aux utilisateurs à l'aide des déclarations de stratégie. Si vous utilisez cette option, assurez-vous que la déclaration de stratégie comprenne toutes les actions et ressources requises pour les applications sans serveur gérées par les utilisateurs.

La bonne pratique avec cette option consiste à refuser aux utilisateurs l'autorisation de créer des rôles, y compris des rôles d'exécution Lambda, afin qu'ils ne puissent pas s'accorder eux-mêmes

des autorisations progressives. Ainsi, en tant qu'administrateur, vous devez d'abord créer un Rôle d'exécution Lambda qui sera spécifié dans les applications sans serveur que les utilisateurs vont gérer. Pour plus d'informations sur la création de rôles d'exécution Lambda, consultez Création d'un rôle d'exécution dans la console IAM.

Pour l'exemple d'application Hello World, AWSLambdaBasicExecutionRolec'est suffisant pour exécuter l'application. Après avoir créé un rôle d'exécution Lambda, modifiez le fichier AWS SAM modèle de l'exemple d'application Hello World pour ajouter la propriété suivante à la `AWS::Serverless::Function` ressource :

```
Role: lambda-execution-role-arn
```

Lorsque l'application Hello World modifiée est en place, la déclaration de stratégie suivante accorde des autorisations suffisantes aux utilisateurs pour le déploiement, la mise à jour et l'élimination de l'application :

> **Note**
>
> L'exemple de déclaration de stratégie de cette section vous permet de déployer, de mettre à jour et d'éliminer l'exemple d'application Hello World. Si vous ajoutez des types de ressources supplémentaires à l'application, vous devez mettre à jour la déclaration de stratégie pour inclure les éléments suivants :
>
> 1. Autorisation permettant à l'application d'appeler les actions du service.
> 2. Le principal du service, si nécessaire pour les actions du service.
>
> Par exemple, si vous ajoutez un flux de travail Step Functions, vous devrez peut-être ajouter des autorisations pour les actions répertoriées ici et le `states.amazonaws.com`principal du service.

Pour plus d'informations sur la gestion des stratégies IAM, consultez Gestion des stratégies IAM dans le Guide de l'utilisateur IAM.

Contrôlez l'accès aux API avec votre AWS SAM modèle

Le contrôle de l'accès à votre API Gateway APIs permet de garantir que votre application sans serveur est sécurisée et n'est accessible que par le biais de l'autorisation que vous autorisez. Vous

pouvez activer l'autorisation dans votre AWS SAM modèle pour contrôler qui peut accéder à votre API Gateway APIs.

AWS SAM prend en charge plusieurs mécanismes pour contrôler l'accès à votre API Gateway APIs. L'ensemble des mécanismes pris en charge diffère entre les types de ressources `AWS::Serverless::HttpApi` et `AWS::Serverless::Api`.

Le tableau suivant récapitule les mécanismes pris en charge par type de ressource.

Mécanismes de contrôle de l'accès	AWS::Serverless::HttpApi	AWS::Serverless::Api
Mécanismes d'autorisation Lambda	✓	✓
Autorisations IAM		✓
Groupes d'utilisateurs Amazon Cognito	✓ *	✓
Clés d'API		✓
Politiques basées sur une ressource		✓
OAuth 2.0/Autorisateurs JWT	✓	

* Vous pouvez utiliser Amazon Cognito en tant qu'auteur de jeton web JSON (JWT) avec le type de ressource `AWS::Serverless::HttpApi`.

- Mécanismes d'autorisation – Un mécanisme d'autorisation Lambda (anciennement appelé Custom Authorizer (mécanisme d'autorisation personnalisée)) est une fonction Lambda vous permettant de contrôler l'accès à vos API. Lorsque votre API est appelée, cette fonction Lambda est appelée avec un contexte de requête ou un jeton d'autorisation fourni par l'application cliente. La fonction Lambda répond si l'appelant est autorisé à effectuer l'opération demandée.

  Les types de ressources `AWS::Serverless::HttpApi` et `AWS::Serverless::Api` prennent tous deux en charge les mécanismes d'autorisation Lambda.

Pour plus d'informations sur les autorisateurs Lambda avec`AWS::Serverless::HttpApi`, consultez la section Working with AWS Lambda authorizers for HTTP dans le Guide du développeur APIs d'API Gateway. Pour plus d'informations sur les mécanismes d'autorisation Lambda avec `AWS::Serverless::Api`, consultez Utilisation des mécanismes d'autorisation Lambda API Gateway dans le Guide du développeur API Gateway.

Pour obtenir des exemples de mécanismes d'autorisation Lambda pour l'un ou l'autre des deux type de ressource, consultez Exemples d'autorisateurs Lambda pour AWS SAM.

- Autorisations IAM – Vous pouvez contrôler les personnes pouvant appeler votre API en utilisant Gestion des identités et des accès AWS (IAM) autorisations. Les utilisateurs appelant votre API doivent être authentifiés avec des informations d'identification IAM. Les appels vers votre API ne réussissent que s'il existe une stratégie IAM attachée à l'utilisateur IAM qui représente l'appelant API, un groupe IAM qui contient l'utilisateur ou un rôle IAM assumé par l'utilisateur.

  Seul le type de ressource `AWS::Serverless::Api` prend en charge les autorisations IAM.

  Pour plus d'informations, consultez le Contrôle de l'accès à une API avec des autorisations IAM dans le Guide du développeur API. Pour obtenir un exemple, consultez Exemple d'autorisation IAM pour AWS SAM.
- Groupes d'utilisateurs Amazon Cognito – Les groupes d'utilisateurs Amazon Cognito sont des répertoires d'utilisateurs dans Amazon Cognito. Un client de votre API doit d'abord se connecter à un groupe d'utilisateurs et obtenir un jeton d'identité ou d'accès pour l'utilisateur. Ensuite, le client appelle votre API avec l'un des jetons renvoyés. L'appel d'API réussit uniquement si le jeton requis est valide.

  Le type de ressource `AWS::Serverless::Api` prend en charge les groupes d'utilisateurs Amazon Cognito. Le type de ressource `AWS::Serverless::HttpApi` prend en charge l'utilisation d'Amazon Cognito en tant qu'auteur de JWT.

  Pour plus d'informations, consultez Contrôle de l'accès à une API REST à l'aide de groupes d'utilisateurs Amazon Cognito en tant que mécanisme d'autorisation dans le Guide du développeur API Gateway. Pour obtenir un exemple, consultez Exemple de groupe d'utilisateurs Amazon Cognito pour AWS SAM.
- Les clés API sont des valeurs de chaînes alphanumériques que vous distribuez aux clients développeurs d'applications pour leur accorder l'accès à votre API.

Seul le type de ressource `AWS::Serverless::Api` prend en charge les clés API.

Pour plus d'informations concernant les clés API, consultez Création et utilisation de plans d'utilisation avec les clés API dans le Guide du développeur API Gateway. Pour obtenir un exemple de clés API, voirExemple de clé d'API pour AWS SAM.

- Stratégies de ressources – Les stratégies de ressources sont des documents de stratégie JSON que vous pouvez attacher à une API API Gateway. Vous pouvez utiliser les stratégies de ressources pour contrôler si un mandataire spécifié (généralement, un utilisateur ou un rôle IAM) peut appeler l'API.

  Seul le type de `AWS::Serverless::Api` ressource prend en charge les politiques de ressources en tant que mécanisme de contrôle de l'accès à API Gateway APIs.

  Pour plus d'informations sur les stratégies de ressources, consultez Contrôle de l'accès à une API avec des stratégies de ressources API Gateway dans le Guide du développeur API. Pour un exemple de stratégies de ressources, consultez Exemple de politique de ressources pour AWS SAM.

- OAuth Autorisateurs 2.0/JWT — Vous pouvez les utiliser dans JWTs le cadre des frameworks OpenID Connect (OIDC) et OAuth 2.0 pour contrôler l'accès à votre. APIs API Gateway valide les JWTs demandes d'API soumises par les clients et autorise ou refuse les demandes en fonction de la validation du jeton et, éventuellement, des étendues du jeton.

  Seul le type de `AWS::Serverless::HttpApi` ressource prend en charge les autorisateurs OAuth 2.0/JWT.

  Pour plus d'informations, consultez la section Contrôle de l'accès au protocole HTTP APIs avec les autorisateurs JWT dans le guide du développeur d'API Gateway. Pour obtenir un exemple, consultez OAuth Exemple d'autorisateur 2.0/JWT pour AWS SAM.

Choix d'un mécanisme de contrôle d'accès

Le mécanisme que vous choisissez d'utiliser pour contrôler l'accès à votre API Gateway APIs dépend de plusieurs facteurs. Par exemple, si vous avez un projet inédit sans autorisation ni contrôle d'accès configuré, les groupes d'utilisateurs Amazon Cognito peuvent constituer votre meilleure option. En effet, lorsque vous configurez des groupes d'utilisateurs, vous configurez également automatiquement l'authentification et le contrôle d'accès.

Cependant, si votre application a déjà configuré l'authentification, l'utilisation de mécanismes d'autorisation Lambda pourrait être votre meilleur choix. Cela est dû au fait que vous pouvez appeler votre service d'authentification existant et renvoyer un document de stratégie basé sur la réponse. En outre, si votre application nécessite une authentification personnalisée ou une logique de contrôle d'accès que les groupes d'utilisateurs ne prennent pas en charge, alors les autorisations Lambda pourraient constituer votre meilleure option.

Lorsque vous avez choisi le mécanisme à utiliser, consultez la section correspondante Exemples pour savoir comment l'utiliser AWS SAM pour configurer votre application afin qu'elle utilise ce mécanisme.

Personnalisation des réponses d'erreur

Vous pouvez l'utiliser AWS SAM pour personnaliser le contenu de certaines réponses d'erreur d'API Gateway. Seul le type de ressource `AWS::Serverless::Api` prend en charge les réponses personnalisées API Gateway.

Pour plus d'informations sur les réponses API Gateway, consultez Réponses de passerelle dans API Gateway dans le Guide du développeur API Gateway. Pour obtenir un exemple de réponses personnalisées, consultez Exemple de réponse personnalisée pour AWS SAM.

Exemples

- Exemples d'autorisateurs Lambda pour AWS SAM
- Exemple d'autorisation IAM pour AWS SAM
- Exemple de groupe d'utilisateurs Amazon Cognito pour AWS SAM
- Exemple de clé d'API pour AWS SAM
- Exemple de politique de ressources pour AWS SAM
- OAuth Exemple d'autorisateur 2.0/JWT pour AWS SAM
- Exemple de réponse personnalisée pour AWS SAM

Exemples d'autorisateurs Lambda pour AWS SAM

Le type de ressource `AWS::Serverless::Api` prend en charge deux types de mécanismes d'autorisation Lambda : les mécanismes d'autorisation de `TOKEN` et les mécanismes d'autorisation de `REQUEST`. Le type de ressource `AWS::Serverless::HttpApi` prend uniquement en charge les mécanismes d' autorisation `REQUEST`. Voici des exemples de chaque type.

Exemples d'autorisation Lambda **`TOKEN`** (AWS::Serverless::Api)

Vous pouvez contrôler l'accès à votre APIs en définissant un `TOKEN` autorisateur Lambda dans votre modèle. AWS SAM Vous devez pour cela utiliser le type de données ApiAuth.

Voici un exemple de section de AWS SAM modèle pour un autorisateur Lambda `TOKEN` :

> **Note**
>
> Dans l'exemple suivant, le SAM `FunctionRole` est généré implicitement.

```
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
```

Pour plus d'informations sur les mécanismes d'autorisation Lambda, consultez Utilisation des mécanismes d'autorisation pour les API Gateway Lambda dans le Guide du développeur API Gateway.

Exemples d'autorisation Lambda **REQUEST** (AWS::Serverless::Api)

Vous pouvez contrôler l'accès à votre APIs en définissant un `REQUEST` autorisateur Lambda dans votre modèle. AWS SAM Vous devez pour cela utiliser le type de données ApiAuth.

Voici un exemple de section de AWS SAM modèle pour un autorisateur Lambda `REQUEST` :

```
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
```

Pour plus d'informations sur les mécanismes d'autorisation Lambda, consultez Utilisation des mécanismes d'autorisation pour les API Gateway Lambda dans le Guide du développeur API Gateway.

Exemples d'autorisation Lambda (AWS::Serverless::HttpApi)

Vous pouvez contrôler l'accès à votre HTTP APIs en définissant un autorisateur Lambda dans votre modèle. AWS SAM Vous devez pour cela utiliser le type de données HttpApiAuth.

Voici un exemple de section de AWS SAM modèle pour un autorisateur Lambda :

```
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
```

Exemple d'autorisation IAM pour AWS SAM

Vous pouvez contrôler l'accès à votre APIs en définissant les autorisations IAM dans votre AWS SAM modèle. Vous devez pour cela utiliser le type de données ApiAuth.

Voici un exemple de AWS SAM modèle utilisé pour les autorisations IAM :

```
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}
```

Pour plus d'informations sur les autorisations IAM, consultez le Contrôle de l'accès à une API dans le Guide du développeur API Gateway.

Exemple de groupe d'utilisateurs Amazon Cognito pour AWS SAM

Vous pouvez contrôler l'accès à votre compte APIs en définissant des groupes d'utilisateurs Amazon Cognito dans votre AWS SAM modèle. Vous devez pour cela utiliser le type de données ApiAuth.

Voici un exemple de section de AWS SAM modèle pour un groupe d'utilisateurs :

```
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
```

Pour plus informations sur les groupes d'utilisateurs Amazon Cognito, consultez Contrôle de l'accès à une API REST à l'aide de groupes d'utilisateurs Amazon Cognito en tant que mécanisme d'autorisation dans le Guide du développeur API Gateway.

Exemple de clé d'API pour AWS SAM

Vous pouvez contrôler l'accès à votre APIs en exigeant des clés d'API dans votre AWS SAM modèle. Vous devez pour cela utiliser le type de données ApiAuth.

Voici un exemple de section de AWS SAM modèle pour les clés d'API :

```
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
```

Pour plus d'informations concernant les clés API, consultez Création et utilisation de plans d'utilisation avec les clés API dans le Guide du développeur API Gateway.

Exemple de politique de ressources pour AWS SAM

Vous pouvez contrôler l'accès à votre APIs en joignant une politique de ressources à votre AWS SAM modèle. Vous devez pour cela utiliser le type de données ApiAuth.

Voici un exemple de AWS SAM modèle pour une API privée. Une API privée doit disposer d'une politique de ressources pour être déployée.

```
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
```

Pour plus d'informations sur les stratégies de ressources, consultez Contrôle de l'accès à une API avec des stratégies de ressources API Gateway dans le Guide du développeur API. Pour plus d'informations sur le mode privé APIs, consultez la section Création d'une API privée dans Amazon API Gateway dans le guide du développeur d'API Gateway.

OAuth Exemple d'autorisateur 2.0/JWT pour AWS SAM

Vous pouvez contrôler l'accès à votre APIs utilisation dans JWTs le cadre des frameworks OpenID Connect (OIDC) et OAuth 2.0. Vous devez pour cela utiliser le type de données HttpApiAuth.

Voici un exemple de section de AWS SAM modèle pour un autorisateur OAuth 2.0/JWT :

```
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
```

Pour plus d'informations sur les autorisateurs OAuth 2.0/JWT, consultez la section Contrôle de l'accès au HTTP APIs avec les autorisateurs JWT dans le guide du développeur d'API Gateway.

Exemple de réponse personnalisée pour AWS SAM

Vous pouvez personnaliser certaines réponses d'erreur API Gateway en définissant des en-têtes de réponse dans votre modèle AWS SAM . Vous devez pour cela utiliser le type de données Objet de réponse de passerelle.

Voici un exemple de AWS SAM modèle qui crée une réponse personnalisée à l'`DEFAULT_5XX`erreur.

```
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
```

Pour plus d'informations sur les réponses API Gateway, consultez Réponses de passerelle dans API Gateway dans le Guide du développeur API Gateway.

Améliorez l'efficacité en utilisant les couches Lambda avec AWS SAM

En utilisant AWS SAM, vous pouvez inclure des couches dans vos applications sans serveur. AWS Lambda les couches vous permettent d'extraire le code d'une fonction Lambda dans une couche Lambda qui peut ensuite être utilisée dans plusieurs fonctions Lambda. Cela vous permet de réduire la taille de vos packages de déploiement, de séparer la logique des fonctions de base des dépendances et de partager les dépendances entre plusieurs fonctions. Pour plus d'informations sur les couches, consultez la section Couches Lambda dans le Guide du AWS Lambda développeur.

Cette rubrique fournit des informations sur les éléments suivants :

- Inclusion de couches dans votre application
- Comment les couches sont mises en cache localement

Pour plus d'informations sur la création de couches personnalisées, consultez Création de couches Lambda dans AWS SAM.

Inclusion de couches dans votre application

Pour inclure des couches dans votre application, utilisez la propriété `Layers` du type de ressource AWS::Serverless::Function.

Voici un exemple de AWS SAM modèle avec une fonction Lambda qui inclut une couche :

```
ServerlessFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: .
    Handler: my_handler
    Runtime: Python3.7
    Layers:
        - <LayerVersion ARN>
```

Comment les couches sont mises en cache localement

Lorsque vous appelez votre fonction à l'aide de l'une des commandes `sam local`, le package de couches de votre fonction est téléchargé et mis en cache sur votre hôte local.

Le tableau suivant montre les emplacements de répertoire des mises en cache par défaut pour les différents systèmes d'exploitation.

Système d’exploitation	Location
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`

Une fois que le package est mis en cache, la CLI AWS SAM superpose les couches sur une image Docker utilisée pour invoquer votre fonction. AWS SAMCLIGénère les noms des images qu'il crée, ainsi que celles LayerVersions qui sont conservées dans le cache. Vous trouverez davantage de détails sur le schéma dans les sections suivantes.

Pour inspecter les couches superposées, exécutez la commande suivante pour démarrer une séance bash dans l'image à inspecter :

```
docker run -it --entrypoint=/bin/bash samcli/lambda:<Tag following the schema outlined
 in Docker Image Tag Schema> -i
```

Schéma de nom de répertoire de mise en cache des couches

Étant donné LayerVersionArn qu'un est défini dans votre modèle, la AWS SAMCLI version LayerName et est extraite de l'ARN. Elle crée un répertoire pour y placer le contenu de la couche, qui sera nommé `LayerName-Version-<first 10 characters of sha256 of ARN>`.

Exemple :

```
ARN = arn:aws:lambda:us-west-2:111111111111:layer:myLayer:1
Directory name = myLayer-1-926eeb5ff1
```

Schéma de balises pour les images Docker

Pour calculer le hachage des couches uniques, combinez tous les noms de couches uniques avec un délimiteur « - », prenez le SHA256 hachage, puis les 10 premiers caractères.

Exemple :

```
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
```

Les noms uniques sont calculés de la même manière que le schéma de nom du répertoire de mise en cache des couches :

```
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
```

Pour calculer le hachage des couches uniques, combinez tous les noms de couches uniques avec un délimiteur de '-', prenez le hachage sha256, puis prenez les 25 premiers caractères.

```
myLayer-1-926eeb5ff1-mySecondLayer-1-6bc1022bdf = 2dd7ac5ffb30d515926aef
```

Combinez ensuite cette valeur avec l'exécution et l'architecture de la fonction, avec un délimiteur de '-' :

```
python3.7-x86_64-2dd7ac5ffb30d515926aefffd
```

Réutilisez le code et les ressources à l'aide d'applications imbriquées dans AWS SAM

Une application sans serveur peut comporter une ou plusieurs applications imbriquées. Une application imbriquée fait partie d'une application plus vaste et peut être empaquetée et déployée en tant qu'artefact autonome ou en tant que composant d'une application plus vaste. Les applications imbriquées vous permettent de transformer le code fréquemment utilisé en une seule application qui peut ensuite être réutilisée dans une application sans serveur plus importante ou dans plusieurs applications sans serveur.

Au fur et à mesure que vos architectures sans serveur se développent, des modèles communs apparaissent généralement dans lesquels les mêmes composants sont définis dans plusieurs modèles d'applications. Les applications imbriquées vous permettent de réutiliser le code, les fonctionnalités, les ressources et les configurations courants dans des AWS SAM modèles distincts, ce qui vous permet de ne gérer que le code provenant d'une source unique. Cela permet de réduire le code et les configurations dupliqués. En outre, cette approche modulaire rationalise le développement, améliore l'organisation du code et facilite la cohérence entre les applications sans serveur. Avec les applications imbriquées, vous pouvez rester plus concentré sur la logique métier propre à votre application.

Pour définir une application imbriquée dans votre application sans serveur, utilisez le type de ressource AWS::Serverless::Application.

Vous pouvez définir des applications imbriquées à partir des deux sources suivantes :

- Une application AWS Serverless Application Repository – Vous pouvez définir des applications imbriquées à l'aide d'applications disponibles pour votre compte dans le AWS Serverless Application Repository. Celles-ci peuvent être des applications privées dans votre compte, des applications qui sont partagé de manière privée avec votre compte, ou des applications qui sont publiquement partagées dans le AWS Serverless Application Repository. Pour plus d'informations

sur les différents niveaux d'autorisation de déploiement, consultez Autorisations de déploiement d'applications et Publication des applications dans le Guide du développeur AWS Serverless Application Repository .

- Une application locale – Vous pouvez définir des applications imbriquées à l'aide d'applications qui sont stockées sur votre système de fichiers local.

Consultez les sections suivantes pour savoir comment AWS SAM définir ces deux types d'applications imbriquées dans votre application sans serveur.

> **Note**
>
> Le nombre maximal d'applications pouvant être imbriquées dans une application sans serveur est de 200.
> Le nombre maximal de paramètres qu'une application imbriquée peut avoir est de 60.

Définition d'une application imbriquée à partir du AWS Serverless Application Repository

Vous pouvez définir des applications imbriquées à l'aide d'applications qui sont disponibles pour votre compte dans le AWS Serverless Application Repository. Vous pouvez également stocker et distribuer des applications qui contiennent des applications imbriquées à l'aide du AWS Serverless Application Repository. Pour consulter les détails d'une application imbriquée dans le AWS Serverless Application Repository, vous pouvez utiliser le AWS SDK, la console Lambda ou la AWS CLI console Lambda.

Pour définir une application hébergée AWS Serverless Application Repository dans le AWS SAM modèle de votre application sans serveur, utilisez le bouton Copier en tant que ressource SAM sur la page détaillée de chaque AWS Serverless Application Repository application. Pour cela, procédez comme suit :

1. Assurez-vous que vous êtes connecté à la AWS Management Console.
2. Trouvez l'application dans laquelle vous souhaitez vous intégrer AWS Serverless Application Repository en suivant les étapes décrites dans la section Navigation, recherche et déploiement d'applications du guide du AWS Serverless Application Repository développeur.
3. Cliquez sur le bouton Copier comme ressource SAM. La section de modèle SAM de l'application que vous consultez est maintenant dans votre Presse-papiers.

4. Collez la section du modèle SAM dans la section `Resources:` du fichier de modèle SAM pour l'application que vous souhaitez imbriquer dans cette application.

Voici un exemple de section de modèle SAM pour une application imbriquée hébergée dans le AWS Serverless Application Repository :

```
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
```

Si aucun paramètre n'est requis, vous pouvez omettre la section `Parameters:` du modèle.

> **Important**
>
> Les applications qui contiennent des applications imbriquées hébergées dans le AWS Serverless Application Repository héritent des restrictions de partage des applications imbriquées.
> Supposons, par exemple, qu'une application soit partagée publiquement, mais qu'elle contienne une application imbriquée qui n'est partagée en privé qu'avec le AWS compte qui a créé l'application parent. Dans ce cas, si votre AWS compte n'est pas autorisé à déployer l'application imbriquée, vous ne pouvez pas déployer l'application parent. Pour plus d'informations sur les autorisation de déploiement des applications, consultez Autorisations de déploiement d'applications et Publication des applications dans le Guide du développeur AWS Serverless Application Repository .

Définition d'une application imbriquée à partir du système de fichiers local

Vous pouvez définir des applications imbriquées à l'aide d'applications qui sont stockées sur votre système de fichiers local. Pour ce faire, spécifiez le chemin d'accès au fichier AWS SAM modèle stocké sur votre système de fichiers local.

Voici un exemple de section de modèle SAM pour une application imbriquée :

```
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
```

S'il n'y a pas de réglage de paramètres, vous pouvez omettre la section `Parameters:` du modèle.

Déploiement d'applications imbriquées

Vous pouvez déployer votre application imbriquée en utilisant la commande `sam deploy` de la CLI AWS SAM. Pour en savoir plus, consultez Déployez votre application et vos ressources avec AWS SAM.

> **Note**
>
> Lorsque vous déployez une application qui contient des applications imbriquées, vous devez reconnaître qu'elle contient des applications imbriquées. Pour ce faire, passez `CAPABILITY_AUTO_EXPAND` à l'CreateCloudFormationChangeSet API ou utilisez la `aws serverlessrepo create-cloud-formation-change-set` AWS CLI commande. Pour plus d'informations sur la reconnaissance des applications imbriquées, consultez Reconnaissance des rôles, des stratégies de ressources et des applications imbriquées IAM lors du déploiement d'applications dans le Guide du développeur AWS Serverless Application Repository .

Gérez les événements temporels avec le EventBridge planificateur dans AWS SAM

Le contenu de cette rubrique explique en détail ce qu'est Amazon EventBridge Scheduler, quelles sont les AWS SAM offres d'assistance, comment créer des événements Scheduler, ainsi que des exemples auxquels vous pouvez vous référer lors de la création d'événements Scheduler.

Qu'est-ce qu'Amazon EventBridge Scheduler ?

Utilisez le EventBridge planificateur pour planifier des événements dans vos AWS SAM modèles. Amazon EventBridge Scheduler est un service de planification qui vous permet de créer, de lancer et de gérer des dizaines de millions d'événements et de tâches dans tous les AWS services. Ce service est particulièrement utile pour les événements liés au temps. Vous pouvez l'utiliser pour planifier des événements et des appels récurrents basés sur le temps. Il prend également en charge les événements ponctuels ainsi que les expressions de taux et de chronologie avec une heure de début et de fin.

Pour en savoir plus sur Amazon EventBridge Scheduler, consultez Qu'est-ce qu'Amazon EventBridge Scheduler ? dans le guide de l'utilisateur EventBridge du planificateur.

Rubriques

- EventBridge Support du planificateur dans AWS SAM
- Création d' EventBridge événements du planificateur dans AWS SAM
- Exemples
- En savoir plus

EventBridge Support du planificateur dans AWS SAM

La spécification du modèle AWS Serverless Application Model (AWS SAM) fournit une syntaxe simple et abrégée que vous pouvez utiliser pour planifier des événements avec EventBridge Scheduler for et. AWS Lambda AWS Step Functions

Création d' EventBridge événements du planificateur dans AWS SAM

Définissez la `ScheduleV2` propriété comme type d'événement dans votre AWS SAM modèle pour définir votre événement EventBridge Scheduler. Cette propriété prend en charge les types de ressources `AWS::Serverless::Function` et `AWS::Serverless::StateMachine`.

```
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 La planification des événements du planificateur prend également en charge les files d'attente de lettres mortes (DLQ) pour les événements non traités. Pour plus d'informations sur les files d'attente de lettres mortes, consultez la section Configuration d'une file d'attente de lettres mortes pour le planificateur dans le guide de l'utilisateur du EventBridge planificateur. EventBridge

Lorsqu'un ARN DLQ est spécifié, AWS SAM configure les autorisations pour que le planificateur planifie l'envoi de messages au DLQ. Lorsqu'aucun ARN DLQ n'est spécifié, la AWS SAM ressource DLQ est créée.

Exemples

Exemple de base de définition d'un événement EventBridge Scheduler avec 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"}'
```

En savoir plus

Pour en savoir plus sur la définition de la propriété `ScheduleV2` EventBridge Scheduler, voir :

- ScheduleV2 pour `AWS::Serverless::Function`.
- ScheduleV2 pour `AWS::Serverless::StateMachine`.

Orchestrer les AWS SAM ressources avec AWS Step Functions

Vous pouvez l'utiliser AWS Step Functionspour orchestrer des AWS Lambda fonctions et d'autres AWS ressources afin de créer des flux de travail complexes et robustes. Step Functions pour indiquer à votre application quand et dans quelles conditions vos AWS ressources, telles que AWS Lambda les fonctions, sont utilisées. Cela simplifie le processus de création de flux de travail complexes et robustes. À l'aide deAWS::Serverless::StateMachine, vous définissez les différentes étapes de votre flux de travail, associez des ressources à chaque étape, puis séquencez ces étapes ensemble. Vous ajoutez également des transitions et des conditions là où elles sont nécessaires. Cela simplifie le processus de création d'un flux de travail complexe et robuste.

> **Note**
>
> Pour gérer les AWS SAM modèles contenant des machines d'état Step Functions, vous devez utiliser la version 0.52.0 ou ultérieure du AWS SAMCLI. Pour vérifier la version dont vous disposez, exécutez la commande `sam --version`.

Step Functions repose sur les concepts de tâches et de machines d'état. Vous définissez les machines d'état à l'aide du Langage des états Amazon basé sur JSON. La console Step Functions affiche une vue graphique de la structure de votre machine d'état, ce qui vous permet de vérifier visuellement la logique de votre machine d'état et de contrôler les exécutions.

Grâce à la prise en charge de Step Functions dans AWS Serverless Application Model (AWS SAM), vous pouvez effectuer les opérations suivantes :

- Définissez des machines à états, soit directement dans un AWS SAM modèle, soit dans un fichier séparé
- Créez des rôles d'exécution de machines à états via des modèles de AWS SAM politiques, des politiques intégrées ou des politiques gérées

- Déclenchez des exécutions automatiques à l'aide d' EventBridge événements API Gateway ou Amazon, selon un calendrier défini dans un AWS SAM modèle ou en appelant APIs directement
- Utiliser des AWS SAM Modèles de stratégie disponibles pour les modèles de développement courants Step Functions.

Exemple

L'extrait d'un fichier AWS SAM modèle suivant définit une machine d'état Step Functions dans un fichier de définition. Remarque : le fichier `my_state_machine.asl.json` doit être écrit dans la Langue des états Amazon.

```
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
      ...
```

Pour télécharger un exemple d' AWS SAM application qui inclut une machine d'état Step Functions, voir Create a Step Functions State Machine Using AWS SAM dans le manuel du AWS Step Functions développeur.

En savoir plus

Pour en savoir plus sur Step Functions et son utilisation avec Step Functions AWS SAM, consultez les rubriques suivantes :

- Fonctionnement de AWS Step Functions
- AWS Step Functions et AWS Serverless Application Model
- Tutoriel : Création d'une machine à états Step Functions à l'aide de AWS SAM
- AWS SAM Spécification : AWS::Serverless::StateMachine

Configurer la signature de code pour votre AWS SAM application

Pour garantir que seul le code fiable est déployé, vous pouvez AWS SAM activer la signature de code avec vos applications sans serveur. La signature de votre code permet de garantir que le code n'a pas été modifié depuis la signature et que seuls les packages de code signés provenant d'éditeurs approuvés s'exécutent dans vos fonctions Lambda. Cela permet aux entreprises de se libérer de la charge de créer des composants de contrôle d'accès dans leurs pipelines de déploiement.

Pour plus d'informations sur la signature de code, consultez la section Configuration de la signature de code pour les fonctions Lambda dans le Guide du AWS Lambda développeur.

Avant de configurer la signature de code pour votre application sans serveur, vous devez créer un profil de signature à l'aide de AWS Signer. Vous utiliserez ce profil de signature pour les tâches suivantes :

1. Création d'une configuration de signature de code – Déclarez une ressource `AWS::Lambda::CodeSigningConfig` pour spécifier les profils de signature des éditeurs approuvés et définir l'action de stratégie pour les vérifications de validation. Vous pouvez déclarer cet objet dans le même AWS SAM modèle que votre fonction sans serveur, dans un autre AWS SAM modèle ou dans un CloudFormation modèle. Vous activez ensuite la signature de code pour une fonction sans serveur en spécifiant la propriété `CodeSigningConfigArn` avec la fonction Amazon Resource Name (ARN) d'une ressource `AWS::Lambda::CodeSigningConfig`.
2. Signature de votre code – Utilisez la commande `sam package` ou `sam deploy` avec l'option `--signing-profiles`.

> **Note**
>
> Afin de signer avec succès votre code avec les commandes `sam package` ou `sam deploy`, la gestion des versions doit être activée pour le compartiment Amazon S3 que vous utilisez avec ces commandes. Si vous utilisez le compartiment Amazon S3 AWS SAM créé pour vous, le versionnement est activé automatiquement. Pour de plus amples informations sur la gestion des versions du compartiment Amazon S3 et des instructions sur l'activation de la gestion des versions sur un compartiment Amazon S3 que vous fournissez, consultez Utilisation de la gestion des versions dans les compartiments Amazon S3 dans le Guide du emploi Amazon Simple Storage Service.

Lorsque vous déployez une application sans serveur, Lambda effectue des vérifications de validation sur toutes les fonctions pour lesquelles vous avez activé la signature de code. Lambda effectue également des vérifications de validation sur toutes les couches dont ces fonctions dépendent. Pour de plus amples informations sur les vérifications de validation de Lambda, consultez Validation de signature dans le Guide du développeur AWS Lambda .

Exemple

Création d'un profil de signature

Pour créer un profil de signature, exécutez la commande suivante :

```
aws signer put-signing-profile --platform-id "AWSLambda-SHA384-ECDSA" --profile-
name MySigningProfile
```

Si la commande précédente réussit, vous voyez l'ARN du profil de signature renvoyé. Par exemple :

```
{
    "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"
}
```

Le champ `profileVersionArn` contient l'ARN à utiliser lors de la création de la configuration de signature de code.

Création d'une configuration de signature de code et activation de la signature de code pour une fonction

L'exemple de AWS SAM modèle suivant déclare une `AWS::Lambda::CodeSigningConfig`ressource et active la signature de code pour une fonction Lambda. Dans cet exemple, il existe un profil approuvé et les déploiements sont rejetés si les vérifications de signature échouent.

```
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"
```

Signature de votre code

Vous pouvez signer votre code lors de l'empaquetage ou du déploiement de votre application. Spécifiez l'option `--signing-profiles` avec la commande `sam package` ou `sam deploy`, comme illustré dans l'exemple de commandes suivant.

Signer votre code de fonction lors de l'empaquetage de votre application :

```
sam package --signing-profiles HelloWorld=MySigningProfile --s3-bucket amzn-s3-demo-
bucket --output-template-file packaged.yaml
```

Signer à la fois votre code de fonction et une couche dont votre fonction dépend, lors de l'empaquetage de votre application :

```
sam package --signing-profiles HelloWorld=MySigningProfile MyLayer=MySigningProfile --
s3-bucket amzn-s3-demo-bucket --output-template-file packaged.yaml
```

Signer votre code de fonction et une couche, puis effectuer un déploiement :

```
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**
>
> Afin de signer avec succès votre code avec les commandes `sam package` ou `sam deploy`, la gestion des versions doit être activée pour le compartiment Amazon S3 que vous utilisez

avec ces commandes. Si vous utilisez le compartiment Amazon S3 AWS SAM créé pour vous, le versionnement est activé automatiquement. Pour de plus amples informations sur la gestion des versions du compartiment Amazon S3 et des instructions sur l'activation de la gestion des versions sur un compartiment Amazon S3 que vous fournissez, consultez Utilisation de la gestion des versions dans les compartiments Amazon S3 dans le Guide du emploi Amazon Simple Storage Service.

Fournir des profils de signature avec **`sam deploy --guided`**

Lorsque vous exécutez la `sam deploy --guided` commande avec une application sans serveur configurée avec la signature de code, elle vous AWS SAM invite à fournir le profil de signature à utiliser pour la signature de code. Pour plus d'informations sur les invites `sam deploy --guided`, consultez sam deploy dans la référence des commandes de la CLI AWS SAM.

Valider les fichiers AWS SAM modèles

Validez vos modèles avec `sam validate`. Actuellement, cette commande valide le fait que le modèle fourni est un JSON/YAML valide. Comme avec la plupart des commandes CLI AWS SAM, elle recherche un fichier `template.[yaml|yml]` par défaut dans votre répertoire de travail actuel. Vous pouvez spécifier un autre modèle à l' file/location aide de l'`--template`option `-t` ou.

Exemple :

```
$ sam validate
<path-to-template>/template.yaml is a valid SAM Template
```

> **Note**
>
> La `sam validate` commande nécessite la configuration des AWS informations d'identification. Pour de plus amples informations, veuillez consulter Configuration de la CLI AWS SAM.

Créez votre application avec AWS SAM

Après avoir ajouté votre infrastructure sous forme de code (iAc) à votre AWS SAM modèle, vous serez prêt à commencer à créer votre application à l'aide de la sam build commande. Cette

commande crée des artefacts de construction à partir des fichiers du répertoire de votre projet d'application (c'est-à-dire votre fichier AWS SAM modèle, le code de l'application et tous les fichiers et dépendances spécifiques au langage applicable). Ces artefacts de build préparent votre application sans serveur pour les étapes ultérieures du développement de votre application, telles que les tests locaux et le déploiement AWS dans le cloud. Les tests et le déploiement utilisent des artefacts de construction comme entrées.

Vous pouvez l'utiliser sam build pour créer l'intégralité de votre application sans serveur. En outre, vous pouvez créer des versions personnalisées, par exemple des versions avec des fonctions, des couches ou des environnements d'exécution personnalisés spécifiques. Pour en savoir plus sur comment et pourquoi vous l'utilisezsam build, consultez les rubriques de cette section. Pour une introduction à l'utilisation de la commande `sam build`, consultez Initiation à la construction avec AWS SAM.

Rubriques

- Initiation à la construction avec AWS SAM
- Construit par défaut avec AWS SAM
- Personnalisez les builds avec AWS SAM

Initiation à la construction avec AWS SAM

Utilisez la AWS Serverless Application Model commande Command Line Interface (AWS SAMCLI) `sam build` pour préparer votre application sans serveur aux étapes suivantes de votre flux de travail de développement, telles que les tests locaux ou le déploiement sur le AWS Cloud. Cette commande crée un répertoire `.aws-sam` qui structure votre application dans un format et un emplacement que `sam local` et `sam deploy` nécessitent.

- Pour une introduction à la AWS SAMCLI, voirQu'est-ce que c'est AWS SAMCLI ?.
- Pour obtenir la liste des options de commande `sam build`, consultez sam build.
- Pour un exemple d'utilisation de `sam build` dans le cadre d'un flux de travail de développement classique, consultez Étape 2 : créer votre application.

> **Note**
>
> L'utilisation de `sam build` nécessite que vous commenciez avec les composants de base d'une application sans serveur sur votre machine de développement. Cela inclut un AWS

SAM modèle, un code de AWS Lambda fonction, ainsi que tous les fichiers et dépendances spécifiques au langage. Pour en savoir plus, veuillez consulter la section Créez votre application dans AWS SAM.

Rubriques

- Création d'applications avec sam build
- Test et déploiement au niveau local
- Bonnes pratiques
- Options pour sam build
- Résolution des problèmes
- Exemples
- En savoir plus

Création d'applications avec sam build

Avant de l'utiliser`sam build`, pensez à configurer les éléments suivants :

1. Fonctions et couches Lambda : la commande `sam build` permet de créer des fonctions et des couches Lambda. Pour en savoir plus sur les couches Lambda, consultez Création de couches Lambda dans AWS SAM.
2. Exécution Lambda : l'exécution fournit un environnement spécifique au langage qui exécute votre fonction dans un environnement d'exécution lorsqu'elle est invoquée. Vous pouvez configurer des environnements d'exécution natifs et personnalisés.
   a. Exécution native : créez vos fonctions Lambda dans une exécution Lambda prise en charge et créez vos fonctions pour utiliser une exécution Lambda native dans le AWS Cloud.
   b. Exécution personnalisée : créez vos fonctions Lambda à l'aide de n'importe quel langage de programmation et créez votre exécution à l'aide d'un processus personnalisé défini dans un makefile ou dans un créateur tiers tel que esbuild. Pour en savoir plus, veuillez consulter la section Création de fonctions Lambda avec des environnements d'exécution personnalisés dans AWS SAM.
3. Type de package Lambda : les fonctions Lambda peuvent être packagées dans les types de packages de déploiement Lambda suivants :
   a. .zip file archive : contient le code de votre application et ses dépendances.

b. Image du conteneur : contient le système d'exploitation de base, l'exécution, les extensions Lambda, le code de votre application et ses dépendances.

Ces paramètres d'application peuvent être configurés lors de l'initialisation d'une application à l'aide de `sam init`.

- Pour en savoir plus sur l'utilisation de `sam init`, consultez Créez votre application dans AWS SAM.
- Pour en savoir plus sur la configuration de ces paramètres dans votre application, consultez Construit par défaut avec AWS SAM.

Pour créer une application

1. `cd` à la racine de votre projet. Il s'agit du même emplacement que celui de votre AWS SAM modèle.

```
$ cd sam-app
```

2. Exécutez les commandes suivantes :

```
sam-app $ sam build <arguments> <options>
```

> **Note**
>
> Une option couramment utilisée est `--use-container`. Pour en savoir plus, veuillez consulter la section Création d'une fonction Lambda à l'intérieur d'un conteneur fourni.

Voici un exemple de réponse générée par la CLI AWS SAM :

```
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. La CLI AWS SAM crée un répertoire de création `.aws-sam`. Voici un exemple :

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

En fonction de la configuration de votre application, la CLI AWS SAM effectue les opérations suivantes :

1. Elle télécharge, installe et organise les dépendances dans le répertoire `.aws-sam/build`.
2. Elle prépare votre code Lambda. Il peut s'agir de compiler votre code, de créer des fichiers binaires exécutables et de créer des images de conteneurs.
3. Elle copie les artefacts de création dans le répertoire `.aws-sam`. Le format varie en fonction du type du package de l'application.
   a. Pour les types de packages .zip, les artefacts ne sont pas encore compressés afin de pouvoir être utilisés pour des tests locaux. La CLI AWS SAM compresse votre application lorsqu'elle utilise `sam deploy`.

   b. Pour les types de packages d'images de conteneurs, une image de conteneur est créée localement et référencée dans le fichier `.aws-sam/build.toml`.

4. Copie le AWS SAM modèle `.aws-sam` dans le répertoire et le modifie avec de nouveaux chemins de fichiers si nécessaire.

Les principaux composants qui constituent vos artefacts de création dans le répertoire `.aws-sam` sont les suivants :

- Le répertoire de création : contient vos fonctions et couches Lambda structurées indépendamment les unes des autres. La structure est donc unique pour chaque fonction ou couche du répertoire `.aws-sam/build`.
- Le AWS SAM modèle : modifié avec des valeurs mises à jour en fonction des modifications apportées au cours du processus de création.
- Le fichier build.toml : fichier de configuration contenant les paramètres de construction utilisés par le. AWS SAMCLI

Test et déploiement au niveau local

Lorsque vous effectuez des tests locaux avec `sam local` ou un déploiement avec `sam deploy`, la CLI AWS SAM effectue les opérations suivantes :

1. Il vérifie d'abord si un `.aws-sam` répertoire existe et si un AWS SAM modèle se trouve dans ce répertoire. Si ces conditions sont remplies, la CLI AWS SAM considère qu'il s'agit du répertoire racine de votre application.
2. Si ces conditions ne sont pas remplies, l' AWS SAMCLIemplacement d'origine de votre AWS SAM modèle est considéré comme le répertoire racine de votre application.

Lors du développement, si des modifications sont apportées à vos fichiers d'application originaux, exécutez `sam build` pour mettre à jour le répertoire `.aws-sam` avant de tester localement.

Bonnes pratiques

- Ne modifiez aucun code dans le répertoire `.aws-sam/build`. Mettez plutôt à jour votre code source original dans le dossier de votre projet et exécutez `sam build` pour mettre à jour le répertoire `.aws-sam/build`.

- Lorsque vous modifiez vos fichiers originaux, exécutez `sam build` pour mettre à jour le répertoire `.aws-sam/build`.
- Il se peut que vous souhaitiez que la CLI AWS SAM fasse référence au répertoire racine d'origine de votre projet plutôt qu'au répertoire `.aws-sam`, par exemple lors du développement et des tests avec `sam local`. Supprimez le `.aws-sam` répertoire ou le AWS SAM modèle dans le `.aws-sam` répertoire pour que le répertoire de votre projet d'origine soit AWS SAMCLI reconnu comme le répertoire racine du projet. Lorsque vous êtes prêt, exécutez à nouveau `sam build` pour créer le répertoire `.aws-sam`.
- Lorsque vous exécutez `sam build`, le répertoire `.aws-sam/build` est remplacé à chaque fois. Le répertoire `.aws-sam` ne l'est pas. Si vous souhaitez stocker des fichiers, tels que des journaux, stockez-les dans `.aws-sam` pour éviter qu'ils ne soient remplacés.

Options pour sam build

Création d'une seule ressource

Fournissez l'ID logique de la ressource pour ne créer que cette ressource. Voici un exemple :

```
$ sam build HelloWorldFunction
```

Pour créer une ressource d'une application ou d'une pile imbriquée, fournissez l'identifiant logique de l'application ou de la pile avec l'identifiant logique de la ressource en utilisant le format *<stack-logical-id>*/*<resource-logical-id>* :

```
$ sam build MyNestedStack/MyFunction
```

Création d'une fonction Lambda à l'intérieur d'un conteneur fourni

L'option `--use-container` télécharge une image de conteneur et l'utilise pour créer vos fonctions Lambda. Le conteneur local est ensuite référencé dans votre fichier `.aws-sam/build.toml`.

Cette option nécessite que Docker soit installé. Pour obtenir des instructions, veuillez consulter Installation de Docker.

Voici un exemple de cette commande :

```
$ sam build --use-container
```

Vous pouvez spécifier l'image de conteneur à utiliser avec l'option `--build-image`. Voici un exemple :

```
$ sam build --use-container --build-image amazon/aws-sam-cli-build-image-nodejs20.x
```

Pour spécifier l'image du conteneur à utiliser pour une seule fonction, fournissez l'ID logique de la fonction. Voici un exemple :

```
$ sam build --use-container --build-image Function1=amazon/aws-sam-cli-build-image-
python3.12
```

Transmettre des variables d'environnement au conteneur de création

Utilisez `--container-env-var` pour transmettre les variables d'environnement au conteneur de création. Voici un exemple :

```
$ sam build --use-container --container-env-var Function1.GITHUB_TOKEN=<token1> --
container-env-var GLOBAL_ENV_VAR=<global-token>
```

Pour transmettre des variables d'environnement à partir d'un fichier, utilisez l'option `--container-env-var-file`. Voici un exemple :

```
$ sam build --use-container --container-env-var-file <env.json>
```

Exemple de fichier `env.json` :

```
{
  "MyFunction1": {
    "GITHUB_TOKEN": "TOKEN1"
  },
  "MyFunction2": {
    "GITHUB_TOKEN": "TOKEN2"
  }
}
```

Accélérez la création d'applications contenant plusieurs fonctions

Lorsque vous exécutez `sam build` sur une application comportant plusieurs fonctions, la CLI AWS SAM crée chaque fonction une par une. Pour accélérer le processus de création, utilisez l'option `--parallel`. Cela crée toutes les fonctions et couches en même temps.

Voici un exemple de cette commande :

```
$ sam build —-parallel
```

Accélérez les temps de création en créant votre projet dans le dossier source

Pour les systèmes d'exécution et les méthodes de création pris en charge, vous pouvez utiliser l'option `--build-in-source` permettant de créer votre projet directement dans le dossier source. Par défaut, il est AWS SAM CLI compilé dans un répertoire temporaire, ce qui implique de copier le code source et les fichiers de projet. Avec`--build-in-source`, les AWS SAM CLI builds se trouvent directement dans votre dossier source, ce qui accélère le processus de compilation en supprimant le besoin de copier des fichiers dans un répertoire temporaire.

Pour obtenir une liste des systèmes d'exécution ainsi que des méthodes de création pris en charge, consultez `--build-in-source`.

Résolution des problèmes

Pour résoudre le problème AWS SAMCLI, voirRésolution des problèmes de la CLI AWS SAM.

Exemples

Création d'une application utilisant une exécution native et un type de package .zip

Pour cet exemple, consultez Tutoriel : Déployer une application Hello World avec AWS SAM.

Création d'une application utilisant une exécution native et un type de package d'images

Tout d'abord, nous exécutons `sam init` pour initialiser une nouvelle application. Au cours du flux interactif, nous sélectionnons le type de package `Image`. Voici un exemple :

```
$ 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

...
```

AWS SAMCLIInitialise une application et crée le répertoire de projet suivant :

```
sam-app
### README.md
### events
#   ### event.json
### hello-world
#   ### Dockerfile
#   ### app.mjs
#   ### package.json
#   ### tests
#       ### unit
#           ### test-handler.mjs
### samconfig.toml
### template.yaml
```

Ensuite, nous exécutons `sam build` pour créer notre application :

```
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
```

Création d'une application qui inclut un langage de programmation compilé

Dans cet exemple, nous créons une application contenant une fonction Lambda à l'aide de l'exécution Go.

Tout d'abord, nous initialisons une nouvelle application en utilisant `sam init` et configurons notre application pour qu'elle utilise 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
```

```
...
```

AWS SAMCLIInitialise ensuite l'application. Voici un exemple de la structure du répertoire de l'application :

```
sam-app
### Makefile
### README.md
### events
#   ### event.json
### hello-world
#   ### go.mod
#   ### go.sum
#   ### main.go
#   ### main_test.go
### samconfig.toml
### template.yaml
```

Nous référençons le fichier `README.md` pour les exigences de cette application.

```
...
## 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)
...
```

Ensuite, nous exécutons `sam local invoke` pour tester notre fonction. Cette commande provoque une erreur, car Go n'est pas installé sur notre machine locale :

```
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"}%
```

Ensuite, nous exécutons `sam build` pour créer notre application. Nous rencontrons une erreur car Go n'est pas installée sur notre machine locale :

```
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
```

Bien que nous puissions configurer notre machine locale pour créer correctement notre fonction, nous utilisons plutôt l'option `--use-container` avec `sam build`. AWS SAMCLITélécharge une image de conteneur, construit notre fonction en utilisant le natif GoModulesBuilder et copie le binaire obtenu `.aws-sam/build/HelloWorldFunction` dans notre répertoire.

```
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
```

Voici un exemple du répertoire `.aws-sam` :

```
.aws-sam
### build
#   ### HelloWorldFunction
#   #   ### hello-world
#   ### template.yaml
### build.toml
### cache
#   ### c860d011-4147-4010-addb-2eaa289f4d95
#       ### hello-world
### deps
```

Ensuite, nous exécutons `sam local invoke`. Notre fonction est appelée avec succès :

```
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"}%
```

En savoir plus

Pour en savoir plus sur l'utilisation de la commande `sam build`, reportez-vous à ce qui suit :

- Learning AWS SAM : sam build — Série « Learning AWS SAM » de Serverless Land surYouTube.
- Learning AWS SAM | sam build | E3 — Serverless Land « Learning AWS SAM » série surYouTube.
- AWS SAM build : comment il fournit des artefacts pour le déploiement (sessions avec SAM S2E8) — Sessions avec AWS SAM série activée. YouTube
- AWS SAM versions personnalisées : Comment utiliser les Makefiles pour personnaliser les versions dans SAM (S2E9) — Sessions avec séries activées. AWS SAM YouTube

Construit par défaut avec AWS SAM

Pour créer votre application sans serveur, utilisez la commande `sam build`. Cette commande rassemble également les artefacts de construction des dépendances de votre application et les place dans le format et l'emplacement appropriés pour les prochaines étapes, telles que les tests locaux, l'empaquetage et le déploiement.

Vous spécifiez les dépendances de votre application dans un fichier manifeste, tel que `requirements.txt` (Python) ou `package.json` (Node.js), ou à l'aide de la propriété `Layers` d'une ressource de fonction. La propriété `Layers` contient une liste de ressources de couche AWS Lambda dont dépend la fonction Lambda.

Le format des artefacts de construction de votre application dépend de la propriété `PackageType` de chaque fonction. Les options pour cette propriété sont :

- **`Zip`** – Une archive de fichiers .zip, comportant le code de votre application et ses dépendances. Si vous empaquetez le code sous la forme d'une archive de fichiers .zip, vous devez spécifier une exécution Lambda pour la fonction.
- **`Image`** : une image de conteneur incluant le système d'exploitation de base, l'exécution et les extensions, en plus du code de l'application et ses dépendances.

Pour de plus amples informations sur les types de packages Lambda, consultez Packages de déploiement Lambda dans le Guide du développeur AWS Lambda .

Rubriques

- Création d'une archive de fichier .zip
- Création d'une image de conteneur
- Fichier de variables d'environnement du conteneur.
- Accélérez les temps de création en créant votre projet dans le dossier source
- Exemples
- Fonctions du bâtiment en dehors de AWS SAM

Création d'une archive de fichier .zip

Pour créer votre application sans serveur en tant qu'archive de fichier .zip, déclarez `PackageType: Zip` pour votre fonction sans serveur.

AWS SAM construit votre application pour l'architecture que vous spécifiez. Si vous ne spécifiez pas d'architecture, AWS SAM utilise `x86_64` par défaut.

Si votre fonction Lambda dépend de packages qui ont compilé nativement des programmes, utilisez l'indicateur `--use-container`. Cet indicateur compile localement vos fonctions dans un conteneur qui se comporte comme un environnement Lambda, afin qu'elles soient au bon format lorsque vous les déployez dans le Cloud. AWS

Lorsque vous utilisez `--use-container` cette option, l'image du conteneur AWS SAM est extraite par défaut d'Amazon ECR Public. Si vous souhaitez extraire une image de conteneur d'un autre référentiel ou pour une version spécifique de AWS SAM CLI, vous pouvez utiliser l'`--build-image`option et fournir l'URI d'une autre image de conteneur. Voici deux exemples de commandes permettant de créer des applications à l'aide d'images de conteneur issues d'une version spécifique de la 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
```

Pour obtenir d'autres exemples de création d'une application d'archivage de fichiers .zip, reportez-vous à la section Exemples plus loin dans cette rubrique.

Création d'une image de conteneur

Pour créer votre application sans serveur en tant qu'image de conteneur, déclarez `PackageType: Image` pour votre fonction sans serveur. Vous devez également déclarer l'attribut de ressource `Metadata` avec les entrées suivantes :

`Dockerfile`

Nom du Dockerfile associé à la fonction Lambda.

`DockerContext`

Emplacement du fichier journal.

`DockerTag`

(Facultatif) Balise à appliquer à l'image créée.

`DockerBuildArgs`

Arguments de construction pour la création.

> **Important**
>
> Le AWS SAMCLI ne supprime ni ne masque les informations que vous incluez dans les arguments. `DockerBuildArgs` Nous vous recommandons vivement de ne pas utiliser cette section pour stocker des informations sensibles, telles que des mots de passe ou des secrets.

Voici un exemple de section d'attribut de ressource `Metadata` :

```
Metadata:
  Dockerfile: Dockerfile
  DockerContext: ./hello_world
  DockerTag: v1
```

Pour télécharger un exemple d'application configuré avec le type de `Image` package, consultez[Tutoriel : Déployer une application Hello World avec AWS SAM]. En réponse à l'invite demandant quel type de package vous souhaitez installer, choisissez `Image`.

> **Note**
>
> Si vous spécifiez une image de base multi-architecture dans votre Dockerfile, AWS SAM crée votre image de conteneur pour l'architecture de votre machine hôte. Pour créer une architecture différente, spécifiez une image de base qui utilise l'architecture cible spécifique.

Fichier de variables d'environnement du conteneur.

Pour fournir un fichier JSON qui contient des variables d'environnement pour le conteneur de construction, utilisez l'argument `--container-env-var-file` avec la commande `sam build`. Vous pouvez fournir une variable d'environnement unique qui s'applique à toutes les ressources sans serveur, ou des variables d'environnement différentes pour chaque ressource.

Format

Le format de transmission des variables d'environnement à un conteneur de construction dépend du nombre de variables d'environnement que vous fournissez pour vos ressources.

Pour fournir une variable d'environnement unique pour toutes les ressources, spécifiez un objet `Parameters` se présentant comme suit :

```
{
  "Parameters": {
    "GITHUB_TOKEN": "TOKEN_GLOBAL"
  }
}
```

Pour fournir différentes variables d'environnement pour chaque ressource, spécifiez des objets pour chaque ressource comme suit :

```
{
  "MyFunction1": {
    "GITHUB_TOKEN": "TOKEN1"
  },
  "MyFunction2": {
    "GITHUB_TOKEN": "TOKEN2"
  }
}
```

Enregistrez vos variables d'environnement en tant que fichier, par exemple nommé `env.json`. La commande suivante utilise ce fichier pour transmettre vos variables d'environnement au conteneur de construction :

```
sam build --use-container --container-env-var-file env.json
```

Priorité

- Les variables d'environnement que vous fournissez pour des ressources spécifiques seront prioritaires sur la variable d'environnement unique pour toutes les ressources.
- Les variables d'environnement que vous fournissez sur la ligne de commande seront prioritaires sur les variables d'environnement d'un fichier.

Accélérez les temps de création en créant votre projet dans le dossier source

Pour les systèmes d'exécution et les méthodes de création pris en charge, vous pouvez utiliser l'option `--build-in-source` permettant de créer votre projet directement dans le dossier source. Par défaut, il est AWS SAM CLI compilé dans un répertoire temporaire, ce qui implique de copier le code source et les fichiers de projet. Avec`--build-in-source`, les AWS SAM CLI builds se trouvent directement dans votre dossier source, ce qui accélère le processus de compilation en supprimant le besoin de copier des fichiers dans un répertoire temporaire.

Pour obtenir une liste des systèmes d'exécution ainsi que des méthodes de création pris en charge, consultez `--build-in-source`.

Exemples

Exemple 1 : archive de fichier .zip

Les commandes `sam build` suivantes créent une archive de fichier .zip :

```
# 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
```

Exemple 2 : image de conteneur

Le AWS SAM modèle suivant est créé sous forme d'image de conteneur :

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      PackageType: Image
      ImageConfig:
        Command: ["app.lambda_handler"]
    Metadata:
      Dockerfile: Dockerfile
      DockerContext: ./hello_world
      DockerTag: v1
```

Voici un exemple de 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"]
```

Exemple 3 : npm ci

Pour les applications Node.js, vous pouvez utiliser `npm ci` au lieu de `npm install` pour installer les dépendances. Pour utiliser `npm ci`, spécifiez `UseNpmCi: True` sous `BuildProperties` dans votre attribut de ressource `Metadata` de la fonction Lambda. Pour utiliser `npm ci`, votre application doit disposer d'un fichier `package-lock.json` ou `npm-shrinkwrap.json` présent dans le `CodeUri`pour votre fonction Lambda.

L'exemple suivant utilise `npm ci` pour installer des dépendances lorsque vous exécutez `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
```

Paquets parents Python

Pour les applications Python, vous pouvez conserver la structure de votre package pendant le processus de construction afin de permettre des importations absolues. Pour préserver la structure du package, spécifiez `ParentPackageMode` under `BuildProperties` dans l'attribut de `Metadata` ressource de votre fonction Lambda.

L'exemple suivant préserve la structure `app` du package lorsque vous exécutez `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
```

Avec cette configuration, votre code peut utiliser des importations absolues `from app.utils import logger` au lieu d'importations relatives telles que`from .utils import logger`.

Fonctions du bâtiment en dehors de AWS SAM

Par défaut, lorsque vous exécutez`sam build`, AWS SAM crée toutes les ressources de vos fonctions. Les autres options sont les suivantes :

- Construire toutes les ressources fonctionnelles en dehors de AWS SAM — Si vous créez toutes vos ressources fonctionnelles manuellement ou via un autre outil, cela n'`sam build`est pas obligatoire. Vous pouvez ignorer `sam build` et passer à l'étape suivante de votre processus, par exemple effectuer des tests locaux ou déployer votre application.
- Création de ressources fonctionnelles en dehors de AWS SAM : si vous AWS SAM souhaitez créer certaines de vos ressources fonctionnelles tout en créant d'autres ressources fonctionnelles en dehors de AWS SAM celles-ci, vous pouvez le spécifier dans votre AWS SAM modèle.

Créez des ressources fonctionnelles en dehors de AWS SAM

Pour AWS SAM ignorer une fonction lors de son utilisation`sam build`, configurez les éléments suivants dans votre AWS SAM modèle :

1. Ajoutez la propriété de métadonnées `SkipBuild: True` à votre fonction.
2. Spécifiez le chemin d'accès à vos ressources de fonction créées.

Voici un exemple, avec `TestFunction` configuré pour être ignoré. Ses ressources créées se trouvent à cet emplacement : `built-resources/TestFunction.zip`.

```
TestFunction:
```

```
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: built-resources/TestFunction.zip
    Handler: TimeHandler::handleRequest
    Runtime: java11
  Metadata:
    SkipBuild: True
```

Maintenant, lorsque vous exécuterezsam build, vous AWS SAM ferez ce qui suit :

1. AWS SAM ignorera les fonctions configurées avec`SkipBuild: True`.
2. AWS SAM créera toutes les autres ressources fonctionnelles et les mettra en cache dans le répertoire de `.aws-sam` construction.
3. Pour les fonctions ignorées, leur modèle dans le répertoire de création `.aws-sam` est automatiquement mis à jour pour référencer le chemin spécifié vers les ressources de vos fonctions créées.

   Voici un exemple du modèle mis en cache pour `TestFunction` dans le répertoire de création `.aws-sam` :

   ```
   TestFunction:
     Type: AWS::Serverless::Function
     Properties:
       CodeUri: ../../built-resources/TestFunction.zip
       Handler: TimeHandler::handleRequest
       Runtime: java11
     Metadata:
       SkipBuild: True
   ```

Personnalisez les builds avec AWS SAM

Vous pouvez personnaliser votre build pour inclure des fonctions Lambda ou des couches Lambda spécifiques. Une fonction est une ressource que vous pouvez appeler pour exécuter votre code dans Lambda. Une couche Lambda vous permet d'extraire le code d'une fonction Lambda qui peut ensuite être réutilisé dans plusieurs fonctions Lambda. Vous pouvez choisir de personnaliser votre build avec des fonctions Lambda spécifiques lorsque vous souhaitez vous concentrer sur le développement et le déploiement de fonctions sans serveur individuelles sans la complexité liée à la gestion de dépendances ou de ressources partagées. En outre, vous pouvez choisir de créer une couche

Lambda pour vous aider à réduire la taille de vos packages de déploiement, à séparer la logique des fonctions de base des dépendances et à partager les dépendances entre plusieurs fonctions.

Les rubriques de cette section explorent certaines des différentes manières de créer des fonctions Lambda. AWS SAM Cela inclut la création de fonctions Lambda avec les environnements d'exécution des clients et la création de couches Lambda. Les environnements d'exécution personnalisés vous permettent d'installer et d'utiliser un langage qui n'est pas répertorié dans les environnements d'exécution Lambda du Guide du développeur. AWS Lambda Cela vous permet de créer un environnement d'exécution spécialisé pour exécuter des fonctions et des applications sans serveur. La création de couches Lambda uniquement (au lieu de créer l'intégralité de votre application) peut vous être bénéfique de plusieurs manières. Cela peut vous aider à réduire la taille de vos packages de déploiement, à séparer la logique des fonctions de base des dépendances et à partager les dépendances entre plusieurs fonctions.

Pour plus d'informations sur les fonctions, consultez les concepts Lambda dans le Guide du AWS Lambda développeur.

Rubriques

- Création de fonctions Lambda dans Node.js avec esbuild AWS SAM
- Création de fonctions Lambda .NET avec une compilation native AOT dans AWS SAM
- Création de fonctions Rust Lambda avec in Cargo Lambda AWS SAM
- Création de fonctions Lambda en Python avec in uv AWS SAM
- Création de fonctions Lambda avec des environnements d'exécution personnalisés dans AWS SAM
- Création de couches Lambda dans AWS SAM

Création de fonctions Lambda dans Node.js avec esbuild AWS SAM

Pour créer et empaqueter AWS Lambda les fonctions Node.js, vous pouvez les utiliser AWS SAMCLI avec le JavaScript bundler esbuild. Le bundler esbuild prend en charge les fonctions Lambda que vous écrivez. TypeScript

Pour créer une fonction Lambda Node.js avec esbuild, ajoutez un objet `Metadata` à votre `AWS:Serverless::Function` ressource et spécifiez `esbuild` pour `BuildMethod`. Lorsque vous exécutez la sam build commande, AWS SAM utilise esbuild pour regrouper le code de votre fonction Lambda.

Propriétés de métadonnées

L'objet `Metadata` prend en charge les propriétés suivantes pour esbuild.

BuildMethod

Spécifie le bundler de votre application. La seule valeur prise en charge est `esbuild`.

BuildProperties

Spécifie les propriétés de création du code de votre fonction Lambda.

L'objet `BuildProperties` prend en charge les propriétés suivantes pour esbuild. Toutes les propriétés sont facultatives. Par défaut, AWS SAM utilise votre gestionnaire de fonctions Lambda comme point d'entrée.

EntryPoints

Spécifie les points d'entrée de votre application.

Externe

Spécifie la liste des packages à omettre de la création. Pour plus d'informations, veuillez consulter la section Externe sur le site Web esbuild (langue française non garantie).

Format

Spécifie le format de sortie des JavaScript fichiers générés dans votre application. Pour plus d'informations, consultez la section Format du site Web esbuild.

Chargeur

Spécifie la liste des configurations de chargement des données pour un type de fichier donné.

MainFields

Spécifie les champs `package.json` à essayer d'importer lors de la résolution d'un package. La valeur par défaut est `main,module`.

Réduire

Spécifie s'il faut réduire le code de sortie groupé. La valeur par défaut est `true`.

OutExtension

Personnalisez l'extension des fichiers générés par esbuild. Pour plus d'informations, veuillez consulter Out extension sur le site Web esbuild.

Carte source

Spécifie si le bundler produit un fichier de carte source. La valeur par défaut est `false`.

Lorsqu'il a la valeur `true`, `NODE_OPTIONS: --enable-source-maps` est ajouté aux variables d'environnement de la fonction Lambda, et une carte source est générée et incluse dans la fonction.

Sinon, lorsque `NODE_OPTIONS: --enable-source-maps` est inclus dans les variables d'environnement de la fonction, `Sourcemap` est automatiquement défini sur `true`.

En cas de conflit, `Sourcemap: false` a la priorité sur `NODE_OPTIONS: --enable-source-maps`.

> **Note**
>
> Par défaut, Lambda chiffre toutes les variables d'environnement au repos avec AWS Key Management Service (AWS KMS). Lorsque vous utilisez des cartes sources, pour que le déploiement réussisse, le rôle d'exécution de votre fonction doit avoir l'autorisation d'effectuer l'action `kms:Encrypt`.

SourcesContent

Indique s'il faut inclure votre code source dans votre fichier de carte source. Configurez cette propriété lorsque `Sourcemap` est définie sur `'true'`.

- Spécifiez `SourcesContent: 'true'` pour inclure tout le code source.
- Spécifiez `SourcesContent: 'false'` pour exclure tout le code source. Cela permet de réduire la taille des fichiers de cartes sources, ce qui est utile en production en réduisant les temps de démarrage. Cependant, le code source ne sera pas disponible dans le débogueur.

La valeur par défaut est `SourcesContent: true`.

Pour plus d'informations, consultez la section Contenu des sources du site web esbuild.

Cible

Spécifie la ECMAScript version cible. La valeur par défaut est `es2020`.

TypeScript Exemple de fonction Lambda

L'exemple d'extrait de AWS SAM modèle suivant utilise esbuild pour créer une fonction Lambda Node.js à partir du code dans. 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>"
```

Création de fonctions Lambda .NET avec une compilation native AOT dans AWS SAM

Créez et empaquetez vos AWS Lambda fonctions .NET 8 avec le AWS Serverless Application Model (AWS SAM), en utilisant la compilation native Ahead-of-Time (AOT) pour améliorer les temps de démarrage à AWS Lambda froid.

Rubriques

- Présentation de .NET 8 Native AOT
- Utilisation AWS SAM avec vos fonctions Lambda .NET 8
- Installer les prérequis
- Définissez les fonctions Lambda .NET 8 dans votre modèle AWS SAM
- Créer votre application à l'aide de la CLI AWS SAM
- En savoir plus

Présentation de .NET 8 Native AOT

Historiquement, les fonctions Lambda .NET ont eu des temps de démarrage à froid affectant l'expérience utilisateur, la latence du système et les coûts d'utilisation de vos applications sans serveur. Avec la compilation .NET Native AOT, vous pouvez améliorer les temps de démarrage à froid de vos fonctions Lambda. Pour en savoir plus sur l'AOT natif pour .NET 8, voir Utilisation de l'AOT natif dans le référentiel GitHub Dotnet.

Utilisation AWS SAM avec vos fonctions Lambda .NET 8

Procédez comme suit pour configurer vos fonctions Lambda .NET 8 avec le AWS Serverless Application Model ()AWS SAM :

- Installez les prérequis sur votre machine de développement.
- Définissez les fonctions Lambda .NET 8 dans votre AWS SAM modèle.
- Créez votre application à l'aide du AWS SAMCLI.

Installer les prérequis

Les prérequis sont les suivants :

- Le AWS SAMCLI
- La CLI .NET Core
- L'outil global .NET Core d'Amazon.Lambda.Tools
- Docker

Installer la CLI AWS SAM

1. Pour vérifier si la CLI AWS SAM est déjà installée, procédez comme suit :

```
sam --version
```

2. Pour installer le AWS SAMCLI, voirInstaller la CLI AWS SAM.
3. Pour mettre à niveau une version installée du AWS SAMCLI, voirMise à niveau de la CLI AWS SAM en cours.

Installer la CLI .NET Core

1. Pour télécharger et installer la CLI .NET Core, veuillez consulter la rubrique Télécharger .NET (langue française non garantie) du site web Microsoft.
2. Pour plus d'informations sur la CLI .NET Core, veuillez consulter la rubrique Interface de ligne de commande .NET Core dans le Guide du développeur AWS Lambda .

Installer l'outil global .NET Core d'Amazon.Lambda.Tools

1. Exécutez la commande suivante :

```
dotnet tool install -g Amazon.Lambda.Tools
```

2. Si cet outil est déjà installé, vous pouvez vérifier qu'il s'agit de la dernière version avec la commande suivante :

```
dotnet tool update -g Amazon.Lambda.Tools
```

3. Pour plus d'informations sur l'outil global .NET Core d'Amazon.Lambda.Tools, consultez les extensions AWS pour le référentiel .NET CLI sur. GitHub

Installer Docker

- Docker doit être installé pour permettre la création avec la compilation anticipée native. Pour obtenir des instructions d’installation, consultez Installation de Docker pour une utilisation avec la CLI AWS SAM.

Définissez les fonctions Lambda .NET 8 dans votre modèle AWS SAM

Pour définir un. NET8 Fonction Lambda dans votre AWS SAM modèle, procédez comme suit :

1. Exécutez la commande suivante depuis le répertoire de départ de votre choix :

```
sam init
```

2. Sélectionnez `AWS Quick Start Templates` pour choisir un modèle de départ.
3. Choisissez le modèle `Hello World Example`.
4. Choisissez de ne pas utiliser le moteur d'exécution et le type de package les plus populaires en saisissantn.
5. Pour l'exécution, choisissez`dotnet8`.
6. Pour le type de package, choisissez`Zip`.
7. Pour votre modèle de démarrage, choisissez`Hello World Example using native AOT`.

Installer Docker

- Docker doit être installé pour permettre la création avec la compilation anticipée native. Pour obtenir des instructions d'installation, consultez Installation de Docker pour une utilisation avec la CLI AWS SAM.

```
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**
>
> Lorsque la `Event` propriété de an `AWS::Serverless::Function` est définie sur`Api`, mais que la `RestApiId` propriété n'est pas spécifiée, AWS SAM la `AWS::ApiGateway::RestApi` CloudFormation ressource est générée.

Créer votre application à l'aide de la CLI AWS SAM

Depuis le répertoire racine de votre projet, exécutez la `sam build` commande pour commencer à créer votre application. Si la `PublishAot` propriété a été définie dans votre fichier de projet .NET 8, elle AWS SAMCLI sera compilée avec la compilation native AOT. Pour en savoir plus sur la propriété `PublishAot`, veuillez consulter la rubrique Déploiement anticipé natif (langue française non garantie) dans la documentation .NET de Microsoft.

Pour créer votre fonction, la CLI AWS SAM invoque la CLI .NET Core qui utilise l'outil global .NET Core d'Amazon.Lambda.Tools.

> **Note**
>
> Lors de la création, si un fichier `.sln` existe dans le même répertoire ou dans le répertoire parent de votre projet, le répertoire contenant le fichier `.sln` sera monté dans le conteneur. Si aucun fichier `.sln` n'est trouvé, seul le dossier du projet est monté. Par conséquent, si vous créez une application multi-projets, assurez-vous que le fichier `.sln` est bien localisé.

En savoir plus

Pour plus d'informations sur la création de fonctions Lambda .NET 8, consultez Présentation du moteur d'exécution .NET 8 pour. AWS Lambda

Pour obtenir la référence de la commande sam build, consultez sam build.

Création de fonctions Rust Lambda avec in Cargo Lambda AWS SAM

Cette fonctionnalité est en version préliminaire AWS SAM et est sujette à modification.

Utilisez l'interface de ligne de AWS Serverless Application Model commande (AWS SAMCLI) avec vos AWS Lambda fonctions Rust.

Rubriques

- Conditions préalables
- Configuration AWS SAM à utiliser avec les fonctions Rust Lambda
- Exemples

Conditions préalables

Langage Rust

Pour installer Rust, consultez la section Installer Rust sur le site web du langage Rust.

Cargo Lambda

La CLI AWS SAM nécessite l'installation de Cargo Lambda, une sous-commande pour Cargo. Pour les instructions d'installation, consultez la rubrique Installation dans la documentation Cargo Lambda.

Docker

La création et le test de fonctions Lambda Rust nécessitent Docker. Pour obtenir des instructions d'installation, consultez Installation de Docker.

Activer la fonctionnalité bêta de la CLI AWS SAM

Cette fonctionnalité étant en cours de prévisualisation, vous devez l'activer en utilisant l'une des méthodes suivantes :

1. Utiliser la variable d'environnement : `SAM_CLI_BETA_RUST_CARGO_LAMBDA=1`.
2. Ajoutez ce qui suit à votre fichier `samconfig.toml` :

```
[default.build.parameters]
beta_features = true
[default.sync.parameters]
beta_features = true
```

3. Utilisez l'option `--beta-features` lorsque vous utilisez une commande de la CLI AWS SAM prise en charge. Par exemple :

```
$ sam build --beta-features
```

4. Choisissez l'option y lorsque la CLI AWS SAM vous y invite. Voici un exemple :

```
$ sam build
Starting Build use cache
Build method "rust-cargolambda" 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
```

Configuration AWS SAM à utiliser avec les fonctions Rust Lambda

Étape 1 : Configuration de votre AWS SAM modèle

Configurez votre AWS SAM modèle avec les éléments suivants :

- Binaire : facultatif. Spécifiez quand votre modèle contient plusieurs fonctions Lambda Rust.
- BuildMethod – `rust-cargolambda`.
- CodeUri— chemin d'accès à votre `Cargo.toml` fichier.
- Gestionnaire : `bootstrap`.
- Exécution : `provided.al2`.

Pour en savoir plus sur les environnements d'exécution personnalisés, consultez la section AWS Lambda Runtimes personnalisés dans le manuel du AWS Lambda développeur.

Voici un exemple de AWS SAM modèle configuré :

```
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.al2
...
```

Étape 2 : utiliser la CLI AWS SAM avec votre fonction Lambda Rust

Utilisez n'importe quelle AWS SAMCLI commande avec votre AWS SAM modèle. Pour de plus amples informations, veuillez consulter AWS SAM CLI.

Exemples

Exemple Hello World

Dans cet exemple, nous créons l'exemple d'application Hello World en utilisant Rust comme notre exécution.

Tout d'abord, nous initialisons une nouvelle application sans serveur en utilisant `sam init`. Au cours du flux interactif, nous sélectionnons l'application Hello World et choisissons l'exécution Rust.

```
$ 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 - ruby3.3
        21 - ruby3.2
        22 - rust (provided.al2)
        23 - rust (provided.al2023)
Runtime: 22
```

```
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.al2)
    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
```

Voici la structure de notre application Hello World :

```
hello-rust
### README.md
### events
#   ### event.json
```

```
### rust_app
#   ### Cargo.toml
#   ### src
#       ### main.rs
### samconfig.toml
### template.yaml
```

Dans notre AWS SAM modèle, notre Rust fonction est définie comme suit :

```
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.al2
      Architectures:
        - x86_64
      Events:
        HelloWorld:
          Type: Api
            Path: /hello
            Method: get
```

Ensuite, nous exécutons `sam build` pour créer notre application et préparer son déploiement. La CLI AWS SAM crée un répertoire `.aws-sam` et organise nos artefacts de création. Notre fonction est construite en utilisant Cargo Lambda et stockée sous forme de binaire exécutable à l'emplacement suivant : `.aws-sam/build/HelloWorldFunction/bootstrap`.

> **Note**
>
> Si vous envisagez d'exécuter la sam local invoke commande sous macOS, vous devez créer des fonctions différentes avant de l'invoquer. Pour ce faire, utilisez la commande suivante :
>
> - SAM_BUILD_MODE=debug sam build

Cette commande n'est nécessaire que si des tests locaux doivent être effectués. Cela n'est pas recommandé lors de la création en vue du déploiement.

```
hello-rust$ sam build
Starting Build use cache
Build method "rust-cargolambda" 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-rust/rust_app runtime: provided.al2 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
```

Ensuite, nous déployons notre application en utilisant `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
```

Pour tester, nous pouvons appeler notre fonction Lambda à l'aide du point de terminaison de l'API.

```
$ curl https://ggdxec9le9.execute-api.us-west-2.amazonaws.com/Prod/hello/
Hello World!%
```

Pour tester notre fonction localement, nous devons d'abord nous assurer que la propriété `Architectures`de notre fonction correspond à notre ordinateur local.

```
...
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.al2
      Architectures:
        - arm64
...
```

Comme nous avons modifié notre architecture en passant de `x86_64` à `arm64` dans cet exemple, nous exécutons `sam build` pour mettre à jour nos artefacts de création. Nous exécutons ensuite `sam local invoke` pour appeler notre fonction localement.

```
hello-rust$ sam local invoke
Invoking bootstrap (provided.al2)
Local image was not found.
Removing rapid images for repo public.ecr.aws/sam/emulation-provided.al2
Building
 image.....................................................................................
Using local image: public.ecr.aws/lambda/provided:al2-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
```

Projet de fonction Lambda unique

Voici un exemple d'application sans serveur contenant une fonction Lambda Rust.

Structure du répertoire du projet :

```
.
### Cargo.lock
### Cargo.toml
### src
#   ### main.rs
```

```
### template.yaml
```

AWS SAM modèle :

```
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.al2
...
```

Projet de fonctions Lambda multiples

Voici un exemple d'application sans serveur contenant plusieurs fonctions Lambda Rust.

Structure du répertoire du projet :

```
.
### Cargo.lock
### Cargo.toml
### src
#   ### function_a.rs
#   ### function_b.rs
### template.yaml
```

AWS SAM modèle :

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  FunctionA:
    Type: AWS::Serverless::Function
    Metadata:
      BuildMethod: rust-cargolambda
```

```
      BuildProperties:
        Binary: function_a
    Properties:
      CodeUri: ./
      Handler: bootstrap
      Runtime: provided.al2
  FunctionB:
    Type: AWS::Serverless::Function
    Metadata:
      BuildMethod: rust-cargolambda
      BuildProperties:
        Binary: function_b
    Properties:
      CodeUri: ./
      Handler: bootstrap
      Runtime: provided.al2
```

Fichier `Cargo.toml` :

```
[package]
name = "test-handler"
version = "0.1.0"
edition = "2021"

[dependencies]
lambda_runtime = "0.6.0"
serde = "1.0.136"
tokio = { version = "1", features = ["macros"] }
tracing = { version = "0.1", features = ["log"] }
tracing-subscriber = { version = "0.3", default-features = false, features = ["fmt"] }

[[bin]]
name = "function_a"
path = "src/function_a.rs"

[[bin]]
name = "function_b"
path = "src/function_b.rs"
```

Création de fonctions Lambda en Python avec in uv AWS SAM

Cette fonctionnalité est en version préliminaire AWS SAM et est sujette à modification.

Utilisez l'interface de ligne de AWS Serverless Application Model commande (AWS SAMCLI) avecuv, un installateur et un résolveur rapides de paquets Python, pour créer vos AWS Lambda fonctions Python.

Rubriques

- Conditions préalables
- Configuration AWS SAM à utiliser avec les fonctions Lambda en Python et uv
- Exemples

Conditions préalables

Python

Pour installer Python, consultez la section Télécharger Python sur le site Web de Python.

uv

AWS SAMCLINécessite l'installation d'uvun programme d'installation et d'un résolveur de paquets Python extrêmement rapides. Pour les instructions d'installation, consultez la rubrique Installation dans la documentation uv.

Activer la fonctionnalité bêta de la CLI AWS SAM

Cette fonctionnalité étant en cours de prévisualisation, vous devez l'activer en utilisant l'une des méthodes suivantes :

1. Utiliser la variable d'environnement : `SAM_CLI_BETA_PYTHON_UV=1`.
2. Ajoutez ce qui suit à votre fichier `samconfig.toml` :

```
[default.build.parameters]
beta_features = true
[default.sync.parameters]
beta_features = true
```

3. Utilisez l'option `--beta-features` lorsque vous utilisez une commande de la CLI AWS SAM prise en charge. Par exemple :

```
$ sam build --beta-features
```

4. Choisissez l'option y lorsque la CLI AWS SAM vous y invite. Voici un exemple :

```
$ 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
```

Configuration AWS SAM à utiliser avec les fonctions Lambda en Python et uv

Étape 1 : Configuration de votre AWS SAM modèle

Configurez votre AWS SAM modèle avec les éléments suivants :

- BuildMethod – `python-uv`.
- CodeUri— chemin d'accès au répertoire de votre code de fonction contenant `pyproject.toml` ou`requirements.txt`.
- Handler — votre gestionnaire de fonctions (par exemple,`app.lambda_handler`).
- Runtime — Version d'exécution de Python (par exemple,`python3.12`).

Voici un exemple de AWS SAM modèle configuré :

```
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
...
```

Exemples

Exemple Hello World

Dans cet exemple, nous créons un exemple d'application Hello World en utilisant Python uv comme gestionnaire de packages.

uvpeut utiliser l'un `pyproject.toml` ou l'autre ou `requirements.txt` pour lire les dépendances. Si les deux sont fournis, `sam build` je lirai depuis `requirements.txt` pour les dépendances.

Voici la structure de notre application Hello World :

```
hello-python-uv
### README.md
### events
#   ### event.json
### hello_world
#   ### __init__.py
#   ### app.py
#   ### pyproject.toml
### samconfig.toml
### template.yaml
```

Fichier `pyproject.toml` :

```
[project]
name = "my-function"
version = "0.1.0"
requires-python = ">=3.12"
dependencies = [
    "requests>=2.31.0",
    "boto3>=1.28.0",
]
```

Dans notre AWS SAM modèle, notre fonction Python est définie comme suit :

```
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
```

Ensuite, nous exécutons `sam build` pour créer notre application et préparer son déploiement. La CLI AWS SAM crée un répertoire `.aws-sam` et organise nos artefacts de création. Nos dépendances fonctionnelles sont installées à uv l'aide de `.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**
>
> La méthode de `python-uv` construction est configurée par fonction dans la `Metadata` section. Chaque fonction de votre modèle peut utiliser une méthode de construction différente, ce qui vous permet de mélanger des fonctions uv `pip` basées avec des fonctions basées dans le même AWS SAM modèle. Si aucune méthode de construction n'est spécifiée, elle `pip` est utilisée par défaut.

Création de fonctions Lambda avec des environnements d'exécution personnalisés dans AWS SAM

Vous pouvez utiliser la commande `sam build` pour créer des exécutions personnalisées requises pour votre fonction Lambda. Vous déclarez votre fonction Lambda pour utiliser une exécution personnalisée en spécifiant `Runtime: provided` pour la fonction.

Pour créer une exécution personnalisée, déclarez l'attribut de ressource `Metadata` avec une entrée `BuildMethod: makefile`. Vous fournissez un makefile personnalisé, où vous déclarez une cible de génération du formulaire `build-`*`function-logical-id`* qui contient les commandes de création de votre exécution. Votre makefile est responsable de la compilation de l'exécution personnalisée si nécessaire, et de la copie des artefacts de construction dans l'emplacement approprié requis pour les étapes suivantes de votre flux. L'emplacement du makefile est spécifié par la propriété `CodeUri` de la ressource de fonction, et doit être nommé `Makefile`.

Exemples

Exemple 1 : exécution personnalisée pour une fonction écrite en Rust

> **Note**
>
> Nous vous recommandons de créer des fonctions Lambda avec Cargo Lambda. Pour en savoir plus, veuillez consulter la section Création de fonctions Rust Lambda avec in Cargo Lambda AWS SAM.

Le AWS SAM modèle suivant déclare une fonction qui utilise un environnement d'exécution personnalisé pour une fonction Lambda écrite en Rust et demande d'exécuter les `sam build` commandes pour la `build-HelloRustFunction` cible de construction.

```
Resources:
  HelloRustFunction:
    Type: AWS::Serverless::Function
    Properties:
      FunctionName: HelloRust
      Handler: bootstrap.is.real.handler
      Runtime: provided
      MemorySize: 512
      CodeUri: .
    Metadata:
      BuildMethod: makefile
```

Le makefile suivant contient la cible de génération et les commandes qui seront exécutées. Notez que la propriété `CodeUri` est définie sur `.`. Le makefile doit donc se trouver dans le répertoire racine du projet (c'est-à-dire dans le même répertoire que le fichier de modèle de l'application AWS SAM). Le nom de fichier doit être `Makefile`.

```
build-HelloRustFunction:
 cargo build --release --target x86_64-unknown-linux-musl
 cp ./target/x86_64-unknown-linux-musl/release/bootstrap $(ARTIFACTS_DIR)
```

Pour de plus amples informations sur la configuration de votre environnement de développement afin d'exécuter la commande `cargo build` dans le `makefile` précédent, consultez l'article de blog Exécution Rust pour AWS Lambda.

Exemple 2 : générateur de fichiers Makefile pour Python3.12 (alternative à l'utilisation du générateur groupé)

Vous pouvez utiliser une bibliothèque ou un module qui n'est pas inclus dans un générateur groupé. Cet exemple montre un AWS SAM modèle pour un environnement d'exécution Python3.12 avec un générateur de makefile.

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: hello_world/
      Handler: app.lambda_handler
      Runtime: python3.12
    Metadata:
      BuildMethod: makefile
```

Le makefile suivant contient la cible de génération et les commandes qui seront exécutées. Notez que la propriété `CodeUri` est définie sur `hello_world`. Le makefile doit donc se trouver dans la racine du répertoire `hello_world`, et le nom de fichier doit être `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
```

Création de couches Lambda dans AWS SAM

Vous pouvez l'utiliser AWS SAM pour créer des couches Lambda personnalisées. Les couches Lambda vous permettent d'extraire le code d'une fonction Lambda qui peut ensuite être réutilisé dans plusieurs fonctions Lambda. La création de couches Lambda uniquement (au lieu de créer l'intégralité de votre application) peut vous être bénéfique de plusieurs manières. Cela peut vous aider à réduire la taille de vos packages de déploiement, à séparer la logique des fonctions de base des dépendances et à partager les dépendances entre plusieurs fonctions. Pour davantage d'informations sur les couches, consultez Couches Lambda AWS dans le Guide du développeur AWS Lambda .

Comment créer une couche Lambda dans AWS SAM

> **Note**
>
> Avant de créer une couche Lambda, vous devez d'abord écrire une couche Lambda dans votre modèle. AWS SAM Pour obtenir des informations et des exemples à ce sujet, consultezAméliorez l'efficacité en utilisant les couches Lambda avec AWS SAM.

Pour créer une couche personnalisée, déclarez-la dans votre fichier modèle AWS Serverless Application Model (AWS SAM) et incluez une section d'attribut de `Metadata` ressource avec une `BuildMethod` entrée. Les valeurs valides pour `BuildMethod` sont des identifiants pour une exécution AWS Lambda, ou `makefile`. Incluez une entrée `BuildArchitecture` pour préciser les architectures de jeux d'instructions prises en charge par votre couche. Les valeurs valides pour `BuildArchitecture` sont Architectures du jeu d'instructions Lambda.

Si vous spécifiez `makefile`, vous fournissez un makefile personnalisé, où vous déclarez une cible de génération du formulaire `build-`*`layer-logical-id`* qui contient les commandes de création

de votre couche. Votre makefile est responsable de la compilation de la couche si nécessaire, et de la copie des artefacts de construction dans l'emplacement approprié requis pour les étapes suivantes de votre flux. L'emplacement du makefile est spécifié par la propriété `ContentUri` de la ressource de couche, et doit être nommé `Makefile`.

> **Note**
>
> Lorsque vous créez une couche personnalisée, AWS Lambda cela dépend des variables d'environnement pour trouver votre code de couche. Les exécutions Lambda incluent des chemins dans le répertoire `/opt` dans lequel votre code de couche est copié. La structure de dossier d'artefact de construction de votre projet doit correspondre à la structure de dossier attendue de l'exécution pour que votre code de couche personnalisé puisse être trouvé. Par exemple, pour Python, vous pouvez placer votre code dans le sous-répertoire `python/`. Pour NodeJS, vous pouvez placer votre code dans le sous-répertoire `nodejs/node_modules/`.
> Pour de plus amples informations, consultez Inclusion de dépendances de bibliothèques dans une couche dans le Guide du développeur AWS Lambda .

Voici un exemple de section d'attribut de ressource `Metadata`.

```
Metadata:
  BuildMethod: python3.12
  BuildArchitecture: arm64
```

> **Note**
>
> Si vous n'incluez pas la section `Metadata` des attributs de ressource, AWS SAM cela ne crée pas la couche. Au lieu de cela, il copie les artefacts de construction à partir de l'emplacement spécifié dans la propriété `CodeUri` de la ressource de couche. Pour plus d'informations, consultez la propriété ContentUri du type de ressource `AWS::Serverless::LayerVersion`.

Lorsque vous incluez la section `Metadata` des attributs de ressource, vous pouvez utiliser la `sam build` commande pour créer la couche, à la fois en tant qu'objet indépendant ou en tant que dépendance d'une AWS Lambda fonction.

- En tant qu'objet indépendant. Vous pouvez souhaiter créer uniquement l'objet de la couche, par exemple si vous testez localement une modification de code de la couche et que vous n'avez pas besoin de créer l'ensemble de votre application. Pour créer la couche indépendamment, spécifiez la ressource de couche à l'aide de la commande `sam build` *`layer-logical-id`*.
- En tant que dépendance d'une fonction Lambda. Lorsque vous incluez l'ID logique d'une couche dans la propriété `Layers` d'une fonction Lambda dans le même fichier modèle AWS SAM , la couche sera une dépendance de cette fonction Lambda. Lorsque cette couche inclut également une section d'attribut de ressource `Metadata` avec une entrée `BuildMethod`, vous créez la couche soit en créant la totalité de l'application à l'aide de l'option `sam build` ou en spécifiant la ressource de fonction avec la commande `sam build` *`function-logical-id`*.

Exemples

Exemple de modèle 1 : création d'une couche dans l'environnement d'exécution Python 3.12

L'exemple de AWS SAM modèle suivant crée une couche sur l'environnement d'exécution Python 3.12.

```
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
```

Exemple de modèle 2 : création d'une couche à l'aide d'un makefile personnalisé

L'exemple de AWS SAM modèle suivant utilise un modèle personnalisé `makefile` pour créer la couche.

```
Resources:
  MyLayer:
    Type: AWS::Serverless::LayerVersion
    Properties:
      ContentUri: my_layer
      CompatibleRuntimes:
        - python3.12
```

```
    Metadata:
      BuildMethod: makefile
```

Le `makefile` suivant contient la cible de génération et les commandes qui seront exécutées. Notez que la propriété `ContentUri` est définie sur `my_layer`. Le makefile doit donc se trouver dans la racine du répertoire `my_layer`, et le nom de fichier doit être `Makefile`. Notez également que les artefacts de construction sont copiés dans le `python/` sous-répertoire afin que AWS Lambda celui-ci puisse trouver le code de couche.

```
build-MyLayer:
  mkdir -p "$(ARTIFACTS_DIR)/python"
  cp *.py "$(ARTIFACTS_DIR)/python"
  python -m pip install -r requirements.txt -t "$(ARTIFACTS_DIR)/python"
```

> **Note**
>
> Lorsque le `makefile` est appelé, la cible appropriée est déclenchée et les artefacts doivent être copiés dans la variable environnementale exposée`$ARTIFACTS_DIR`. Pour plus d'informations, reportez-vous aws-lambda-builders à GitHub.

Exemple de commandes de création sam

Procédez comme suit : les commandes `sam build` créent des couches qui incluent la section d'attribut de ressource `Metadata`.

```
# 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
```

Testez votre application sans serveur avec AWS SAM

Après avoir écrit et créé votre application, vous serez prêt à tester votre application pour vérifier qu'elle fonctionne correctement. Grâce à l'interface de ligne de AWS SAM commande (CLI), vous pouvez tester localement votre application sans serveur avant de la télécharger dans le AWS Cloud. Le test de votre application vous permet de confirmer la fonctionnalité, la fiabilité et les performances de l'application, tout en identifiant les problèmes (bogues) qui devront être résolus.

Cette section fournit des conseils sur les pratiques courantes que vous pouvez suivre pour tester votre application. Les rubriques de cette section se concentrent principalement sur les tests locaux que vous pouvez effectuer avant le déploiement dans le AWS cloud. Les tests avant le déploiement vous aident à identifier les problèmes de manière proactive, réduisant ainsi les coûts inutiles associés aux problèmes de déploiement. Chaque rubrique de cette section décrit un test que vous pouvez effectuer et indique à yoxi include href ? u les avantages de son utilisation, et inclut des exemples vous montrant comment effectuer le test. Après avoir testé votre application, vous serez prêt à corriger tous les problèmes que vous aurez découverts.

Rubriques

- Présentation des tests à l'aide de la sam local commande
- Invoquez localement des fonctions Lambda avec AWS SAM
- Exécutez API Gateway localement avec AWS SAM
- Présentation des tests dans le cloud avec sam remote test-event
- Présentation des tests dans le cloud avec sam remote invoke
- Automatisez les tests d'intégration locaux avec AWS SAM
- Générez des exemples de charges utiles d'événements avec AWS SAM
- Tester et déboguer des fonctions durables

Présentation des tests à l'aide de la sam local commande

Utilisez la AWS Serverless Application Model commande Command Line Interface (AWS SAMCLI) `sam local` pour tester vos applications sans serveur localement.

Pour une introduction à la AWS SAMCLI, voirQu'est-ce que c'est AWS SAMCLI ?.

Conditions préalables

Pour utiliser `sam local`, installez la CLI AWS SAM en procédant comme suit :

- AWS SAM prérequis.
- Installer la CLI AWS SAM.

Avant d'utiliser `sam local`, nous vous recommandons d'avoir des connaissances de base sur les points suivants :

- Configuration de la CLI AWS SAM.
- Créez votre application dans AWS SAM.
- Initiation à la construction avec AWS SAM.
- Présentation du déploiement avec AWS SAM.

Utilisation de la commande sam local

Utilisez la commande `sam local` avec l'une de ses sous-commandes pour effectuer différents types de tests locaux pour votre application.

```
$ sam local <subcommand>
```

Pour en savoir plus sur chaque sous-commande, consultez les ressources suivantes :

- Présentation de sam local generate-event— Génère Service AWS des événements pour les tests locaux.
- Présentation de sam local invoke : lancez un appel unique d'une fonction AWS Lambda au niveau local.
- Présentation de sam local start-api : exécutez vos fonctions Lambda à l'aide d'un serveur HTTP local.
- Présentation de sam local start-lambda— Exécutez vos fonctions Lambda à l'aide d'un serveur HTTP local à utiliser avec le AWS CLI ou. SDKs

Présentation des tests avec sam local generate-event

Utilisez la AWS Serverless Application Model `sam local generate-event` sous-commande Command Line Interface (AWS SAMCLI) pour générer des échantillons de charge utile d'événements pour les applications prises en charge. Services AWS Vous pouvez ensuite modifier et transmettre ces événements à des ressources locales pour les tester.

- Pour une introduction à la AWS SAMCLI, voir Qu'est-ce que c'est AWS SAMCLI ?
- Pour obtenir la liste des options de commande `sam local generate-event`, consultez sam local generate-event.

Un événement est un objet JSON qui est généré lorsqu'un utilisateur Service AWS exécute une action ou une tâche. Ces événements contiennent des informations spécifiques, telles que les données traitées ou le timestamp de l'événement. La plupart des Services AWS génèrent des événements et les événements de chaque service sont formatés de manière unique pour le service en question.

Les événements générés par un service sont transmis à d'autres services en tant que source d'événements. Par exemple, un élément placé dans un compartiment Amazon Simple Storage Service (Amazon S3) peut générer un événement. Cet événement peut ensuite être utilisé comme source d'événement pour une fonction AWS Lambda afin de poursuivre le traitement des données.

Les événements que vous générez avec `sam local generate-event` sont formatés selon la même structure que les événements réels créés par le AWS service. Vous pouvez modifier le contenu de ces événements et les utiliser pour tester les ressources de votre application.

Conditions préalables

Pour utiliser `sam local generate-event`, installez la CLI AWS SAM en procédant comme suit :

- AWS SAM prérequis.
- Installer la CLI AWS SAM.

Avant d'utiliser `sam local generate-event`, nous vous recommandons d'avoir des connaissances de base sur les points suivants :

- Configuration de la CLI AWS SAM.
- Créez votre application dans AWS SAM.

- Initiation à la construction avec AWS SAM.
- Présentation du déploiement avec AWS SAM.

Générer des exemples d'événements

Utilisez la AWS SAMCLI `sam local generate-event` sous-commande pour générer des événements pour les personnes prises en charge Services AWS.

Pour voir la liste des produits pris en charge Services AWS

1. Exécutez les commandes suivantes :

```
$ sam local generate-event
```

2. La liste des produits pris en charge Services AWS s'affichera. Voici un exemple :

```
$ sam local generate-event
...
Commands:
  alb
  alexa-skills-kit
  alexa-smart-home
  apigateway
  appsync
  batch
  cloudformation
  ...
```

Pour générer un événement local

1. Exécutez `sam local generate-event` et fournissez le nom du service pris en charge. La liste des types d'événements que vous pouvez générer s'affichera. Voici un exemple :

```
$ 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. Pour générer l'exemple d'événement, exécutez `sam local generate-event` en fournissant le service et le type d'événement.

```
$ sam local generate-event <service> <event>
```

Voici un exemple :

```
$ 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"
        }
      }
    }
  ]
}
```

Ces exemples d'événements contiennent des valeurs d'espace réservé. Vous pouvez modifier ces valeurs pour faire référence aux ressources réelles dans votre application ou pour faciliter les tests locaux.

Pour modifier un exemple d'événement

1. Vous pouvez modifier les exemples d'événements à l'invite de commande. Pour afficher vos options, exécutez ce qui suit :

```
$ sam local generate-event <service> <event> --help
```

Voici un exemple :

```
$ 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. Utilisez l'une de ces options à l'invite de commande pour modifier votre exemple de charge utile d'événement. Voici un exemple :

```
$ 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"
        }
```

```
      }
    }
  ]
}
```

Utiliser les événements générés pour les tests locaux

Enregistrez les événements générés localement et utilisez d'autres sous-commandes `sam local` pour les tester.

Pour enregistrer vos événements générés localement

- Exécutez les commandes suivantes :

  ```
  $ sam local generate-event <service> <event> <event-option> > <filename.json>
  ```

  Voici un exemple d'événement enregistré sous forme de fichier `s3.json` dans le dossier `events` de notre projet.

  ```
  sam-app$ sam local generate-event s3 put --bucket amzn-s3-demo-bucket > events/
  s3.json
  ```

Pour utiliser un événement généré pour des tests locaux

- Transmettez l'événement avec d'autres sous-commandes `sam local` à l'aide de l'option `--event`.

  Voici un exemple d'utilisation de l'événement `s3.json` pour appeler notre fonction Lambda localement :

  ```
  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
```

En savoir plus

Pour obtenir la liste de toutes les options `sam local generate-event`, consultez sam local generate-event.

Pour une démonstration de l'utilisation de `sam local`, consultez AWS SAM pour le développement local. Tester AWS Cloud des ressources issues d'environnements de développement locaux dans le cadre de la série Serverless Land Sessions with SAM on YouTube.

Présentation des tests avec sam local invoke

Utilisez la AWS Serverless Application Model `sam local invoke` sous-commande Command Line Interface (AWS SAMCLI) pour lancer un appel unique d'une AWS Lambda fonction localement.

- Pour une introduction à la AWS SAMCLI, voir Qu'est-ce que c'est AWS SAMCLI ?
- Pour obtenir la liste des options de commande `sam local invoke`, consultez sam local invoke.
- Pour un exemple d'utilisation de `sam local invoke` dans le cadre d'un flux de travail de développement classique, consultez Étape 7 : (Facultatif) testez votre application localement.

> **Note**
>
> `sam local invoke`prend en charge des fonctions durables avec des fonctionnalités de point de contrôle et de rediffusion automatiques. Lorsque des fonctions durables sont invoquées localement, l'état d'exécution est automatiquement géré.

Conditions préalables

Pour utiliser `sam local invoke`, installez la CLI AWS SAM en procédant comme suit :

- AWS SAM prérequis.
- Installer la CLI AWS SAM.

Avant d'utiliser `sam local invoke`, nous vous recommandons d'avoir des connaissances de base sur les points suivants :

- Configuration de la CLI AWS SAM.
- Créez votre application dans AWS SAM.
- Initiation à la construction avec AWS SAM.
- Présentation du déploiement avec AWS SAM.

Appel d'une fonction Lambda localement

Lorsque vous utilisez `sam local invoke`, la CLI AWS SAM suppose que le répertoire de travail actuel est le répertoire racine du projet. La CLI AWS SAM recherche d'abord un fichier `template.[yaml|yml]` dans un sous-dossier `.aws-sam`. Si elle ne le trouve pas, la CLI AWS SAM recherche un fichier `template.[yaml|yml]` dans votre répertoire de travail actuel.

Pour appeler une fonction Lambda localement

1. À partir du répertoire racine de votre projet, effectuez les actions suivantes :

```
$ sam local invoke <options>
```

2. Si votre application contient plusieurs fonctions, indiquez l'identifiant logique de la fonction. Voici un exemple :

```
$ sam local invoke HelloWorldFunction
```

3. La CLI AWS SAM crée votre fonction dans un conteneur local à l'aide de Docker. Elle invoque ensuite votre fonction et génère la réponse de votre fonction.

   Voici un exemple :

```
$ 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\"}"}%
```

Gestion des journaux

Lorsque vous utilisez `sam local invoke`, le résultat d'exécution de la fonction Lambda (par exemple, les journaux) sort vers `stderr` et le résultat de la fonction Lambda sors vers `stdout`.

Voici un exemple de fonction Lambda de base :

```
def handler(event, context):
    print("some log") # this goes to stderr
    return "hello world" # this goes to stdout
```

Vous pouvez enregistrer ces résultats standard. Voici un exemple :

```
$ 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
```

Vous pouvez utiliser ces résultats standard pour automatiser davantage vos processus de développement locaux.

Options

Transmettez des événements personnalisés pour appeler la fonction Lambda

Pour transmettre un événement à la fonction Lambda, utilisez l'option `--event`. Voici un exemple :

```
$ sam local invoke --event events/s3.json S3JsonLoggerFunction
```

Vous pouvez créer des événements à l'aide de la sous-commande `sam local generate-event`. Pour en savoir plus, veuillez consulter la section Présentation des tests avec sam local generate-event.

Transmettre des variables d'environnement lors de l'appel de votre fonction Lambda

Si votre fonction Lambda utilise des variables d'environnement, vous pouvez les transmettre lors des tests locaux à l'aide de l'option `--env-vars`. C'est un excellent moyen de tester une fonction Lambda localement avec des services de votre application déjà déployés dans le cloud. Voici un exemple :

```
$ sam local invoke --env-vars locals.json
```

Spécifier un modèle ou une fonction

Pour spécifier un modèle à référencer par la CLI AWS SAM, utilisez l'option `--template`. Ils AWS SAMCLI chargeront uniquement ce AWS SAM modèle et les ressources vers lesquelles il pointe.

Pour appeler une fonction d'une application ou d'une pile imbriquée, fournissez l'identifiant logique de l'application ou de la pile avec l'identifiant logique. Voici un exemple :

```
$ sam local invoke StackLogicalId/FunctionLogicalId
```

Tester une fonction Lambda à partir de votre projet Terraform

Utilisez l'option `--hook-name` pour tester localement les fonctions Lambda de vos projets Terraform. Pour en savoir plus, veuillez consulter la section Utilisation de la CLI AWS SAM avec Terraform pour le débogage et les tests locaux.

Voici un exemple :

```
$ sam local invoke --hook-name terraform --beta-features
```

Bonnes pratiques

Si votre application possède un répertoire `.aws-sam` qui exécute `sam build`, assurez-vous d'exécuter `sam build` chaque fois que vous mettez à jour le code de votre fonction. Exécutez ensuite `sam local invoke` pour tester localement votre code de fonction mis à jour.

Les tests locaux constituent une excellente solution pour un développement et des tests rapides avant le déploiement dans le cloud. Toutefois, les tests locaux ne valident pas tout, notamment les autorisations entre vos ressources dans le cloud. Dans la mesure du possible, testez vos applications dans le cloud. Nous vous recommandons d'utiliser `sam sync` pour accélérer vos flux de travail de test dans le cloud.

Exemples

Générer un exemple d'événement Amazon API Gateway et l'utiliser pour appeler une fonction Lambda localement

Tout d'abord, nous générons une charge utile d'événement API HTTP API Gateway et l'enregistrons dans notre dossier `events`.

```
$ sam local generate-event apigateway http-api-proxy > events/apigateway_event.json
```

Ensuite, nous modifions notre fonction Lambda pour qu'elle renvoie une valeur de paramètre à partir de l'événement.

```
def lambda_handler(event, context):
    print("HelloWorldFunction invoked")
    return {
        "statusCode": 200,
        "body": json.dumps({
```

```
            "message": event['queryStringParameters']['parameter2'],
        }),
    }
```

Ensuite, nous invoquons localement notre fonction Lambda et fournissons notre événement personnalisé.

```
$ 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\"}"}%
```

Transmettre des variables d'environnement lors de l'appel d'une fonction Lambda localement

Cette application possède une fonction Lambda qui utilise une variable d'environnement pour un nom de table Amazon DynamoDB. Voici un exemple de la fonction définie dans le AWS SAM modèle :

```
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
...
```

Nous souhaitons tester localement notre fonction Lambda tout en la faisant interagir avec notre table DynamoDB dans le cloud. Pour ce faire, nous créons notre fichier de variables d'environnement et l'enregistrons dans le répertoire racine de notre projet sous le nom `locals.json`. La valeur fournie ici pour `SAMPLE_TABLE` fait référence à notre table DynamoDB dans le cloud.

```
{
    "getAllItemsFunction": {
        "SAMPLE_TABLE": "dev-demo-SampleTable-1U991234LD5UM98"
    }
}
```

Ensuite, nous exécutons `sam local invoke` et transmettons nos variables d'environnement avec l'option `--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":"{}"}
```

En savoir plus

Pour obtenir la liste de toutes les options `sam local invoke`, consultez sam local invoke.

Pour une démonstration de l'utilisation de `sam local`, consultez AWS SAM pour le développement local. Tester AWS Cloud des ressources issues d'environnements de développement locaux dans le cadre de la série Serverless Land Sessions with SAM on YouTube.

Présentation des tests avec sam local start-api

Utilisez la AWS Serverless Application Model `sam local start-api` sous-commande Command Line Interface (AWS SAMCLI) pour exécuter vos AWS Lambda fonctions localement et les tester via

un hôte de serveur HTTP local. Ce type de test est utile pour les fonctions Lambda qui sont appelées par un point de terminaison Amazon API Gateway.

- Pour une introduction à la AWS SAMCLI, voir Qu'est-ce que c'est AWS SAMCLI ?
- Pour obtenir la liste des options de commande `sam local start-api`, consultez sam local start-api.
- Pour un exemple d'utilisation de `sam local start-api` dans le cadre d'un flux de travail de développement classique, consultez Étape 7 : (Facultatif) testez votre application localement.

> **Note**
>
> `sam local start-api`prend en charge des fonctions durables avec des fonctionnalités de point de contrôle et de rediffusion automatiques. Les fonctions durables fonctionnent parfaitement avec le point de terminaison API Gateway local.

Conditions préalables

Pour utiliser `sam local start-api`, installez la CLI AWS SAM en procédant comme suit :

- AWS SAM prérequis.
- Installer la CLI AWS SAM.

Avant d'utiliser `sam local start-api`, nous vous recommandons d'avoir des connaissances de base sur les points suivants :

- Configuration de la CLI AWS SAM.
- Créez votre application dans AWS SAM.
- Initiation à la construction avec AWS SAM.
- Présentation du déploiement avec AWS SAM.

Utilisation de sam local start-api

Lorsque vous utilisez `sam local start-api`, la CLI AWS SAM suppose que le répertoire de travail actuel est le répertoire racine du projet. La CLI AWS SAM recherche d'abord un fichier

`template.[yaml|yml]` dans un sous-dossier `.aws-sam`. Si elle ne le trouve pas, la CLI AWS SAM recherche un fichier `template.[yaml|yml]` dans votre répertoire de travail actuel.

Pour démarrer un serveur HTTP local

1. À partir du répertoire racine de votre projet, effectuez les actions suivantes :

```
$ sam local start-api <options>
```

2. La CLI AWS SAM crée vos fonctions Lambda dans un conteneur Docker local. Elle affiche ensuite l'adresse locale du point de terminaison de votre serveur HTTP. Voici un exemple :

```
$ 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. Vous pouvez appeler votre fonction Lambda via le navigateur ou l'invite de commande. Voici un exemple :

```
sam-app$ curl http://127.0.0.1:3000/hello
{"message": "Hello world!"}%
```

4. Lorsque vous modifiez le code de votre fonction Lambda, tenez compte des points suivants pour actualiser votre serveur HTTP local :

- Si votre application ne possède pas de répertoire `.aws-sam` et que votre fonction utilise un langage interprété, la CLI AWS SAM met automatiquement à jour votre fonction en créant un nouveau conteneur et en l'hébergeant.
- Si votre application possède un répertoire `.aws-sam`, vous devez exécuter `sam build` pour mettre à jour votre fonction. Exécutez ensuite à nouveau `sam local start-api` pour héberger la fonction.
- Si votre fonction utilise un langage compilé ou si votre projet nécessite une prise en charge complexe d'empaquetage, exécutez votre propre solution de création pour mettre à jour votre fonction. Exécutez ensuite à nouveau `sam local start-api` pour héberger la fonction.

Fonctions Lambda qui utilisent des mécanismes d'autorisation Lambda

> **Note**
>
> Cette fonctionnalité est nouvelle dans la version 1.80.0 de la CLI AWS SAM. Pour effectuer une mise à niveau, consultez Mise à niveau de la CLI AWS SAM en cours.

Pour les fonctions Lambda qui utilisent des mécanismes d'autorisation Lambda, la CLI AWS SAM invoquera automatiquement votre mécanisme d'autorisation Lambda avant d'appeler le point de terminaison de votre fonction Lambda.

Voici un exemple de démarrage d'un serveur HTTP local pour une fonction qui utilise un mécanisme d'autorisation Lambda :

```
$ 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
```

Lorsque vous invoquez le point de terminaison de votre fonction Lambda via le serveur HTTP local, la CLI AWS SAM invoque d'abord votre mécanisme d'autorisation Lambda. Si l'autorisation est réussie, la CLI AWS SAM invoquera le point de terminaison de votre fonction Lambda. Voici un exemple :

```
$ 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 -
```

Options

Réutiliser en permanence les conteneurs pour accélérer les appels de fonctions locales

Par défaut, la CLI AWS SAM crée un nouveau conteneur chaque fois que votre fonction est invoquée via le serveur HTTP local. Utilisez l'option `--warm-containers` pour réutiliser

automatiquement votre conteneur pour les appels de fonctions. Cela accélère le temps nécessaire à la CLI AWS SAM pour préparer votre fonction Lambda pour l'invocation locale. Vous pouvez personnaliser davantage cette option en fournissant l'argument `eager` ou `lazy`.

- `eager` : les conteneurs pour toutes les fonctions sont chargés au démarrage et persistent entre les appels.
- `lazy` : les conteneurs ne sont chargés que lorsque chaque fonction est appelée pour la première fois. Ils persistent ensuite pour des appels supplémentaires.

Voici un exemple :

```
$ sam local start-api --warm-containers eager
```

Lorsque vous utilisez `--warm-containers` et modifiez le code de votre fonction Lambda :

- Si votre application dispose d'un répertoire `.aws-sam`, exécutez `sam build` pour mettre à jour le code de votre code de fonction dans les artefacts de création de votre application.
- Lorsqu'une modification de code est détectée, la CLI AWS SAM arrête automatiquement le conteneur de fonctions Lambda.
- Lorsque vous invoquez à nouveau la fonction, la CLI AWS SAM crée automatiquement un nouveau conteneur.

Spécifier une image de conteneur à utiliser pour vos fonctions Lambda

Par défaut, la CLI AWS SAM utilise les images de base Lambda provenant d'Amazon Elastic Container Registry (Amazon ECR) pour invoquer vos fonctions localement. Utilisez l'option `--invoke-image` pour référencer une image de conteneur personnalisée. Voici un exemple :

```
$ sam local start-api --invoke-image public.ecr.aws/sam/emu-python3.8
```

Vous pouvez spécifier la fonction à utiliser avec l'image de conteneur personnalisée. Voici un exemple :

```
$ sam local start-api --invoke-image Function1=amazon/aws/sam-cli-emulation-image-
python3.8
```

Spécifier un modèle à tester localement

Pour spécifier un modèle à référencer par la CLI AWS SAM, utilisez l'option `--template`. Ils AWS SAMCLI chargeront uniquement ce AWS SAM modèle et les ressources vers lesquelles il pointe. Voici un exemple :

```
$ sam local start-api --template myTemplate.yaml
```

Spécifier l'environnement de développement hôte de votre fonction Lambda

Par défaut, la sous-commande `sam local start-api` crée un serveur HTTP utilisant `localhost` avec l'adresse IP `127.0.0.1`. Vous pouvez personnaliser ces valeurs si votre environnement de développement local est isolé de votre ordinateur local.

Utilisez l'option `--container-host` pour spécifier un hôte. Voici un exemple :

```
$ sam local start-api --container-host host.docker.internal
```

Utilisez l'option `--container-host-interface` pour spécifier l'adresse IP du réseau hôte à laquelle les ports de conteneur doivent se relier. Voici un exemple :

```
$ sam local start-api --container-host-interface 0.0.0.0
```

Bonnes pratiques

Si votre application possède un répertoire `.aws-sam` qui exécute `sam build`, assurez-vous d'exécuter `sam build` chaque fois que vous mettez à jour le code de votre fonction. Exécutez ensuite `sam local start-api` pour tester localement votre code de fonction mis à jour.

Les tests locaux constituent une excellente solution pour un développement et des tests rapides avant le déploiement dans le cloud. Toutefois, les tests locaux ne valident pas tout, notamment les autorisations entre vos ressources dans le cloud. Dans la mesure du possible, testez vos applications dans le cloud. Nous vous recommandons d'utiliser `sam sync` pour accélérer vos flux de travail de test dans le cloud.

En savoir plus

Pour obtenir la liste de toutes les options `sam local start-api`, consultez sam local start-api.

Présentation des tests avec sam local start-lambda

Utilisez la AWS SAMCLI sous-commande `sam local start-lambda` pour appeler votre fonction Lambda via AWS CLI le et. SDKs Cette commande démarre un point de terminaison local qui émule Lambda.

- Pour une introduction à la AWS SAMCLI, voir Qu'est-ce que c'est AWS SAMCLI ?
- Pour obtenir la liste des options de commande `sam local start-lambda`, consultez sam local start-lambda.

> **Note**
>
> `sam local start-lambda`prend en charge des fonctions durables avec des fonctionnalités de point de contrôle et de rediffusion automatiques. Les fonctions durables fonctionnent parfaitement avec le point de terminaison Lambda local.

Conditions préalables

Pour utiliser `sam local start-lambda`, installez la CLI AWS SAM en procédant comme suit :

- AWS SAM prérequis.
- Installer la CLI AWS SAM.

Avant d'utiliser `sam local start-lambda`, nous vous recommandons d'avoir des connaissances de base sur les points suivants :

- Configuration de la CLI AWS SAM.
- Créez votre application dans AWS SAM.
- Initiation à la construction avec AWS SAM.
- Présentation du déploiement avec AWS SAM.

Utilisation de sam local start-lambda

Lorsque vous utilisez `sam local start-lambda`, la CLI AWS SAM suppose que le répertoire de travail actuel est le répertoire racine du projet. La CLI AWS SAM recherche d'abord un fichier

`template.[yaml|yml]` dans un sous-dossier `.aws-sam`. Si elle ne le trouve pas, la CLI AWS SAM recherche un fichier `template.[yaml|yml]` dans votre répertoire de travail actuel.

Pour utiliser sam local start-lambda

1. À partir du répertoire racine de votre projet, effectuez les actions suivantes :

```
$ sam local start-lambda <options>
```

2. La CLI AWS SAM crée vos fonctions Lambda dans un conteneur Docker local. Elle transmet ensuite l'adresse locale au point de terminaison de votre serveur HTTP. Voici un exemple :

```
$ 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. Utilisez le AWS CLI ou SDKs pour appeler votre fonction Lambda localement.

   Voici un exemple avec utilisation de la AWS CLI :

```
$ aws lambda invoke --function-name "HelloWorldFunction" --endpoint-
url "http://127.0.0.1:3001" --no-verify-ssl out.txt

StatusCode: 200
(END)
```

   Voici un exemple d'utilisation du AWS SDK for Python :

```
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")
```

Options

Spécifier un modèle

Pour spécifier un modèle à référencer par la CLI AWS SAM, utilisez l'option `--template`. Ils AWS SAMCLI chargeront uniquement ce AWS SAM modèle et les ressources vers lesquelles il pointe. Voici un exemple :

```
$ sam local start-lambda --template myTemplate.yaml
```

Pour plus d'informations sur les AWS SAM modèles, consultezAWS SAM anatomie du modèle.

Bonnes pratiques

Si votre application possède un répertoire `.aws-sam` qui exécute `sam build`, assurez-vous d'exécuter `sam build` chaque fois que vous mettez à jour le code de votre fonction. Exécutez ensuite `sam local start-lambda` pour tester localement votre code de fonction mis à jour.

Les tests locaux constituent une excellente solution pour un développement et des tests rapides avant le déploiement dans le cloud. Toutefois, les tests locaux ne valident pas tout, notamment les autorisations entre vos ressources dans le cloud. Dans la mesure du possible, testez vos applications dans le cloud. Nous vous recommandons d'utiliser `sam sync` pour accélérer vos flux de travail de test dans le cloud.

En savoir plus

Pour obtenir la liste de toutes les options `sam local start-lambda`, consultez sam local start-lambda.

Invoquez localement des fonctions Lambda avec AWS SAM

L'invocation locale d'une fonction Lambda avant de la tester ou de la déployer dans le cloud peut présenter de nombreux avantages. Cela vous permet de tester plus rapidement la logique de votre fonction. Les tests locaux réduisent d'abord la probabilité d'identifier les problèmes lors des tests dans le cloud ou pendant le déploiement, ce qui peut vous aider à éviter des coûts inutiles. De plus, les tests locaux facilitent le débogage.

Vous pouvez appeler votre fonction Lambda localement en utilisant la sam local invoke commande et en fournissant l'identifiant logique de la fonction et un fichier d'événements. sam local invokeaccepte également `stdin` en tant qu'événement. Pour plus d'informations sur les événements, consultez Evénement dans le Guide du développeur AWS Lambda . Pour plus d'informations sur les formats de messages d'événements provenant de différents AWS services, consultez la section Utilisation AWS Lambda avec d'autres services dans le Guide du AWS Lambda développeur.

> **Note**
>
> La sam local invoke commande correspond à la commande AWS Command Line Interface (AWS CLI) `aws lambda invoke`. Vous pouvez utiliser l'une ou l'autre commande pour appeler une fonction Lambda.

Vous devez exécuter la commande sam local invoke dans le répertoire du projet qui contient la fonction que vous voulez appeler.

Exemples :

```
# 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
```

Fichier de variable d'environnement

Pour déclarer localement des variables d'environnement qui remplacent les valeurs définies dans vos modèles, procédez comme suit :

1. Créez un fichier JSON qui contient les variables d'environnement à remplacer.
2. Utilisez l'argument `--env-vars` pour remplacer les valeurs définies dans vos modèles.

Déclaration des variables d'environnement

Pour déclarer des variables d'environnement qui s'appliquent globalement à toutes les ressources, spécifiez un objet `Parameters` comme suit :

```
{
    "Parameters": {
        "TABLE_NAME": "localtable",
        "BUCKET_NAME": "amzn-s3-demo-bucket",
        "STAGE": "dev"
    }
}
```

Pour déclarer des variables d'environnement différentes pour chaque ressource, spécifiez des objets pour chaque ressource comme suit :

```
{
    "MyFunction1": {
        "TABLE_NAME": "localtable",
        "BUCKET_NAME": "amzn-s3-demo-bucket",
    },
    "MyFunction2": {
        "TABLE_NAME": "localtable",
        "STAGE": "dev"
    }
}
```

Lorsque vous spécifiez des objets pour chaque ressource, vous pouvez utiliser les identifiants suivants, énumérés dans l'ordre de la plus haute à la plus basse priorité :

1. `logical_id`
2. `function_id`
3. `function_name`
4. Identifiant de chemin complet

Vous pouvez utiliser les deux méthodes précédentes de déclaration des variables d'environnement dans un seul fichier. Ce faisant, les variables d'environnement que vous avez fournies pour des ressources spécifiques ont la priorité sur les variables d'environnement globales.

Enregistrez vos variables d'environnement dans un fichier JSON, tel que `env.json`.

Remplacement des valeurs des variables d'environnement

Pour remplacer les variables d'environnement par celles définies dans votre fichier JSON, utilisez l'argument `--env-vars` avec les commandes invoke ou start-api. Par exemple :

```
sam local invoke --env-vars env.json
```

Layers

Si votre application comporte des couches, pour plus d'informations sur la façon de déboguer les problèmes liés aux couches sur votre hôte local, consultez Améliorez l'efficacité en utilisant les couches Lambda avec AWS SAM.

En savoir plus

Pour un exemple pratique d'invocation de fonctions localement, voir le module 2 - Exécuter localement dans The Complete AWS SAM Workshop.

Exécutez API Gateway localement avec AWS SAM

L'exécution locale d'Amazon API Gateway peut présenter de nombreux avantages. Par exemple, l'exécution locale d'API Gateway vous permet de tester les points de terminaison d'API localement avant le déploiement AWS dans le cloud. Si vous effectuez d'abord des tests locaux, vous pouvez souvent réduire les tests et le développement dans le cloud, ce qui peut contribuer à réduire les coûts. De plus, l'exécution locale facilite le débogage.

Pour démarrer une instance locale d'API Gateway que vous pouvez utiliser pour tester la fonctionnalité de requête/réponse HTTP, utilisez la commande. `sam local start-api` AWS SAMCLI Cette fonctionnalité dispose d'un rechargement à chaud afin que vous puissiez rapidement développer et itérer vos fonctions.

> **Note**
>
> Le rechargement à chaud consiste à actualiser uniquement les fichiers qui ont été modifiés, l'état de l'application restant inchangé. En revanche, le rechargement en direct consiste à actualiser l'application entière et à perdre l'état de l'application.

Pour des instructions relatives à l'utilisation de la commande `sam local start-api`, veuillez consulter la section Présentation des tests avec sam local start-api.

Par défaut, AWS SAM utilise des intégrations de AWS Lambda proxy et prend en charge les deux `HttpApi` types de `Api` ressources. Pour plus d'informations sur les intégrations de proxy pour les types de `HttpApi` ressources, consultez la section Travailler avec les intégrations de AWS Lambda proxy pour HTTP APIs dans le guide du développeur d'API Gateway. Pour plus d'informations sur les intégrations de proxy avec les types de ressources `Api`, consultez Présentation de l'intégration de proxy Lambda API Gateway dans le Guide du développeur API Gateway.

Exemple :

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

AWS SAM trouve automatiquement dans votre AWS SAM modèle toutes les fonctions dont les sources d'`Api`événements `HttpApi` ou les sources d'événements sont définies. Ensuite, il monte la fonction sur les chemins HTTP définis.

Dans l'exemple `Api` ci-dessous, la fonction `Ratings` monte `ratings.py:handler()` dans `/ratings` pour les demandes `GET`.

```
Ratings:
  Type: AWS::Serverless::Function
  Properties:
    Handler: ratings.handler
    Runtime: python3.9
    Events:
      Api:
        Type: Api
        Properties:
          Path: /ratings
          Method: get
```

Voici un exemple de réponse `Api` :

```
// 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"
    });
}
```

Si vous modifiez le code de votre fonction, exécutez la commande `sam build` pour que `sam local start-api` détecte vos modifications.

Fichier de variable d'environnement

Pour déclarer localement des variables d'environnement qui remplacent les valeurs définies dans vos modèles, procédez comme suit :

1. Créez un fichier JSON qui contient les variables d'environnement à remplacer.
2. Utilisez l'argument `--env-vars` pour remplacer les valeurs définies dans vos modèles.

Déclaration des variables d'environnement

Pour déclarer des variables d'environnement qui s'appliquent globalement à toutes les ressources, spécifiez un objet `Parameters` comme suit :

```
{
    "Parameters": {
        "TABLE_NAME": "localtable",
        "BUCKET_NAME": "amzn-s3-demo-bucket",
        "STAGE": "dev"
    }
}
```

Pour déclarer des variables d'environnement différentes pour chaque ressource, spécifiez des objets pour chaque ressource comme suit :

```
{
```

```
    "MyFunction1": {
        "TABLE_NAME": "localtable",
        "BUCKET_NAME": "amzn-s3-demo-bucket",
    },
    "MyFunction2": {
        "TABLE_NAME": "localtable",
        "STAGE": "dev"
    }
}
```

Lorsque vous spécifiez des objets pour chaque ressource, vous pouvez utiliser les identifiants suivants, énumérés dans l'ordre de la plus haute à la plus basse priorité :

1. `logical_id`
2. `function_id`
3. `function_name`
4. Identifiant de chemin complet

Vous pouvez utiliser les deux méthodes précédentes de déclaration des variables d'environnement dans un seul fichier. Ce faisant, les variables d'environnement que vous avez fournies pour des ressources spécifiques ont la priorité sur les variables d'environnement globales.

Enregistrez vos variables d'environnement dans un fichier JSON, tel que `env.json`.

Remplacement des valeurs des variables d'environnement

Pour remplacer les variables d'environnement par celles définies dans votre fichier JSON, utilisez l'argument `--env-vars` avec les commandes invoke ou start-api. Par exemple :

```
$ sam local start-api --env-vars env.json
```

Layers

Si votre application comporte des couches, pour plus d'informations sur la façon de déboguer les problèmes liés aux couches sur votre hôte local, consultez Améliorez l'efficacité en utilisant les couches Lambda avec AWS SAM.

Présentation des tests dans le cloud avec sam remote test-event

Utilisez la AWS Serverless Application Model commande Command Line Interface (AWS SAM CLI) `sam remote test-event` pour accéder aux événements de test partageables pour vos AWS Lambda fonctions et les gérer.

Pour en savoir plus sur les événements de test partageables, consultez la rubrique relative aux événements de test partageables dans le Guide du développeur AWS Lambda .

Rubriques

- Configurer l'CLI AWS SAM pour utiliser sam remote test-event
- Utilisation de la commande sam remote test-event
- Utilisation des événements de test partageables
- Gestion des événements de test partageables

Conditions préalables

Pour utiliser `sam remote test-event`, installez la CLI AWS SAM en procédant comme suit :

- AWS SAM prérequis.
- Installer la CLI AWS SAM.

Si vous l'avez déjà AWS SAM CLI installé, nous vous recommandons de passer à la dernière version de la AWS SAMCLI version. Pour en savoir plus, veuillez consulter la section Mise à niveau de la CLI AWS SAM en cours.

Avant d'utiliser `sam remote test-event`, nous vous recommandons d'avoir des connaissances de base sur les points suivants :

- Configuration de la CLI AWS SAM.
- Créez votre application dans AWS SAM.
- Initiation à la construction avec AWS SAM.
- Présentation du déploiement avec AWS SAM.
- Présentation de l'utilisation sam sync de la synchronisation avec AWS Cloud.

Configurer l'CLI AWS SAM pour utiliser sam remote test-event

Effectuez les étapes de configuration suivantes pour utiliser la AWS SAM CLI `sam remote test-event` commande :

1. Configurez le AWS SAM CLI pour utiliser votre Compte AWS — Les événements de test partageables pour Lambda sont accessibles et gérés par les utilisateurs de Lambda. Compte AWS Pour configurer le AWS SAM CLI afin d'utiliser votre Compte AWS, voirConfiguration de la CLI AWS SAM.
2. Configurer les autorisations pour les événements de test partageables – Pour accéder aux événements de test partageables et les gérer, vous devez disposer des autorisations appropriées. Pour en savoir plus, consultez la rubrique relative aux événements de test partageables dans le Guide du développeur AWS Lambda .

Utilisation de la commande sam remote test-event

La AWS SAM CLI `sam remote test-event` commande fournit les sous-commandes suivantes que vous pouvez utiliser pour accéder à vos événements de test partageables et les gérer :

- `delete`— Supprime un événement de test partageable du registre des EventBridge schémas Amazon.
- `get`— Obtenez un événement de test partageable depuis le registre des EventBridge schémas.
- `list`— Répertorie les événements de test partageables existants pour une fonction dans le registre des EventBridge schémas.
- `put`— Enregistrez un événement depuis un fichier local dans le registre des EventBridge schémas.

Pour répertorier ces sous-commandes à l'aide du AWS SAM CLI, exécutez ce qui suit :

```
$ sam remote test-event --help
```

Suppression d'événements de test partageables

Vous pouvez supprimer un événement de test partageable à l'aide de la sous-commande `delete` et en indiquant :

- le nom de l'événement de test partageable à supprimer ;

- un ID acceptable pour la fonction Lambda associée à l'événement ;
- Si vous fournissez l'ID logique de la fonction Lambda, vous devez également fournir le nom de AWS CloudFormation pile associé à la fonction Lambda.

Voici un exemple :

```
$ sam remote test-event delete HelloWorldFunction --stack-name sam-app --name demo-
event
```

Pour obtenir la liste des options à utiliser avec la sous-commande `delete`, consultez sam remote test-event delete. Vous pouvez également exécuter les opérations suivantes depuis AWS SAM CLI :

```
$ sam remote test-event delete --help
```

Obtention d'événements de test partageables

Vous pouvez obtenir un événement de test partageable à partir du registre des EventBridge schémas en utilisant la `get` sous-commande associée à ce qui suit :

- le nom de l'événement de test partageable à obtenir ;
- un ID acceptable pour la fonction Lambda associée à l'événement ;
- Si vous fournissez l'ID logique de la fonction Lambda, vous devez également fournir le nom de AWS CloudFormation pile associé à la fonction Lambda.

Voici un exemple qui permet d'obtenir un événement de test partageable nommé `demo-event` qui est associé à la fonction Lambda `HelloWorldFunction` de la pile `sam-app`. Cette commande imprime l'événement sur votre console.

```
$ sam remote test-event get HelloWorldFunction --stack-name sam-app --name demo-event
```

Pour obtenir un événement de test partageable et l'enregistrer sur votre machine locale, utilisez l'option `--output-file` et fournissez un chemin et un nom de fichier. Voici un exemple d'enregistrement de `demo-event.json` comme `demo-event` dans le répertoire de travail actuel :

```
$ sam remote test-event get HelloWorldFunction --stack-name sam-app --name demo-event
 --output-file demo-event.json
```

Pour obtenir la liste des options à utiliser avec la sous-commande `get`, consultez sam remote test-event get. Vous pouvez également exécuter les opérations suivantes depuis AWS SAM CLI :

```
$ sam remote test-event get --help
```

Liste d'événements de test partageables

Vous pouvez répertorier tous les événements de test partageables pour une fonction Lambda spécifique à partir du registre de schémas. Utilisez la sous-commande `list` en indiquant :

- un ID acceptable pour la fonction Lambda associée aux événements ;
- Si vous fournissez l'ID logique de la fonction Lambda, vous devez également fournir le nom de CloudFormation pile associé à la fonction Lambda.

Voici un exemple qui permet d'obtenir une liste de tous les événements de test partageables associés à la fonction Lambda `HelloWorldFunction` de la pile `sam-app` :

```
$ sam remote test-event list HelloWorldFunction --stack-name sam-app
```

Pour obtenir la liste des options à utiliser avec la sous-commande `list`, consultez sam remote test-event list. Vous pouvez également exécuter les opérations suivantes depuis AWS SAM CLI :

```
$ sam remote test-event list --help
```

Enregistrement d'événements de test partageables

Vous pouvez enregistrer des événements de test partageables dans le registre des EventBridge schémas. Utilisez la sous-commande `put` en indiquant :

- un ID acceptable pour la fonction Lambda associée à l'événement de test partageable ;
- un nom d'événement de test partageable ;
- le chemin du fichier et le nom de l'événement local à charger.

Voici un exemple qui enregistre l'événement local `demo-event.json` comme `demo-event` et qui l'associe à la fonction Lambda `HelloWorldFunction` de la pile `sam-app` :

```
$ sam remote test-event put HelloWorldFunction --stack-name sam-app --name demo-event
 --file demo-event.json
```

Si un événement de test partageable portant le même nom existe dans le registre du EventBridge schéma, il ne le AWS SAM CLI remplacera pas. Pour le remplacer, ajoutez l'option `--force` à votre commande.

Pour obtenir la liste des options à utiliser avec la sous-commande `put`, consultez sam remote test-event put. Vous pouvez également exécuter les opérations suivantes depuis AWS SAM CLI :

```
$ sam remote test-event put --help
```

Utilisation des événements de test partageables

Utilisez des événements de test partageables pour tester vos fonctions Lambda dans AWS Cloud la commande. `sam remote invoke` Pour en savoir plus, veuillez consulter la section Transmettre des événements de test partageables à une fonction Lambda dans le cloud.

Gestion des événements de test partageables

Cette rubrique contient des exemples de gestion et d'utilisation d'événements de test partageables.

Obtenir un événement de test partageable, le modifier et l'utiliser

Vous pouvez obtenir un événement de test partageable depuis le registre des EventBridge schémas, le modifier localement et utiliser l'événement de test local avec votre fonction Lambda dans le. AWS Cloud Voici un exemple :

1. Récupérer l'événement de test partageable – Utilisez la sous-commande `sam remote test-event get` pour récupérer un événement de test partageable pour une fonction Lambda spécifique et l'enregistrer localement :

   ```
   $ sam remote test-event get HelloWorldFunction --stack-name sam-app --name demo-event
    --output-file demo-event.json
   ```

2. Modifier l'événement de test partageable – Utilisez l'éditeur de texte de votre choix pour modifier l'événement de test partageable.
3. Utiliser l'événement de test partageable – Utilisez la commande `sam remote invoke` et indiquez le chemin et le nom du fichier de l'événement avec `--event-file` :

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --event-file demo-
event.json
```

Obtenir un événement de test partageable, le modifier, le charger et l'utiliser

Vous pouvez obtenir un événement de test partageable à partir du registre des EventBridge schémas, le modifier localement et le télécharger. Vous pouvez ensuite transmettre l'événement de test partageable directement à votre fonction Lambda dans l' AWS Cloud. Voici un exemple :

1. Récupérer l'événement de test partageable – Utilisez la sous-commande `sam remote test-event get` pour récupérer un événement de test partageable pour une fonction Lambda spécifique et l'enregistrer localement :

```
$ sam remote test-event get HelloWorldFunction --stack-name sam-app --name demo-event
 --output-file demo-event.json
```

2. Modifier l'événement de test partageable – Utilisez l'éditeur de texte de votre choix pour modifier l'événement de test partageable.
3. Télécharger l'événement de test partageable : utilisez la `sam remote test-event put` sous-commande pour télécharger et enregistrer l'événement de test partageable dans le registre du EventBridge schéma. Dans cet exemple, nous utilisons l'option `--force` pour remplacer une ancienne version de notre test partageable :

```
$ sam remote test-event put HelloWorldFunction --stack-name sam-app --name demo-event
 --file demo-event.json --force
```

4. Transmettre l'événement de test partageable à votre fonction Lambda – Utilisez la commande `sam remote invoke` pour transmettre l'événement de test partageable directement à votre fonction Lambda dans l' AWS Cloud :

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --test-event-name demo-
event
```

Présentation des tests dans le cloud avec sam remote invoke

Utilisez la AWS Serverless Application Model commande Command Line Interface (AWS SAM CLI) `sam remote invoke` pour interagir avec les AWS ressources prises en charge dans le AWS Cloud. Vous pouvez utiliser `sam remote invoke` pour appeler les ressources suivantes :

- Amazon Kinesis Data Streams : envoyez des enregistrements de données aux applications Kinesis Data Streams.
- AWS Lambda— Invoquez et transmettez des événements à vos fonctions Lambda.
- Amazon Simple Queue Service (Amazon SQS) — Envoyez des messages aux files d'attente Amazon SQS.
- AWS Step Functions— Invoquez les machines d'état Step Functions pour démarrer l'exécution.

Pour une introduction à la AWS SAMCLI, voir Qu'est-ce que c'est AWS SAMCLI ?

Pour un exemple d'utilisation de `sam remote invoke` dans le cadre d'un flux de travail de développement classique, consultez Étape 5 : Interagissez avec votre fonction dans AWS Cloud.

Rubriques

- Utilisation de la même commande d'invocation à distance
- Utilisation des options de commande sam remote invoke
- Configurez le fichier de configuration de votre projet
- Exemples
- Liens connexes

Conditions préalables

Pour utiliser `sam remote invoke`, installez la CLI AWS SAM en procédant comme suit :

- AWS SAM prérequis.
- Installer la CLI AWS SAM.

Nous vous recommandons également de passer à la dernière version du AWS SAMCLI. Pour en savoir plus, veuillez consulter la section Mise à niveau de la CLI AWS SAM en cours.

Avant d'utiliser `sam remote invoke`, nous vous recommandons d'avoir des connaissances de base sur les points suivants :

- Configuration de la CLI AWS SAM.
- Créez votre application dans AWS SAM.
- Initiation à la construction avec AWS SAM.
- Présentation du déploiement avec AWS SAM.
- Présentation de l'utilisation sam sync de la synchronisation avec AWS Cloud.

Utilisation de la même commande d'invocation à distance

Avant d'utiliser cette commande, votre ressource doit être déployée sur AWS Cloud.

Utilisez la structure de commande suivante et exécutez-la à partir du répertoire racine de votre projet :

```
$ sam remote invoke <arguments> <options>
```

> **Note**
>
> Cette page affiche les options proposées à l'invite de commande. Vous pouvez également configurer les options dans le fichier de configuration de votre projet au lieu de les transmettre à l'invite de commandes. Pour en savoir plus, consultez Configuration des paramètres de projet.

Pour une description des `sam remote invoke` arguments et des options, consultezsam remote invoke.

Utilisation avec Kinesis Data Streams

Vous pouvez envoyer des enregistrements de données à une application Kinesis Data Streams. Il AWS SAM CLI enverra votre enregistrement de données et renverra un identifiant de partition et un numéro de séquence. Voici un exemple :

```
$ 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"
 }%
```

Pour envoyer un enregistrement de données

1. Entrez une valeur d'ID de ressource comme argument pour votre application Kinesis Data Streams. Pour plus d'informations sur une ressource valide IDs, consultez la section Identifiant de ressource.
2. Fournissez l'enregistrement de données sous forme d'événement à envoyer à votre application Kinesis Data Streams. Vous pouvez fournir l'événement sur la ligne de commande à l'aide de l'`--event`option, ou à partir d'un fichier à l'aide de`--event-file`. Si vous ne fournissez aucun événement, AWS SAM CLI envoie un événement vide.

Utilisation avec les fonctions Lambda

Vous pouvez appeler une fonction Lambda dans le cloud et transmettre un événement vide ou fournir un événement sur la ligne de commande ou à partir d'un fichier. Il AWS SAM CLI invoquera votre fonction Lambda et renverra sa réponse. Voici un exemple :

```
$ 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\"}"}%
```

Pour appeler une fonction Lambda

1. Fournissez une valeur d'ID de ressource comme argument pour votre fonction Lambda. Pour plus d'informations sur une ressource valide IDs, consultez la section Identifiant de ressource.
2. Fournissez un événement à envoyer à votre fonction Lambda. Vous pouvez fournir l'événement sur la ligne de commande à l'aide de l'`--event`option, ou à partir d'un fichier à l'aide de`--event-file`. Si vous ne fournissez aucun événement, AWS SAM CLI envoie un événement vide.

Fonctions Lambda configurées avec le streaming de réponses

La commande `sam remote invoke` prend en charge les fonctions Lambda configurées pour diffuser les réponses. Vous pouvez configurer une fonction Lambda pour diffuser les réponses à l'aide de la `FunctionUrlConfig` propriété de vos AWS SAM modèles. Lorsque vous utilisez `sam remote invoke`, la CLI AWS SAM détecte automatiquement votre configuration Lambda et l'invoque avec le streaming des réponses.

Pour obtenir un exemple, consultez Configuration d'une fonction Lambda pour le streaming des réponses.

Transmettre des événements de test partageables à une fonction Lambda dans le cloud

Les événements de test partageables sont des événements de test que vous pouvez partager avec d'autres personnes dans le même Compte AWS. Pour en savoir plus, consultez la rubrique relative aux événements de test partageables dans le Guide du développeur AWS Lambda .

Accès et gestion des événements de test partageables

Vous pouvez utiliser la AWS SAM CLI `sam remote test-event` commande pour accéder aux événements de test partageables et les gérer. Par exemple, vous pouvez utiliser `sam remote test-event` pour effectuer les opérations suivantes :

- Récupérez les événements de test partageables depuis le registre des EventBridge schémas Amazon.
- Modifiez les événements de test partageables localement et téléchargez-les dans le registre des EventBridge schémas.
- Supprimez les événements de test partageables du registre des EventBridge schémas.

Pour en savoir plus, veuillez consulter la section Présentation des tests dans le cloud avec sam remote test-event.

Transmettre un événement de test partageable à une fonction Lambda dans le cloud

Pour transmettre un événement de test partageable du registre de EventBridge schémas à votre fonction Lambda dans le cloud, utilisez l'option et `--test-event-name` indiquez le nom de l'événement de test partageable. Voici un exemple :

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --test-event-name demo-
event
```

Si vous enregistrez l'événement de test partageable localement, vous pouvez utiliser l'option `--event-file` et fournir le chemin et le nom du fichier de l'événement de test local. Voici un exemple :

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --event-file demo-
event.json
```

Utilisation d’ avec Amazon SQS

Vous pouvez envoyer des messages aux files d'attente Amazon SQS. Le résultat AWS SAM CLI renvoie ce qui suit :

- ID de message
- MD5 du corps du message
- Métadonnées de réponse

Voici un exemple :

```
$ sam remote invoke MySqsQueue --stack-name sqs-example -event hello

Sending message to SQS queue MySqsQueue

{
  "MD5OfMessageBody": "5d41402abc4b2a76b9719d911017c592",
  "MessageId": "05c7af65-9ae8-4014-ae28-809d6d8ec652"
}%
```

Pour envoyer un message

1. Fournissez une valeur d'ID de ressource comme argument pour la file d'attente Amazon SQS. Pour plus d'informations sur une ressource valide IDs, consultez la section Identifiant de ressource.
2. Indiquez un événement à envoyer à votre file d'attente Amazon SQS. Vous pouvez fournir l'événement sur la ligne de commande à l'aide de l'`--event`option, ou à partir d'un fichier à l'aide de`--event-file`. Si vous ne fournissez aucun événement, AWS SAM CLI envoie un événement vide.

Utilisation avec Step Functions

Vous pouvez appeler une machine d'état Step Functions pour démarrer l'exécution. AWS SAM CLIAttendra que le flux de travail de la machine à états soit terminé et renverra un résultat de la dernière étape de l'exécution. Voici un exemple :

```
$ sam remote invoke HelloWorldStateMachine --stack-name state-machine-example --
event '{"is_developer": true}'

Invoking Step Function HelloWorldStateMachine

"Hello Developer World"%
```

Pour invoquer une machine à états

1. Fournissez une valeur d'ID de ressource comme argument pour la machine d'état Step Functions. Pour plus d'informations sur une ressource valide IDs, consultez la section Identifiant de ressource.
2. Fournissez un événement à envoyer à votre machine d'état. Vous pouvez fournir l'événement sur la ligne de commande à l'aide de l'`--event`option, ou à partir d'un fichier à l'aide de`--event-file`. Si vous ne fournissez aucun événement, AWS SAM CLI envoie un événement vide.

Utilisation des options de commande sam remote invoke

Cette section décrit certaines des principales options que vous pouvez utiliser avec la `sam remote invoke` commande. Pour une liste complète des options, voirsam remote invoke.

Transférez un événement à votre ressource

Utilisez les options suivantes pour transmettre des événements à vos ressources dans le cloud :

- `--event`— Passez un événement sur la ligne de commande.
- `--event-file`— Transmet un événement depuis un fichier.

Exemples Lambda

`--event`À utiliser pour transmettre un événement sur la ligne de commande sous forme de valeur de chaîne :

```
$ 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!\"}"}%
```

`--event-file`À utiliser pour transmettre un événement depuis un fichier et fournir le chemin d'accès au fichier :

```
$ 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\"}"}%
```

Réussissez un événement en utilisant **`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\"}"}%
```

Configuration de la sortie de AWS SAMCLI réponse

Lorsque vous invoquez une ressource prise en charge avec`sam remote invoke`, le AWS SAMCLI renvoie une réponse contenant les éléments suivants :

- Métadonnées de la demande : métadonnées associées à la demande. Cela inclut un ID de demande et l'heure de début de la demande.
- Réponse de la ressource : réponse de votre ressource après avoir été invoquée dans le cloud.

Vous pouvez utiliser `--output` cette option pour configurer la réponse AWS SAM CLI de sortie. Les valeurs suivantes sont disponibles avec :

- `json`— Les métadonnées et la réponse aux ressources sont renvoyées dans une JSON structure. La réponse contient le résultat complet SDK.
- `text`— Les métadonnées sont renvoyées sous forme de structure de texte. La réponse de la ressource est renvoyée dans le format de sortie de la ressource.

Voici un exemple de résultat `json` :

```
$ 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\\\"}\"}"
}%
```

Lorsque vous spécifiez un résultat `json`, la réponse complète est renvoyée à `stdout`. Voici un exemple :

```
$ 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\\\"}\"}"
}%
```

Voici un exemple de résultat `text` :

```
$ 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\"}"}%
```

Lorsque vous spécifiez un résultat `text`, la sortie d'exécution de la fonction Lambda (par exemple, les journaux) est renvoyée vers `stderr`. La charge utile de la fonction Lambda est renvoyée à `stdout`. Voici un exemple :

```
$ 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\"}"}%
```

Personnaliser les paramètres Boto3

En `sam remote invoke` effet, il AWS SAM CLI utilise le AWS SDK pour Python (Boto3) afin d'interagir avec vos ressources dans le cloud. Vous pouvez utiliser l'option `--parameter` pour personnaliser les paramètres Boto3. Pour obtenir la liste des paramètres pris en charge que vous pouvez personnaliser, reportez-vous à la section `--parameter`.

Exemples

Appelez une fonction Lambda pour valider les valeurs des paramètres et vérifier les autorisations :

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --
parameter InvocationType="DryRun"
```

Utilisez l'**`--parameter`**option plusieurs fois dans une seule commande pour fournir plusieurs paramètres :

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --
parameter InvocationType="Event" --parameter LogType="None"
```

Autres options

Pour obtenir la liste complète des options `sam remote invoke`, consultez la section sam remote invoke.

Configurez le fichier de configuration de votre projet

Pour configurer `sam remote invoke` dans votre fichier de configuration, utilisez `remote_invoke` dans votre tableau. Voici un exemple de fichier `samconfig.toml` qui configure les valeurs par défaut de la commande `sam remote invoke`.

```
...
version =0.1

[default]
...
[default.remote_invoke.parameters]
stack_name = "cloud-app"
event = '{"message": "Hello!"}'
```

Exemples

Pour un exemple d'utilisation de basesam `remote invoke`, consultez la section Tester des AWS Lambda fonctions AWS SAM à distance dans le AWS Compute Blog.

Exemples de Kinesis Data Streams

Exemples de base

Envoyez un enregistrement de données à une application Kinesis Data Streams à partir d'un fichier. L'application Kinesis Data Streams est identifiée en fournissant un ARN pour l'ID de ressource :

```
$ sam remote invoke arn:aws:kinesis:us-west-2:01234567890:stream/kinesis-example-
KinesisStream-BgnLcAey4xUQ --event-file event.json
```

Envoyez un événement fourni sur la ligne de commande à une application Kinesis Data Streams :

```
$ 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"
}%
```

Obtenez l'identifiant physique de l'application Kinesis Data Streams. Indiquez ensuite un événement sur la ligne de commande :

```
$ 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"
}%
```

Fournissez une chaîne JSON sur la ligne de commande en tant qu'événement :

```
$ sam remote invoke KinesisStream --stack-name kinesis-example --event '{"method":
 "GET", "body": ""}'

Putting record to Kinesis data stream KinesisStream

{
  "ShardId": "shardId-000000000000",
  "SequenceNumber": "49646251411914806775980904492868617924990209230536441858"
}%
```

Envoyez un événement vide à l'application Kinesis Data Streams :

```
$ sam remote invoke KinesisStream --stack-name kinesis-example

Putting record to Kinesis data stream KinesisStream

{
  "ShardId": "shardId-000000000000",
  "SequenceNumber": "49646251411914806775980904866469008589597168190416224258"
}%
```

Renvoie la AWS SAM CLI réponse au format JSON :

```
$ 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
  }
}%
```

Renvoyez la sortie JSON à stdout :

```
$ 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
  }
}%
```

Exemples Lambda

Exemples de base

Invoquez une fonction Lambda en fournissant l'ARN comme ID de ressource :

```
$ sam remote invoke arn:aws:lambda:us-west-2:012345678910:function:sam-app-
HelloWorldFunction-ohRFEn2RuAvp
```

Invoquez une fonction Lambda en fournissant l'identifiant logique en tant qu'ID de ressource :

Vous devez également fournir le nom de la CloudFormation pile à l'aide de l'`--stack-name`option. Voici un exemple :

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

Si votre application contient une seule fonction Lambda, vous n'avez pas à spécifier son ID logique. Vous pouvez uniquement fournir l'option `--stack-name`. Voici un exemple :

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

Invoquez une fonction Lambda en fournissant l'identifiant physique en tant qu'identifiant de ressource :

L'identifiant physique est créé lorsque vous déployez à l'aide de CloudFormation.

```
$ sam remote invoke sam-app-HelloWorldFunction-TZvxQRFNv0k4
```

Invoquez une fonction Lambda d'une pile enfant :

Pour cet exemple, notre application contient la structure de répertoires suivante :

```
lambda-example
### childstack
#   ### function
#   #   ### __init__.py
#   #   ### app.py
#   #   ### requirements.txt
#   ### template.yaml
### events
#   ### event.json
### samconfig.toml
### template.yaml
```

Pour appeler la fonction Lambda de notre`childstack`, nous exécutons ce qui suit :

```
$ 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\": {}}"}%
```

Configuration d'une fonction Lambda pour le streaming des réponses

Dans cet exemple, nous utilisons la CLI AWS SAM pour initialiser une nouvelle application sans serveur qui contient une fonction Lambda configurée pour diffuser sa réponse. Nous déployons notre application sur le AWS Cloud et l'utilisons `sam remote invoke` pour interagir avec notre fonction dans le cloud.

Nous commençons par exécuter la commande `sam init` pour créer une nouvelle application sans serveur. Nous sélectionnons le modèle de démarrage rapide Lambda Response Streaming et nommons notre application. lambda-streaming-nodejs-app

```
$ 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
```

AWS SAMCLICela crée notre projet avec la structure suivante :

```
lambda-streaming-nodejs-app
 ### README.md
 ### __tests__
 #   ### unit
 #       ### index.test.js
```

```
### package.json
### samconfig.toml
### src
#   ### index.js
### template.yaml
```

Voici un exemple de code de fonction Lambda :

```
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();
   }
 );
```

Voici un exemple de fichier `template.yaml`. Le streaming des réponses pour notre fonction Lambda est configuré à l'aide de la propriété `FunctionUrlConfig`.

```
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
```

Généralement, vous pouvez utiliser `sam build` et `sam deploy --guided` pour créer et déployer une application de production. Dans cet exemple, nous allons utiliser un environnement de développement et utiliser la `sam sync` commande pour créer et déployer notre application.

> **Note**
>
> La commande `sam sync` est recommandée pour les environnements de développement. Pour en savoir plus, veuillez consulter la section Présentation de l'utilisation sam sync de la synchronisation avec AWS Cloud.

Avant de lancer `sam sync`, nous vérifions que notre projet est correctement configuré dans notre fichier `samconfig.toml`. Plus important encore, nous vérifions les valeurs de `stack_name` et `watch`. Ces valeurs étant spécifiées dans notre fichier de configuration, nous n'avons pas à les fournir sur la ligne de commande.

```
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"
```

Ensuite, nous exécutons `sam sync` pour déployer notre application. Comme l'option `--watch` est configurée dans notre fichier de configuration, la CLI AWS SAM créera notre application, la déploiera et surveillera les modifications.

```
$ 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.
```

Maintenant que notre fonction est déployée dans le cloud, nous pouvons utiliser `sam remote invoke` pour interagir avec notre fonction. La CLI AWS SAM détecte automatiquement que notre fonction est configurée pour le streaming des réponses et commence immédiatement à générer une réponse diffusée de notre fonction en temps réel.

```
$ 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
```

Lorsque nous modifions notre code de fonction, la CLI AWS SAM détecte et déploie instantanément nos modifications. Voici un exemple de réponse générée par la CLI AWS SAM après que des modifications ont été apportées à notre code de fonction :

```
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.
```

Nous pouvons à présent utiliser `sam remote invoke` à nouveau pour interagir avec notre fonction dans le cloud et tester nos modifications.

Exemples SQS

Exemples de base

Appelez une file d'attente Amazon SQS en fournissant l'ARN comme ID de ressource :

```
$ 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
  }
}%
```

Exemples de Step Functions

Exemples de base

Invoquez une machine à états en fournissant son identifiant physique en tant qu'identifiant de ressource :

Tout d'abord, nous utilisons `sam list resources` pour obtenir notre identifiant physique :

```
$ 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"
  }
]
```

Ensuite, nous invoquons notre machine à états en utilisant l'identifiant physique comme identifiant de ressource. Nous transmettons un événement en ligne de commande avec l'`--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"%
```

Invoquez une machine à états en transmettant un événement vide :

```
$ sam remote invoke HelloWorldStateMachine --stack-name state-machine-example

Invoking Step Function HelloWorldStateMachine

"Hello World"%
```

Liens connexes

Pour la documentation relative à `sam remote invoke` et à l'utilisation du AWS SAMCLI, consultez les rubriques suivantes :

- sam remote invoke
- Résolution des problèmes de la CLI AWS SAM

Automatisez les tests d'intégration locaux avec AWS SAM

Bien que vous puissiez l'utiliser Présentation des tests avec sam local invoke pour tester le code manuellement, il vous permet AWS SAM également de tester votre code à l'aide de tests d'intégration automatisés. Les tests d'intégration vous aident à détecter les problèmes au début du cycle de développement, à améliorer la qualité de votre code et à gagner du temps tout en réduisant les coûts.

Pour créer des tests d'intégration automatisés dans AWS SAM, vous devez d'abord exécuter des tests sur des fonctions Lambda locales avant de les déployer dans le AWS Cloud. La Présentation des tests avec sam local start-lambda commande démarre un point de terminaison local qui émule le point de terminaison d'appel Lambda. Vous pouvez l'appeler à partir de vos tests automatisés. Comme ce point de terminaison émule le point de terminaison Lambda Invoke, vous pouvez écrire des tests une seule fois, puis les exécuter (sans aucune modification) sur la fonction Lambda locale

ou sur une fonction Lambda déployée. Vous pouvez également exécuter les mêmes tests sur une AWS SAM pile déployée dans votre CI/CD pipeline.

Voici comment fonctionne le processus :

1. Démarrez le point de terminaison Lambda local.

   Démarrez le point de terminaison Lambda local en exécutant la commande suivante dans le répertoire contenant votre AWS SAM modèle :

   ```
   sam local start-lambda
   ```

   Cette commande démarre un point de terminaison local sur `http://127.0.0.1:3001` qui émule AWS Lambda. Vous pouvez exécuter des tests automatisés par rapport à ce point de terminaison Lambda local. Lorsque vous appelez ce point de terminaison à l'aide du SDK AWS CLI ou, il exécute localement la fonction Lambda spécifiée dans la demande et renvoie une réponse.

2. Exécutez un test d'intégration sur le point de terminaison Lambda local.

   Lors de votre test d'intégration, vous pouvez utiliser le AWS SDK pour appeler votre fonction Lambda avec les données de test, attendre une réponse et vérifier que la réponse est conforme à vos attentes. Pour exécuter le test d'intégration en local, vous devez configurer le SDK AWS pour envoyer l'appel d'API de Lambda au point de terminaison Lambda local démarré à l'étape précédente.

   Voici un exemple en Python (les quatre AWS SDKs autres langages ont des configurations similaires) :

   ```
   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"
```

Vous pouvez utiliser ce code pour tester les fonctions Lambda déployées en définissant `running_locally` sur `False`. Cela permet de configurer le AWS SDK auquel se connecter AWS Lambda dans le AWS cloud.

Générez des exemples de charges utiles d'événements avec AWS SAM

Pour tester vos fonctions Lambda, vous pouvez générer et personnaliser des exemples de charges utiles d'événements qui imitent les données que vos fonctions Lambda recevront lorsqu'elles seront déclenchées par d'autres services. AWS Cela inclut des services tels que API Gateway CloudFormation, Amazon S3, etc.

La génération d'exemples de charges utiles d'événements vous permet de tester le comportement de votre fonction Lambda à l'aide de différentes entrées sans avoir à travailler dans un environnement réel. Cette approche permet également de gagner du temps par rapport à la création manuelle d'échantillons d'événements de AWS service pour tester les fonctions.

Pour obtenir la liste complète des services pour lesquels vous pouvez générer des exemples de charges utiles d'événements, utilisez la commande suivante :

```
sam local generate-event --help
```

Pour obtenir la liste des options que vous pouvez utiliser pour un service particulier, utilisez la commande suivante :

```
sam local generate-event [SERVICE] --help
```

Exemples :

```
#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
```

Tester et déboguer des fonctions durables

Le test et le débogage de fonctions durables en local fonctionnent de la même manière que les fonctions Lambda classiques, avec une prise en charge automatique du point de contrôle et de la rediffusion. Ce guide couvre les scénarios de test courants et les techniques de dépannage.

Flux de travail de test local

Lorsque vous testez des fonctions durables localement, le flux de travail diffère des fonctions Lambda classiques :

Flux de travail de test de fonctionnement durable

1. Créez votre application :

   ```
   $ sam build
   ```

2. Invoquez votre fonction durable :

   ```
   $ sam local invoke MyDurableFunction --durable-execution-name test
   ```

3. Vérifiez l'historique des exécutions si nécessaire :

```
$ sam local execution history execution-id
```

Scénarios de test courants

Tester le comportement des points de contrôle

Pour vérifier que votre fonction fonctionne correctement, les points de contrôle indiquent :

```
# 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"}
```

Lorsque vous appelez cette fonction localement, le délai d'attente est géré automatiquement.

Tester des scénarios de rappel

Pour les fonctions qui attendent des rappels externes :

1. Démarrez votre fonction durable qui attend un rappel
2. Dans un autre terminal, résolvez le rappel :

```
$ sam local callback succeed callback-id
```

3. Observez l'exécution de la fonction reprendre

Résolution des problèmes

La fonction durable ne s'exécute pas correctement

Problème : La fonction ne se comporte pas comme une fonction durable.

Solutions :

- Vérifiez que cela `DurableConfig` est défini dans votre modèle SAM

- Assurez-vous que votre code de fonction utilise des méthodes durables du SDK de fonctions (par exemple,`context.wait()`)
- Vérifiez que vous utilisez un environnement d'exécution compatible (TypeScript, JavaScript, Python)

Impossible de récupérer l'historique des exécutions

Problème : La `local execution history` commande ne renvoie aucun résultat.

Solutions :

- Vérifiez que l'ID d'exécution est correct
- Vérifiez que la fonction a été invoquée au moins une fois

Les commandes de rappel ne fonctionnent pas

Problème : les commandes de rappel ne résolvent pas les opérations en attente.

Solutions :

- Vérifiez que l'identifiant de rappel est correct
- Assurez-vous que la fonction attend réellement un rappel
- Vérifiez que vous utilisez la bonne syntaxe de commande de rappel

Conseils de débogage

- Utiliser l'historique des exécutions : passez en revue l'historique des exécutions pour comprendre le flux de votre fonction durable
- Testez de manière incrémentielle : commencez par de simples opérations d'attente avant d'ajouter une logique complexe
- Utiliser la journalisation détaillée - Activez la journalisation détaillée pour suivre le flux d'exécution

En savoir plus

Pour plus d'informations sur les tests et le débogage, voir :

- Présentation des tests avec sam local invoke- Documentation d'appel local
- sam local execution history- Historique d'exécution

Déboguez votre application sans serveur avec AWS SAM

Après avoir testé votre application, vous serez prêt à corriger tous les problèmes que vous aurez découverts. Grâce à l'interface de ligne de AWS SAM commande (CLI), vous pouvez tester et déboguer localement votre application sans serveur avant de la télécharger dans le Cloud. AWS Le débogage de votre application identifie et corrige les problèmes ou les erreurs de votre application.

Vous pouvez l'utiliser AWS SAM pour effectuer un débogage étape par étape, qui consiste à exécuter du code une ligne ou une instruction à la fois. Lorsque vous invoquez localement une fonction Lambda en mode de débogage dans le AWS SAMCLI, vous pouvez alors y associer un débogueur. Avec le débogueur, vous pouvez parcourir votre code ligne par ligne, voir les valeurs des différentes variables et résoudre les problèmes de la même manière que vous le feriez pour n'importe quelle autre application. Vous pouvez vérifier si votre application se comporte comme prévu, déboguer ce qui ne va pas et résoudre les problèmes, avant de passer par les étapes d'empaquetage et de déploiement de votre application.

> **Note**
>
> Si votre application comprend une ou plusieurs couches, lorsque vous exécutez et déboguez localement votre application, le package de couches est téléchargé et mis en cache sur votre hôte local. Pour plus d'informations, consultez Comment les couches sont mises en cache localement.

Rubriques

- Fonctions de débogage locales avec AWS SAM
- Passez plusieurs arguments d'exécution lors du débogage avec AWS SAM
- Validez vos AWS SAM applications avec CloudFormation Linter

Fonctions de débogage locales avec AWS SAM

Vous pouvez utiliser AWS SAM une variété de AWS boîtes à outils et de débogueurs pour tester et déboguer vos applications sans serveur localement. Le débogage progressif de vos fonctions Lambda vous permet d'identifier et de résoudre les problèmes de votre application, ligne par ligne ou instruction, dans votre environnement local.

Vous pouvez notamment effectuer un débogage étape par étape local en définissant des points d'arrêt, en inspectant des variables et en exécutant le code de fonction ligne par ligne. Le débogage par étapes local renforce la boucle de rétroaction, ce qui vous permet de trouver et de résoudre les problèmes que vous pourriez rencontrer dans le cloud.

Vous pouvez utiliser des AWS boîtes à outils pour déboguer, et vous pouvez également exécuter AWS SAM en mode débogage. Consultez les rubriques de cette section pour plus de détails.

Utilisation de AWS boîtes à outils

AWS Les toolkits sont des plugins d'environnement de développement intégré (IDE) qui vous permettent d'effectuer de nombreuses tâches de débogage courantes, telles que la définition de points d'arrêt, l'inspection de variables et l'exécution de code de fonction ligne par ligne. AWS Les boîtes à outils facilitent le développement, le débogage et le déploiement d'applications sans serveur conçues à l'aide de. AWS SAM Elles offrent une expérience de création, de test, de débogage, de déploiement et d'appel de fonctions Lambda qui sont intégrées à votre IDE.

Pour plus d'informations sur les AWS boîtes à outils que vous pouvez utiliser avec AWS SAM, consultez les rubriques suivantes :

- AWS Toolkit for Visual Studio Code
- AWS Cloud9
- AWS Toolkit for JetBrains

Il existe une variété de AWS boîtes à outils qui fonctionnent avec différentes combinaisons IDEs et environnements d'exécution. Le tableau suivant répertorie les IDE/runtime combinaisons courantes qui prennent en charge le débogage progressif des AWS SAM applications :

IDE	Exécution	AWS Boîte à outils	Instructions pour le débogage par étapes
Code Visual Studio	• Node.js • Python • .NET • Java • Go	AWS Toolkit for Visual Studio Code	Utilisation de l'éditeur de balises Application sans serveur AWS dans le Guide de l'utilisateur AWS

IDE	Exécution	AWS Boîte à outils	Instructions pour le débogage par étapes
			Toolkit for Visual Studio Code
AWS Cloud9	• Node.js • Python	AWS Cloud9, avec AWS Toolkit activé [1]	Utilisation d'applications AWS sans serveur à l'aide du AWS kit d'outils figurant dans le guide de AWS Cloud9 l'utilisateur.
WebStorm	Node.js	AWS Toolkit for JetBrains[2]	Exécution (appel) ou débogage d'une fonction locale dans la AWS Toolkit for JetBrains
PyCharm	Python	AWS Toolkit for JetBrains[2]	Exécution (appel) ou débogage d'une fonction locale dans la AWS Toolkit for JetBrains
Rider	.NET	AWS Toolkit for JetBrains[2]	Exécution (appel) ou débogage d'une fonction locale dans la AWS Toolkit for JetBrains
IntelliJ	Java	AWS Toolkit for JetBrains[2]	Exécution (appel) ou débogage d'une fonction locale dans la AWS Toolkit for JetBrains

IDE	Exécution	AWS Boîte à outils	Instructions pour le débogage par étapes
GoLand	Go	AWS Toolkit for JetBrains[2]	Exécution (appel) ou débogage d'une fonction locale dans la AWS Toolkit for JetBrains

Remarques :

1. Pour être utilisé AWS Cloud9 pour déboguer des AWS SAM applications étape par étape, le AWS Toolkit doit être activé. Pour plus d'informations, consultez la section Activation du AWS kit d'outils dans le guide de AWS Cloud9 l'utilisateur.
2. Pour utiliser les AWS SAM applications AWS Toolkit for JetBrains de débogage étape par étape, vous devez d'abord les installer et les configurer en suivant les instructions de la section Installation du AWS Toolkit for JetBrains. AWS Toolkit for JetBrains

Exécution AWS SAM locale en mode debug

Outre l'intégration avec AWS Toolkits, vous pouvez également l'exécuter AWS SAM en mode debug pour vous connecter à des débogueurs tiers tels que ptvsd ou delve.

Pour exécuter AWS SAM en mode débogage, utilisez des commandes sam local invoke ou utilisez sam local start-api l'`-d`option `--debug-port` ou.

Par exemple :

```
# 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**
>
> Si vous utilisez `sam local start-api`, l'instance API Gateway locale expose la totalité de vos fonctions Lambda. Mais, du fait vous ne pouvez spécifier qu'un seul port de débogage,

vous ne pouvez déboguer qu'une seule fonction à la fois. Vous devez appeler votre API avant que la CLI AWS SAM se lie au port, ce qui permet au débogueur de se connecter.

Passez plusieurs arguments d'exécution lors du débogage avec AWS SAM

Vous pouvez choisir de transmettre des arguments d'exécution supplémentaires AWS SAM pour inspecter les problèmes et résoudre les variables de manière plus efficace. Cela permet de renforcer le contrôle et la flexibilité de votre processus de débogage, ce qui peut vous aider à personnaliser les configurations d'exécution et les environnements d'exécution.

Pour transmettre des arguments d'exécution supplémentaires lorsque vous déboguez votre fonction, utilisez la variable d'environnement `DEBUGGER_ARGS`. Cela transmet une chaîne d'arguments directement dans la Run Command que la CLI AWS SAM utilise pour démarrer votre fonction.

Par exemple, si vous souhaitez charger un débogueur tel que i KPdb au moment de l'exécution de votre fonction Python, vous pouvez passer ce qui suit sous la forme. `DEBUGGER_ARGS: -m ikpdb --ikpdb-port=5858 --ikpdb-working-directory=/var/task/ --ikpdb-client-working-directory=/myApp --ikpdb-address=0.0.0.0` Cela le chargerait KPdb au moment de l'exécution avec les autres arguments que vous avez spécifiés.

Dans ce cas, la totalité de votre commande CLI AWS SAM serait :

```
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
```

Vous pouvez transmettre des arguments de débogueur aux fonctions de toutes les exécutions.

Validez vos AWS SAM applications avec CloudFormation Linter

AWS CloudFormation Linter (cfn-lint) est un outil open source que vous pouvez utiliser pour effectuer une validation détaillée de vos modèles. CloudFormation CFN-Lint contient des règles guidées par la spécification de la CloudFormation ressource. Utilisez cfn-lint pour comparer vos ressources à ces règles afin de recevoir des messages détaillés sur les erreurs, les avertissements ou des suggestions informatives. Vous pouvez également créer vos propres règles personnalisées à des fins de validation. Pour en savoir plus sur cfn-lint, consultez cfn-lint dans le dépôt.CloudFormation GitHub

Vous pouvez utiliser cfn-lint pour valider vos modèles AWS Serverless Application Model (AWS SAM) via l'interface de ligne de AWS SAM commande (AWS SAMCLI) en exécutant sam validate l'option. --lint

```
sam validate --lint
```

Pour personnaliser le comportement de cfn-lint, par exemple en créant des règles personnalisées ou en spécifiant des options de validation, vous pouvez définir un fichier de configuration. Pour en savoir plus, consultez la section Fichier de configuration dans le dépôt cfn-lint CloudFormation GitHub . Lorsque vous exécutez sam validate --lint, le comportement de cfn-lint défini dans votre fichier de configuration sera appliqué.

Exemples

Effectuer la validation cfn-lint sur un modèle AWS SAM

```
sam validate --lint --template myTemplate.yaml
```

En savoir plus

Pour en savoir plus sur la commande sam validate, consultez sam validate.

Déployez votre application et vos ressources avec AWS SAM

Le déploiement de votre application fournit et configure vos AWS ressources dans le AWS cloud, ce qui permet à votre application de s'exécuter dans le cloud. AWS SAM utilise CloudFormationcomme mécanisme de déploiement sous-jacent. AWS SAM utilise les artefacts de construction que vous créez lors de l'exécution de la sam build commande comme entrées standard pour le déploiement de votre application sans serveur.

Vous pouvez ainsi déployer votre application sans serveur manuellement ou automatiser les déploiements. AWS SAM Pour automatiser les déploiements, vous utilisez des AWS SAM pipelines dotés d'un système d'intégration et de déploiement continus (CI/CD) de votre choix. Votre pipeline de déploiement est une séquence automatique d'étapes effectuées pour publier une nouvelle version de votre application sans serveur.

Les rubriques de cette section fournissent des conseils sur les déploiements automatisés et manuels. Pour déployer votre application manuellement, vous devez utiliser des AWS SAMCLI commandes. Pour automatiser les déploiements, consultez les rubriques de cette section. Ils fournissent spécifiquement un contenu détaillé sur l'automatisation des déploiements à l'aide de pipelines et d'un CI/CD système. Cela inclut la génération d'un pipeline de démarrage, la configuration de l'automatisation, le dépannage des déploiements, l'utilisation de l'authentification utilisateur OpenID Connect (OIDC) et le téléchargement de fichiers locaux lors du déploiement.

Rubriques

- Présentation du déploiement avec AWS SAM
- Options pour déployer votre application avec AWS SAM
- Utilisation CI/CD de systèmes et de pipelines pour déployer avec AWS SAM
- Présentation de l'utilisation sam sync de la synchronisation avec AWS Cloud

Présentation du déploiement avec AWS SAM

Utilisez la AWS Serverless Application Model commande Command Line Interface (AWS SAMCLI) `sam deploy` pour déployer votre application sans serveur sur. AWS Cloud

- Pour une introduction à la AWS SAMCLI, voirQu'est-ce que c'est AWS SAMCLI ?.

- Pour obtenir la liste des options de commande `sam deploy`, consultez sam deploy.
- Pour un exemple d'utilisation de `sam deploy` dans le cadre d'un flux de travail de développement classique, consultez Étape 3 : Déployez votre application sur AWS Cloud.

Rubriques

- Conditions préalables
- Déploiement d'applications avec sam deploy
- Bonnes pratiques
- Options pour sam deploy
- Résolution des problèmes
- Exemples
- En savoir plus

Conditions préalables

Pour utiliser `sam deploy`, installez la CLI AWS SAM en procédant comme suit :

- AWS SAM prérequis.
- Installer la CLI AWS SAM.

Avant d'utiliser `sam deploy`, nous vous recommandons d'avoir des connaissances de base sur les points suivants :

- Configuration de la CLI AWS SAM.
- Créez votre application dans AWS SAM.
- Initiation à la construction avec AWS SAM.

Déploiement d'applications avec sam deploy

Lorsque vous déployez une application sans serveur pour la première fois, utilisez l'option `--guided`. La CLI AWS SAM vous guidera à travers un flux interactif pour configurer les paramètres de déploiement de votre application.

Pour déployer une application à l'aide du flux interactif

1. Accédez au répertoire racine de votre projet. Il s'agit du même emplacement que celui de votre AWS SAM modèle.

```
$ cd sam-app
```

2. Exécutez la commande suivante :

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

3. Au cours du flux interactif, la CLI AWS SAM vous invite à configurer les paramètres de déploiement de votre application.

   Les crochets (`[ ]`) indiquent les valeurs par défaut. Laissez la réponse vide pour sélectionner la valeur par défaut. Les valeurs par défaut sont obtenues à partir des fichiers de configuration suivants :

   - `~/.aws/config`— Les paramètres généraux de votre AWS compte.
   - `~/.aws/credentials`— Les informations d'identification de votre AWS compte.
   - `<project>/samconfig.toml` : le fichier de configuration de votre projet.

   Fournissez des valeurs en répondant aux invites de la CLI AWS SAM. Par exemple, vous pouvez saisir **y** pour oui, **n** pour non ou des valeurs de chaîne.

   La CLI AWS SAMécrit vos réponses dans le fichier `samconfig.toml` de votre projet. Pour les déploiements suivants, vous pouvez utiliser `sam deploy` pour déployer à l'aide de ces valeurs configurées. Pour reconfigurer ces valeurs, utilisez à nouveau `sam deploy --guided` ou modifiez directement vos fichiers de configuration.

   Voici un exemple de résultat :

```
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. Ensuite, AWS SAMCLI déploie votre application dans le AWS Cloud. Pendant le déploiement, la progression s'affiche dans votre invite de commande. Les principales étapes du déploiement sont les suivantes :

   - Pour les applications dont AWS Lambda les fonctions sont regroupées sous forme d'archive de fichier .zip, le package est AWS SAMCLI compressé et chargé dans un compartiment Amazon Simple Storage Service (Amazon S3). Si nécessaire, la CLI AWS SAM créera un nouveau compartiment.
   - Pour les applications contenant un package de fonctions Lambda en tant qu'image de conteneur, la CLI AWS SAM charge l'image dans Amazon Elastic Container Registry (Amazon ECR). Si nécessaire, la CLI AWS SAM créera un nouveau référentiel.
   - AWS SAMCLIcrée un ensemble de AWS CloudFormation modifications et déploie votre application CloudFormation sous forme de pile.
   - AWS SAMCLImodifie votre AWS SAM modèle déployé avec la nouvelle `CodeUri` valeur de vos fonctions Lambda.

   Vous trouverez ci-dessous un exemple du résultat du déploiement de la CLI AWS SAM :

```
        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. Pour visualiser votre application déployée, procédez comme suit :

   1. Ouvrez la CloudFormation console directement avec l'URL [https://console.aws.amazon.com/cloudformation.](https://console.aws.amazon.com/cloudformation)
   2. Sélectionner Piles
   3. Identifiez votre pile par le nom de l'application et sélectionnez-la.

Vérifiez les modifications avant le déploiement

Vous pouvez configurer le AWS SAMCLI pour afficher votre ensemble de CloudFormation modifications et demander une confirmation avant le déploiement.

Pour confirmer les modifications avant le déploiement

1. Lors de `sam deploy --guided`, saisissez **Y** pour confirmer les modifications avant le déploiement.

   ```
   #Shows you resources changes to be deployed and require a 'Y' to initiate deploy
   Confirm changes before deploy [Y/n]: Y
   ```

   Vous pouvez également modifier votre fichier `samconfig.toml` comme suit :

   ```
   [default.deploy]
   [default.deploy.parameters]
   ```

```
confirm_changeset = true
```

2. Pendant le déploiement, la CLI AWS SAM vous demandera de confirmer les modifications avant le déploiement. Voici un exemple :

```
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
```

Spécifier des paramètres supplémentaires lors du déploiement

Vous pouvez spécifier des valeurs de paramètres supplémentaires à configurer lors du déploiement. Pour ce faire, modifiez votre modèle AWS SAM et configurez la valeur de votre paramètre pendant le déploiement.

Pour spécifier des paramètres supplémentaires

1. Modifiez la `Parameters` section de votre AWS SAM modèle. Voici un exemple :

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Globals:
...
Parameters:
  DomainName:
    Type: String
    Default: example
    Description: Domain name
```

2. Exécutez `sam deploy --guided`. Voici un exemple de résultat :

```
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
```

Configuration de la signature de code pour vos fonctions Lambda

Vous pouvez configurer la signature de code pour vos fonctions Lambda au moment du déploiement. Pour ce faire, modifiez votre AWS SAM modèle et configurez la signature de code lors du déploiement.

Pour configurer la signature de code

1. Spécifiez `CodeSigningConfigArn` dans votre AWS SAM modèle. Voici un exemple :

```
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. Exécutez `sam deploy --guided`. La CLI AWS SAM vous invite à configurer la signature de code. Voici un exemple de résultat :

```
#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):
```

Bonnes pratiques

- Lors de l'utilisation de `sam deploy`, la AWS SAM CLI déploie les artefacts de création de votre application situés dans le répertoire `.aws-sam`. Lorsque vous apportez des modifications aux

fichiers d'origine de votre application, exécutez `sam build` pour mettre à jour le répertoire `.aws-sam` avant de procéder au déploiement.

- Lorsque vous déployez une application pour la première fois, utilisez `sam deploy --guided` pour configurer les paramètres de déploiement. Pour les déploiements suivants, vous pouvez utiliser `sam deploy` pour déployer avec vos paramètres configurés.

Options pour sam deploy

Les options suivantes sont couramment utilisées pour `sam deploy`. Pour obtenir la liste de toutes les options, consultez sam deploy.

Utiliser le flux interactif guidé pour déployer votre application

Utilisez l'option `--guided` pour configurer les paramètres de déploiement de votre application à travers un flux interactif. Voici un exemple :

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

Les paramètres de déploiement de votre application sont enregistrés dans le fichier `samconfig.toml` de votre projet. Pour en savoir plus, veuillez consulter la section Configuration des paramètres de projet.

Résolution des problèmes

Pour résoudre le problème AWS SAMCLI, voirRésolution des problèmes de la CLI AWS SAM.

Exemples

Déployer une application Hello World qui contient une fonction Lambda packagée sous la forme d'une archive de fichier .zip

Pour un exemple, consultez Étape 3 : Déployez votre application sur AWS Cloud dans le didacticiel de l'application Hello World.

Déployer une application Hello World qui contient une fonction Lambda packagée en tant qu'image de conteneur

Tout d'abord, nous l'utilisons `sam init` pour créer notre application Hello World. Au cours du flux interactif, nous sélectionnons l'exécution `Python3.9` et le type de package `Image`.

```
$ 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
...
```

Ensuite, nous `cd` vers le répertoire racine de notre projet et exécutons `sam build`. La CLI AWS SAM crée notre fonction Lambda localement en utilisant Docker.

```
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
```

Ensuite, nous exécutons `sam deploy --guided` pour déployer notre application. La CLI AWS SAM nous guide dans la configuration de nos paramètres de déploiement. Ensuite, AWS SAMCLI déploie notre application dans le AWS Cloud.

```
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
```

En savoir plus

Pour en savoir plus sur l'utilisation de AWS SAMCLI `sam deploy` cette commande, consultez les rubriques suivantes :

- L' AWS SAM atelier complet : Module 3 - Déploiement manuel — Découvrez comment créer, empaqueter et déployer une application sans serveur à l'aide du AWS SAMCLI.

Options pour déployer votre application avec AWS SAM

Avec AWS SAM, vous pouvez déployer votre application manuellement et automatiser les déploiements. Utilisez le AWS SAMCLI pour déployer manuellement votre application. Pour automatiser le déploiement, utilisez des pipelines et un système d'intégration et de déploiement continus (CI/CD). Les rubriques de cette section fournissent des informations sur les deux approches.

Rubriques

- Comment utiliser le AWS SAMCLI pour déployer manuellement
- Déployez avec CI/CD des systèmes et des pipelines
- Déploiements graduels
- Résolution de problèmes de déploiement à l'aide de la CLI AWS SAM
- En savoir plus

Comment utiliser le AWS SAMCLI pour déployer manuellement

Après avoir développé et testé votre application sans serveur localement, vous pouvez déployer votre application à l'aide de la commande sam deploy.

Pour vous AWS SAM guider tout au long du déploiement à l'aide d'instructions, spécifiez le --guided drapeau. Lorsque vous spécifiez cet indicateur, la commande sam deploy compresse vos artefacts d'application, les télécharge soit sur Amazon Simple Storage Service (Amazon S3) (pour les archives de fichiers .zip), soit sur Amazon Elastic Container Registry (Amazon ECR) (pour les images de conteneur). La commande déploie ensuite votre application dans le AWS Cloud.

Exemple :

```
# Deploy an application using prompts:
sam deploy --guided
```

Déployez avec CI/CD des systèmes et des pipelines

AWS SAM vous aide à automatiser le déploiement à l'aide de pipelines et d'un système d'intégration et de déploiement continus (CI/CD). AWS SAM peut être utilisé pour créer des pipelines et simplifier CI/CD les tâches des applications sans serveur. Plusieurs CI/CD systèmes prennent en charge la AWS SAM création d'images de conteneurs et fournissent AWS SAM également un ensemble de modèles de pipeline par défaut pour plusieurs CI/CD systèmes qui encapsulent les meilleures pratiques AWS de déploiement.

Pour de plus amples informations, veuillez consulter Utilisation CI/CD de systèmes et de pipelines pour déployer avec AWS SAM.

Déploiements graduels

Si vous souhaitez déployer votre AWS SAM application progressivement plutôt qu'en une seule fois, vous pouvez spécifier les configurations de déploiement qui AWS CodeDeploy fournissent. Pour plus d'informations, consultez la section Utilisation des configurations de déploiement CodeDeploy dans le Guide de AWS CodeDeploy l'utilisateur.

Pour plus d'informations sur la configuration de votre AWS SAM application pour un déploiement progressif, consultezDéploiement progressif d'applications sans serveur avec AWS SAM.

Résolution de problèmes de déploiement à l'aide de la CLI AWS SAM

Erreur CLI AWS SAM : « Contraintes de sécurité non satisfaites »

Lors de l'exécution de sam deploy --guided, vous devrez répondre à la question `HelloWorldFunction may not have authorization defined, Is this okay? [y/N]`. Si vous répondez à cette question par non **N** (réponse par défaut), vous voyez apparaître l'erreur suivante :

```
Error: Security Constraints Not Satisfied
```

L'invite vous informe que l'application que vous êtes sur le point de déployer peut comporter une API Amazon API Gateway configurée sans autorisation. En répondant **N** à cette question, vous dites que ce n'est pas OK.

Pour résoudre le problème, vous disposez des options suivantes :

- Configurez votre application avec l'autorisation. Pour plus d'informations sur la configuration de l'autorisation, consultez Contrôlez l'accès aux API avec votre AWS SAM modèle.
- Répondez à cette question par **Y** pour indiquer que vous êtes d'accord avec le déploiement d'une application sur laquelle une API API Gateway est configurée sans autorisation.

En savoir plus

Pour des exemples pratiques de déploiement d'applications sans serveur, consultez le document suivant tiré de The Complete AWS SAM Workshop :

- Module 3 - Déploiement manuel — Découvrez comment créer, empaqueter et déployer une application sans serveur à l'aide du AWS SAMCLI.
- Module 4 – CI/CD : découvrez comment automatiser les phases de création, de package et de déploiement en créant un pipeline d'intégration et de livraison continues (CI/CD).

Utilisation CI/CD de systèmes et de pipelines pour déployer avec AWS SAM

AWS SAM aide les entreprises à créer des pipelines pour leurs CI/CD systèmes préférés, afin qu'elles puissent en tirer parti CI/CD avec un minimum d'efforts, tels que l'accélération de la fréquence des déploiements, la réduction des délais de modification et la réduction des erreurs de déploiement.

AWS SAM simplifie CI/CD les tâches des applications sans serveur à l'aide de la création d'images de conteneur. Les images AWS SAM fournies incluent les outils AWS SAMCLI et les outils de génération pour un certain nombre d'environnements d' AWS Lambda exécution pris en charge. Cela facilite la création et le package d'applications sans serveur à l'aide du AWS SAMCLI. Ces images évitent également aux équipes de créer et de gérer leurs propres images pour les CI/CD systèmes. Pour plus d'informations sur la AWS SAM création d'images de conteneurs, consultezRéférentiels d'images pour AWS SAM.

Plusieurs CI/CD systèmes prennent en charge la AWS SAM création d'images de conteneurs. CI/CD Le système à utiliser dépend de plusieurs facteurs. Il s'agit notamment de savoir si votre application utilise une seule ou plusieurs exécutions, ou si vous souhaitez créer votre application dans une image de conteneur ou directement sur une machine hôte, une machine virtuelle (VM) ou un hôte de matériel nu.

AWS SAM fournit également un ensemble de modèles de pipeline par défaut pour plusieurs CI/CD systèmes qui encapsulent les meilleures pratiques AWS de déploiement. Ces modèles de pipeline par défaut utilisent des formats de configuration de JSON/YAML pipeline standard, et les meilleures pratiques intégrées permettent d'effectuer des déploiements multicomptes et multirégions, et de vérifier que les pipelines ne peuvent pas apporter de modifications involontaires à l'infrastructure.

Vous disposez de deux options principales AWS SAM pour déployer vos applications sans serveur : 1) Modifier la configuration de votre pipeline existante pour utiliser des AWS SAMCLI commandes, ou 2) Générer un exemple de configuration de CI/CD pipeline que vous pouvez utiliser comme point de départ pour votre propre application.

Rubriques

- Qu'est-ce qu'un pipeline ?
- Comment AWS SAM télécharge les fichiers locaux lors du déploiement
- Générez un CI/CD pipeline de démarrage avec AWS SAM
- Comment personnaliser les pipelines de démarrage avec AWS SAM
- Automatisez le déploiement de votre AWS SAM application
- Comment utiliser l'authentification OIDC avec les pipelines AWS SAM

Qu'est-ce qu'un pipeline ?

Un pipeline est une séquence automatisée d'étapes effectuées pour publier une nouvelle version d'une application. Vous pouvez utiliser de nombreux CI/CD systèmes courants pour déployer vos applications, notamment Jenkins AWS CodePipeline, GitLab CI/CD et Actions. AWS SAM GitHub

Les modèles de pipeline incluent les meilleures pratiques de AWS déploiement pour faciliter les déploiements multicomptes et multirégions. AWS les environnements tels que le développement et la production existent généralement dans AWS des comptes différents. Cela permet aux équipes de développement de configurer des pipelines de déploiement sécurisés, sans apporter de modifications involontaires à l'infrastructure.

Vous pouvez également fournir vos propres modèles de pipeline personnalisés pour aider à standardiser les pipelines au sein des équipes de développement.

Comment AWS SAM télécharge les fichiers locaux lors du déploiement

Lorsque vous déployez votre application sur le AWS Cloud, AWS CloudFormation vos fichiers locaux doivent d'abord être téléchargés vers un AWS service accessible, tel qu'Amazon Simple Storage Service (Amazon S3). Cela inclut les fichiers locaux auxquels votre AWS SAM modèle fait référence. Pour répondre à cette exigence, AWS SAMCLI procédez comme suit lorsque vous utilisez la `sam package` commande `sam deploy` ou :

1. Télécharge automatiquement vos fichiers locaux vers un AWS service accessible.
2. Met automatiquement à jour votre modèle d'application pour faire référence au nouveau chemin de fichier.

Rubriques

- Démo : utilisation de la CLI AWS SAM pour charger le code d'une fonction Lambda
- Cas d'utilisation pris en charge
- En savoir plus

Démo : utilisation de la CLI AWS SAM pour charger le code d'une fonction Lambda

Dans cette démo, nous initialisons l'exemple d'application Hello World en utilisant un type de package .zip pour notre fonction Lambda. Nous utilisons la CLI AWS SAM pour charger automatiquement notre code de fonction Lambda dans Amazon S3 et référencer son nouveau chemin dans notre modèle d'application.

Tout d'abord, nous exécutons `sam init` pour initialiser notre application Hello World.

```
$ 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

...
```

Le code de notre fonction Lambda est organisé dans le sous-répertoire `hello_world` de notre projet.

```
demo
### README.md
### hello_world
#   ### __init__.py
#   ### app.py
#   ### requirements.txt
### template.yaml
### tests
```

Dans notre AWS SAM modèle, nous référençons le chemin local vers notre code de fonction Lambda à l'aide de la `CodeUri` propriété.

```
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
      ...
```

Ensuite, nous exécutons `sam build` pour créer notre application et préparer son déploiement.

```
$ 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
...
```

Ensuite, nous exécutons `sam deploy --guided` pour déployer notre application.

```
$ 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
```

Lors du déploiement, la CLI AWS SAM charge automatiquement notre code de fonction Lambda dans Amazon S3 et met à jour notre modèle. Notre modèle modifié dans la CloudFormation console reflète le chemin du compartiment Amazon S3.

```
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
      ...
```

Cas d’utilisation pris en charge

Ils AWS SAMCLI peuvent automatiquement faciliter ce processus pour un certain nombre de types de fichiers, de types de CloudFormation ressources et de CloudFormation macros.

Types de fichiers

Les fichiers d'application et les images Docker sont pris en charge.

CloudFormation types de ressources

Vous trouverez ci-dessous une liste des types de ressources pris en charge et de leurs propriétés :

Ressource	Propriétés
`AWS::ApiGateway::RestApi`	`BodyS3Location`
`AWS::ApiGatewayV2::Api`	`BodyS3Location`

`AWS::AppSync::FunctionConfiguration`	`CodeS3Location`
	`RequestMappingTemplateS3Location`
	`ResponseMappingTemplateS3Location`

`AWS::AppSync::GraphQLSchema`	`DefinitionS3Location`

`AWS::AppSync::Resolver`	`CodeS3Location`
	`RequestMappingTemplateS3Location`
	`ResponseMappingTemplateS3Location`

`AWS::CloudFormation::ModuleVersion`	`ModulePackage`
`AWS::CloudFormation::ResourceVersion`	`SchemaHandlerPackage`

Ressource	Propriétés
`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`
`AWS::StepFunctions::StateMachine`	`DefinitionS3Location`

CloudFormation macros

Les fichiers référencés à l'aide de la macro `AWS::Include` transform sont pris en charge.

En savoir plus

Pour en savoir plus sur la `AWS::Include` transformation, voir AWS::Include transformation dans le guide de AWS CloudFormation l'utilisateur.

Pour voir un exemple d'utilisation de la `AWS::Include` transformation dans un AWS SAM modèle, consultez le modèle API Gateway HTTP API to SQS sur Serverless Land.

Générez un CI/CD pipeline de démarrage avec AWS SAM

Lorsque vous êtes prêt à automatiser le déploiement, vous pouvez utiliser l'un des modèles AWS SAM de pipeline de démarrage pour générer un pipeline de déploiement pour le CI/CD système que vous choisissez d'utiliser. Votre pipeline de déploiement est ce que vous configurez et utilisez pour automatiser le déploiement de votre application sans serveur. Un modèle de pipeline de démarrage est préconfiguré pour vous aider à configurer rapidement votre pipeline de déploiement pour votre application sans serveur.

Avec un modèle de pipeline de démarrage, vous pouvez générer des pipelines en quelques minutes à l'aide de la sam pipeline init commande.

Les modèles de pipeline de démarrage utilisent la JSON/YAML syntaxe familière du CI/CD système et intègrent les meilleures pratiques, telles que la gestion des artefacts sur plusieurs comptes et régions, et l'utilisation du minimum d'autorisations requises pour déployer l'application. Actuellement, la AWS SAM CLI prend en charge la génération de configurations de CI/CD pipeline de démarrage pour les AWS CodePipelinepipelines Jenkins, GitLab CI/CD, GitHub Actions et Bitbucket.

Voici les tâches de haut niveau que vous devez effectuer pour générer une configuration de pipeline de démarrage :

1. Création de ressources d'infrastructure : votre pipeline nécessite certaines AWS ressources, par exemple l'utilisateur IAM et les rôles dotés des autorisations nécessaires, un compartiment Amazon S3 et éventuellement un référentiel Amazon ECR.
2. Connectez votre dépôt Git à votre CI/CD système : votre CI/CD système doit savoir quel dépôt Git déclenchera l'exécution du pipeline. Notez que cette étape peut ne pas être nécessaire, selon la combinaison du référentiel Git et CI/CD du système que vous utilisez.
3. Générez une configuration de pipeline : cette étape génère une configuration de pipeline de démarrage qui inclut deux étapes de déploiement.

4. Validez la configuration de votre pipeline dans votre dépôt Git : cette étape est nécessaire pour s'assurer que votre CI/CD système est au courant de la configuration de votre pipeline et qu'elle sera exécutée lorsque les modifications seront validées.

Après avoir généré la configuration du pipeline de démarrage et l'avoir associée au référentiel Git, le pipeline se déclenche pour s'exécuter automatiquement chaque fois que quelqu'un associe une modification de code au référentiel.

L'ordre de ces étapes, ainsi que les détails de chaque étape, varient en fonction du système CI/CD :

- Si vous utilisez AWS CodePipeline, consultezGénération d'un pipeline de démarrage pour AWS CodePipeline in AWS SAM.
- Si vous utilisez Jenkins, GitLab CI/CD, GitHub Actions ou Bitbucket Pipelines, consultez. AWS SAM À utiliser pour générer des pipelines de démarrage pour Jenkins, GitLab CI/CD, GitHub Actions, Bitbucket Pipelines

Génération d'un pipeline de démarrage pour AWS CodePipeline in AWS SAM

Pour générer une configuration de pipeline de démarrage pour AWS CodePipeline, effectuez les tâches suivantes dans cet ordre :

1. Créer des ressources d'infrastructure
2. Générer la configuration du pipeline
3. Associer la configuration du pipeline au référentiel Git
4. Connectez votre dépôt Git à votre CI/CD système

> **Note**
>
> La procédure suivante utilise deux commandes de la CLI AWS SAM, `sam pipeline bootstrap` et `sam pipeline init`. La raison pour laquelle il existe deux commandes est de gérer le cas d'utilisation où les administrateurs (c'est-à-dire les utilisateurs qui ont besoin d'une autorisation pour configurer des AWS ressources d'infrastructure telles que les utilisateurs et les rôles IAM) ont plus d'autorisations que les développeurs (c'est-à-dire les utilisateurs qui ont juste besoin d'une autorisation pour configurer des pipelines individuels, mais pas les AWS ressources d'infrastructure requises).

Étape 1 : créer des ressources d'infrastructure

Les pipelines utilisés AWS SAM nécessitent certaines AWS ressources, comme un utilisateur IAM et des rôles dotés des autorisations nécessaires, un compartiment Amazon S3 et éventuellement un référentiel Amazon ECR. Vous devez disposer d'un ensemble de ressources d'infrastructure pour chaque étape de déploiement du pipeline.

Vous pouvez exécuter la commande suivante pour vous aider avec cette configuration :

```
sam pipeline bootstrap
```

> **Note**
>
> Exécutez la commande précédente pour chaque étape de déploiement du pipeline.

Étape 2 : générer la configuration du pipeline

Pour générer la configuration du pipeline, exécutez la commande suivante :

```
sam pipeline init
```

Étape 3 : associer la configuration du pipeline au référentiel Git

Cette étape est nécessaire pour s'assurer que votre CI/CD système connaît la configuration de votre pipeline et qu'elle sera exécutée lorsque les modifications seront validées.

Étape 4 : Connectez votre dépôt Git à votre CI/CD système

Car AWS CodePipeline vous pouvez désormais créer la connexion en exécutant la commande suivante :

```
sam deploy -t codepipeline.yaml --stack-name <pipeline-stack-name> --
capabilities=CAPABILITY_IAM --region <region-X>
```

Si vous utilisez GitHub Bitbucket, après avoir exécuté la sam deploy commande précédemment, terminez la connexion en suivant les étapes décrites dans la section Pour établir une connexion trouvée dans la rubrique Mettre à jour une connexion en attente dans le guide de l'utilisateur de la console Developer Tools. En outre, stockez une copie `CodeStarConnectionArn` de la sortie de la

sam deploy commande, car vous en aurez besoin si vous souhaitez l'utiliser AWS CodePipeline avec une autre branche que`main`.

Configuration d'autres branches

Par défaut, AWS CodePipeline utilise la `main` branche avec AWS SAM. Si vous voulez utiliser une branche autre que `main`, vous devez exécuter à nouveau la commande sam deploy. Notez que selon le référentiel Git que vous utilisez, il se peut que vous deviez aussi fournir le `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>"
```

En savoir plus

Pour un exemple pratique de configuration d'un CI/CD pipeline, voir CI/CD with AWS CodePipeline dans The Complete AWS SAM Workshop.

AWS SAM À utiliser pour générer des pipelines de démarrage pour Jenkins, GitLab CI/CD, GitHub Actions, Bitbucket Pipelines

Pour générer une configuration de pipeline de démarrage pour les pipelines Jenkins, GitLab CI/CD, GitHub Actions ou Bitbucket, effectuez les tâches suivantes dans cet ordre :

1. Créer des ressources d'infrastructure
2. Connectez votre dépôt Git à votre CI/CD système
3. Créer des objets d'identification
4. Générer la configuration du pipeline
5. Associer la configuration du pipeline au référentiel Git

> **Note**
>
> La procédure suivante utilise deux commandes de la CLI AWS SAM, `sam pipeline bootstrap` et `sam pipeline init`. La raison pour laquelle il existe deux commandes est de gérer le cas d'utilisation où les administrateurs (c'est-à-dire les utilisateurs qui ont besoin d'une autorisation pour configurer des AWS ressources d'infrastructure telles que les utilisateurs et les rôles IAM) ont plus d'autorisations que les développeurs (c'est-à-dire les utilisateurs qui ont juste besoin d'une autorisation pour configurer des pipelines individuels, mais pas les AWS ressources d'infrastructure requises).

Étape 1 : créer des ressources d'infrastructure

Les pipelines utilisés AWS SAM nécessitent certaines AWS ressources, comme un utilisateur IAM et des rôles dotés des autorisations nécessaires, un compartiment Amazon S3 et éventuellement un référentiel Amazon ECR. Vous devez disposer d'un ensemble de ressources d'infrastructure pour chaque étape de déploiement du pipeline.

Vous pouvez exécuter la commande suivante pour vous aider avec cette configuration :

```
sam pipeline bootstrap
```

> **Note**
>
> Exécutez la commande précédente pour chaque étape de déploiement du pipeline.

Vous devez saisir les AWS informations d'identification (identifiant de clé et clé secrète) des utilisateurs du pipeline pour chaque étape de déploiement de votre pipeline, car elles sont nécessaires pour les étapes suivantes.

Étape 2 : Connectez votre dépôt Git à votre CI/CD système

La connexion de votre dépôt Git à votre CI/CD système est nécessaire pour que le CI/CD système puisse accéder au code source de votre application pour les builds et les déploiements.

> **Note**
>
> Vous pouvez ignorer cette étape si vous utilisez l'une des combinaisons suivantes, car la connexion est effectuée automatiquement :
>
> 1. GitHub Actions avec le GitHub référentiel
> 2. GitLab CI/CD avec dépôt GitLab
> 3. Bitbucket Pipelines avec un référentiel Bitbucket

Pour connecter votre dépôt Git à votre CI/CD système, effectuez l'une des opérations suivantes :

- Si vous utilisez Jenkins, consultez « Ajout d'une source de branche » dans la section Documentation Jenkins.
- Si vous utilisez GitLab CI/CD et un dépôt Git autre que celui-ci GitLab, consultez la GitLabdocumentation relative à la « connexion à un dépôt externe ».

Étape 3 : créer des objets des informations d'identification

Chaque CI/CD système dispose de sa propre méthode de gestion des informations d'identification nécessaires pour accéder à votre dépôt Git. CI/CD

Pour créer les objets des informations d'identification nécessaires, effectuez l'une des opérations suivantes :

- Si vous utilisez Jenkins, créez une seule « information d'identification » qui stocke à la fois l'identifiant de la clé et la clé secrète. Suivez les instructions du blog Création d'un pipeline Jenkins avec AWS SAM, dans la section Configurer Jenkins. Vous aurez besoin de l'« identifiant des informations d'identification » pour l'étape suivante.
- Si vous utilisez le GitLab CI/CD, créez deux « variables protégées », une pour chaque identifiant de clé et une clé secrète. Suivez les instructions de la GitLab documentation. Vous aurez besoin de deux « clés variables » pour l'étape suivante.
- Si vous utilisez GitHub Actions, créez deux « secrets chiffrés », un pour chaque clé et une pour chaque clé secrète. Suivez les instructions de la GitHubdocumentation. Vous aurez besoin de deux « noms secrets » pour l'étape suivante.

- Si vous utilisez Bitbucket Pipelines, créez deux « variables sécurisées », une pour chaque identifiant de clé et clé secrète. Suivez les instructions de la section Variables et secrets . Vous aurez besoin de deux « noms secrets » pour l'étape suivante.

Étape 4 : générer la configuration du pipeline

Pour générer la configuration du pipeline, exécutez la commande suivante : Vous devez entrer l'objet des informations d'identification que vous avez créé à l'étape précédente :

```
sam pipeline init
```

Étape 5 : associer la configuration du pipeline au référentiel Git

Cette étape est nécessaire pour s'assurer que votre CI/CD système connaît la configuration de votre pipeline et qu'elle sera exécutée lorsque les modifications seront validées.

En savoir plus

Pour un exemple pratique de configuration d'un CI/CD pipeline en utilisantGitHub Actions, voir CI/CD with GitHub dans The Complete AWS SAM Workshop.

Comment personnaliser les pipelines de démarrage avec AWS SAM

En tant qu' CI/CD administrateur, vous souhaiterez peut-être personnaliser un modèle de pipeline de démarrage, ainsi que les instructions guidées associées, que les développeurs de votre organisation peuvent utiliser pour créer des configurations de pipeline.

La CLI AWS SAM utilise des modèles Cookiecutter lors de la création des modèles de démarrage. Pour en savoir plus sur les modèles cookie cutter, consultez Cookiecutter.

Vous pouvez également personnaliser les invites que la CLI AWS SAM affiche aux utilisateurs lors de la création de configurations de pipeline à l'aide de la commande `sam pipeline init`. Pour personnaliser les invites des utilisateurs, procédez comme suit :

1. Créez un fichier **`questions.json`** – Le fichier `questions.json` doit être à la racine du référentiel du projet. Il s'agit du même répertoire que pour le fichier `cookiecutter.json`. Pour afficher le schéma du fichier `questions.json`, consultez questions.json.schema. Pour afficher un exemple de fichier `questions.json`, consultez questions.json.
2. Mapper les clés de questions avec les noms des coupeurs de cookies – Chaque objet du fichier `questions.json` nécessite une clé qui correspond à un nom dans le modèle Cookiecutter. Cette

correspondance de clé est la façon dont la CLI AWS SAM met en correspondance les réponses des utilisateurs au modèle de cookie cutter. Pour voir des exemples de correspondance de cette clé, consultez la section Exemples de fichier plus loin dans cette rubrique.

3. Créez un fichier **`metadata.json`** – Déclarez le nombre d'étapes que le pipeline aura dans le fichier `metadata.json`. Le nombre d'étapes indique à la commande `sam pipeline init` le nombre d'étapes nécessaire pour fournir des informations ou, dans le cas de l'option `--bootstrap`, le nombre d'étapes pour lesquelles créer des ressources d'infrastructure. Pour accéder à un exemple de fichier `metadata.jsonqui` déclare un pipeline à deux étapes, consultez metadata.json.

Exemples de projets

Voici des exemples de projets, qui comprennent chacun un modèle Cookiecutter, un fichier `questions.json` et un fichier `metadata.json` :

- Exemple de Jenkins : Modèle de pipeline Jenkins en deux étapes
- CodePipeline exemple : modèle de CodePipeline pipeline en deux étapes

Exemples de fichier

L'ensemble de fichiers suivant montre comment les questions dans le fichier `questions.json` sont associées aux entrées du fichier de modèle Cookiecutter. Notez que ces exemples sont des extraits de fichiers, non des fichiers complets. Pour voir des exemples de fichiers complets, consultez la section Exemples de projets plus loin dans cette rubrique.

Exemple **`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"
  }, {
    ...
```

Exemple **`cookiecutter.json`** :

```
{
  "outputDir": "aws-sam-pipeline",
  "pipeline_user_jenkins_credential_id": "",
  "sam_template": "",
    ...
```

Exemple **`Jenkinsfile`** :

```
pipeline {
  agent any
  environment {
    PIPELINE_USER_CREDENTIAL_ID =
 '{{cookiecutter.pipeline_user_jenkins_credential_id}}'
    SAM_TEMPLATE = '{{cookiecutter.sam_template}}'
    ...
```

Automatisez le déploiement de votre AWS SAM application

Dans AWS SAM, la manière dont vous automatisez le déploiement de votre AWS SAM application varie en fonction du CI/CD système que vous utilisez. C'est pourquoi les exemples présentés dans cette section vous montrent comment configurer différents CI/CD systèmes pour automatiser la création d'applications sans serveur dans une image de conteneur de AWS SAM génération. Ces images de conteneur de génération facilitent la création et le package d'applications sans serveur à l'aide du AWS SAMCLI.

Les procédures de votre CI/CD pipeline existant pour déployer des applications sans serveur AWS SAM sont légèrement différentes selon le CI/CD système que vous utilisez.

Les rubriques suivantes fournissent des exemples de configuration de votre CI/CD système pour créer des applications sans serveur dans une image de conteneur de AWS SAM génération :

Rubriques

- Utilisation AWS CodePipeline pour déployer avec AWS SAM

- Utilisation de Bitbucket Pipelines pour déployer avec AWS SAM
- Utiliser Jenkins pour déployer avec AWS SAM
- Utilisation de GitLab CI/CD pour déployer avec AWS SAM
- Utiliser GitHub des actions pour déployer avec AWS SAM

Utilisation AWS CodePipeline pour déployer avec AWS SAM

Pour configurer votre AWS CodePipelinepipeline afin d'automatiser la création et le déploiement de votre AWS SAM application, votre CloudFormation modèle et votre `buildspec.yml` fichier doivent contenir les lignes suivantes :

1. Référencer une image de conteneur de création avec l'exécution nécessaire à partir des images disponibles. L'exemple suivant utilise l'image de conteneur de création `public.ecr.aws/sam/build-nodejs20.x`.
2. Configurez les étapes du pipeline pour exécuter les commandes d'interface de ligne de AWS SAM commande (CLI) nécessaires. L'exemple suivant exécute deux commandes de la CLI AWS SAM : sam build et sam deploy (avec les options nécessaires).

Cet exemple suppose que vous avez déclaré toutes les fonctions et couches de votre fichier AWS SAM modèle avec`runtime: nodejs20.x`.

CloudFormation extrait de modèle :

```
  CodeBuildProject:
    Type: AWS::CodeBuild::Project
    Properties:
      Environment:
        ComputeType: BUILD_GENERAL1_SMALL
        Image: public.ecr.aws/sam/build-nodejs20.x
        Type: LINUX_CONTAINER
      ...
```

`buildspec.yml` extrait :

```
version: 0.2
phases:
  build:
    commands:
```

```
      - sam build
      - sam deploy --no-confirm-changeset --no-fail-on-empty-changeset
```

Pour obtenir la liste des images de conteneur de création Amazon Elastic Container Registry (Amazon ECR) disponibles pour différentes exécutions, consultez Référentiels d'images pour AWS SAM.

Utilisation de Bitbucket Pipelines pour déployer avec AWS SAM

Pour configurer votre pipeline Bitbucket afin d'automatiser la création et le déploiement de votre AWS SAM application, votre `bitbucket-pipelines.yml` fichier doit contenir les lignes suivantes :

1. Référencer une image de conteneur de création avec l'exécution nécessaire à partir des images disponibles. L'exemple suivant utilise l'image de conteneur de création `public.ecr.aws/sam/build-nodejs20.x`.
2. Configurez les étapes du pipeline pour exécuter les commandes d'interface de ligne de AWS SAM commande (CLI) nécessaires. L'exemple suivant exécute deux commandes de la CLI AWS SAM : sam build et sam deploy (avec les options nécessaires).

Cet exemple suppose que vous avez déclaré toutes les fonctions et couches de votre fichier AWS SAM modèle avec`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
```

Pour obtenir la liste des images de conteneur de construction Amazon Elastic Container Registry (Amazon ECR) disponibles pour différentes exécutions, consultez Référentiels d'images pour AWS SAM.

Utiliser Jenkins pour déployer avec AWS SAM

Pour configurer votre pipeline Jenkins afin d'automatiser la création et le déploiement de votre AWS SAM application, vous `Jenkinsfile` devez contenir des lignes qui effectuent les opérations suivantes :

1. Référencer une image de conteneur de création avec l'exécution nécessaire à partir des images disponibles. L'exemple suivant utilise l'image de conteneur de création `public.ecr.aws/sam/build-nodejs20.x`.
2. Configurez les étapes du pipeline pour exécuter les commandes d'interface de ligne de AWS SAM commande (CLI) nécessaires. L'exemple suivant exécute deux commandes de la CLI AWS SAM : sam build et sam deploy (avec les options nécessaires).

Cet exemple suppose que vous avez déclaré toutes les fonctions et couches de votre fichier AWS SAM modèle avec`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'
            }
        }
    }
}
```

Pour obtenir la liste des images de conteneur de construction Amazon Elastic Container Registry (Amazon ECR) disponibles pour différentes exécutions, consultez Référentiels d'images pour AWS SAM.

Utilisation de GitLab CI/CD pour déployer avec AWS SAM

Pour configurer votre GitLabpipeline afin d'automatiser la création et le déploiement de votre AWS SAM application, votre `gitlab-ci.yml` fichier doit contenir les lignes suivantes :

1. Référencer une image de conteneur de création avec l'exécution nécessaire à partir des images disponibles. L'exemple suivant utilise l'image de conteneur de création `public.ecr.aws/sam/build-nodejs20.x`.

2. Configurez les étapes du pipeline pour exécuter les commandes d'interface de ligne de AWS SAM commande (CLI) nécessaires. L'exemple suivant exécute deux commandes de la CLI AWS SAM : sam build et sam deploy (avec les options nécessaires).

Cet exemple suppose que vous avez déclaré toutes les fonctions et couches de votre fichier AWS SAM modèle avec`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
```

Pour obtenir la liste des images de conteneur de construction Amazon Elastic Container Registry (Amazon ECR) disponibles pour différentes exécutions, consultez Référentiels d'images pour AWS SAM.

Utiliser GitHub des actions pour déployer avec AWS SAM

Pour configurer votre GitHubpipeline afin d'automatiser la création et le déploiement de votre AWS SAM application, vous devez d'abord installer l'interface de ligne de AWS SAM commande (CLI) sur votre hôte. Vous pouvez utiliser GitHub les actions dans votre GitHub flux de travail pour faciliter cette configuration.

L'exemple de GitHub flux de travail suivant configure un hôte Ubuntu à l'aide d'une série d' GitHubactions, puis exécute des AWS SAMCLI commandes pour créer et déployer une AWS SAM application :

```
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
```

Pour obtenir la liste des images de conteneur de construction Amazon Elastic Container Registry (Amazon ECR) disponibles pour différentes exécutions, consultez Référentiels d'images pour AWS SAM.

Comment utiliser l'authentification OIDC avec les pipelines AWS SAM

AWS Serverless Application Model (AWS SAM) prend en charge l'authentification utilisateur OpenID Connect (OIDC) pour Bitbucket, GitHub Actions, ainsi que l'intégration GitLab continue et la livraison continue (comptes CI/CD) platforms. With this support, you can use authorized CI/CD utilisateurs) de l'une de ces plateformes pour gérer vos pipelines d'applications sans serveur. Dans le cas contraire, vous devrez créer et gérer plusieurs utilisateurs Gestion des identités et des accès AWS (IAM) pour contrôler l'accès aux AWS SAM pipelines.

Configuration de l'OIDC avec un pipeline AWS SAM

Au cours du processus de `sam pipeline bootstrap` configuration, procédez comme suit pour configurer OIDC avec votre AWS SAM pipeline.

1. Lorsque vous êtes invité à choisir un fournisseur d'identité, sélectionnez OIDC.
2. Ensuite, sélectionnez un fournisseur OIDC pris en charge.
3. Saisissez l'URL du fournisseur OIDC, en commençant par **`https://`**.

   > **Note**
   >
   > AWS SAM fait référence à cette URL lorsqu'elle génère le type de `AWS::IAM::OIDCProvider` ressource.

4. Suivez ensuite les instructions et entrez les informations de CI/CD plate-forme nécessaires pour accéder à la plate-forme sélectionnée. Ces informations varient selon la plateforme et peuvent inclure :
   - un ID client OIDC ;
   - le nom du référentiel ou l'identifiant universel unique (UUID) du code ;

- le nom de groupe ou d'organisation associé au référentiel ;
- GitHub organisation à laquelle appartient le référentiel de code.
- GitHub nom du référentiel.
- la branche à partir de laquelle les déploiements seront effectués.

5. AWS SAM affiche un résumé de la configuration OIDC saisie. Saisissez le numéro d'un paramètre pour le modifier, ou appuyez sur Enter pour continuer.
6. Lorsque vous êtes invité à confirmer la création des ressources nécessaires pour prendre en charge la connexion OIDC saisie, appuyez sur Y pour continuer.

AWS SAM génère une `AWS::IAM::OIDCProvider` AWS CloudFormation ressource avec la configuration fournie qui assume le rôle d'exécution du pipeline. Pour en savoir plus sur ce type de CloudFormation ressource, consultez AWS : :IAM : : OIDCProvider dans le guide de l'AWS CloudFormation utilisateur.

> **Note**
>
> Si la ressource du fournisseur d'identité (IdP) existe déjà dans votre répertoire Compte AWS, AWS SAM référencez-la au lieu de créer une nouvelle ressource.

Exemple

Voici un exemple de configuration d'OIDC avec un AWS SAM pipeline.

```
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]:
```

En savoir plus

Pour plus d'informations sur l'utilisation de l'OIDC avec un AWS SAM pipeline, consultez sam pipeline bootstrap.

Présentation de l'utilisation sam sync de la synchronisation avec AWS Cloud

La AWS Serverless Application Model commande Command Line Interface (AWS SAMCLI) `sam sync` fournit des options permettant de synchroniser rapidement les modifications apportées aux applications locales avec AWS Cloud. Utiliser `sam sync` lors du développement de vos applications pour :

1. Détectez et synchronisez automatiquement les modifications locales apportées au AWS Cloud.
2. Personnaliser les modifications locales qui sont synchronisées avec le AWS Cloud.
3. Préparer votre application dans le cloud pour les tests et la validation.

Avec `sam sync`, vous pouvez créer un flux de travail de développement rapide qui réduit le temps nécessaire à la synchronisation de vos modifications locales avec le cloud pour les tester et les valider.

> **Note**
>
> La commande `sam sync` est recommandée pour les environnements de développement. Pour les environnements de production, nous vous recommandons d'utiliser `sam deploy` ou de configurer un pipeline d'intégration et de livraison continues (CI/CD). Pour en savoir plus, veuillez consulter la section Déployez votre application et vos ressources avec AWS SAM.

La `sam sync` commande fait partie de AWS SAM Accelerate. AWS SAM Acceleratefournit des outils que vous pouvez utiliser pour accélérer l'expérience de développement et de test d'applications sans serveur dans le AWS Cloud.

Rubriques

- Détectez et synchronisez automatiquement les modifications locales apportées au AWS Cloud
- Personnalisez les modifications locales qui sont synchronisées avec AWS Cloud
- Préparer votre application dans le cloud pour les tests et la validation
- Options pour la commande sam sync
- Résolution des problèmes
- Exemples

- En savoir plus

Détectez et synchronisez automatiquement les modifications locales apportées au AWS Cloud

Exécutez `sam sync` avec l'option `--watch` pour commencer à synchroniser votre application avec le AWS Cloud. Cela effectue les opérations suivantes :

1. Créer votre application : ce processus est similaire à l'utilisation de la commande `sam build`.
2. Déployer votre application : la CLI AWS SAM déploie votre application sur AWS CloudFormation en utilisant vos paramètres par défaut. Les valeurs par défaut suivantes sont utilisées :
   a. AWS les informations d'identification et les paramètres de configuration généraux se trouvent dans votre dossier `.aws` utilisateur.
   b. Les paramètres de déploiement de l'application se trouvent dans le fichier `samconfig.toml` de votre application.

   Si les valeurs par défaut sont introuvables, la CLI AWS SAM vous en informera et quittera le processus de synchronisation.
3. Surveiller les modifications locales : la CLI AWS SAM continue de fonctionner et surveille les modifications locales apportées à votre application. C'est ce que permet l'option `--watch`.

   Cette option peut être activée par défaut. Pour les valeurs par défaut, consultez le fichier `samconfig.toml` de votre application. Voici un exemple de fichier de :

   ```
   ...
   [default.sync]
   [default.sync.parameters]
   watch = true
   ...
   ```
4. Synchroniser les modifications locales avec le AWS Cloud — Lorsque vous apportez des modifications locales, les AWS SAMCLI détecte et les synchronise AWS Cloud par le biais de la méthode la plus rapide disponible. Selon le type de modification, les éléments suivants peuvent se produire :
   a. Si votre ressource mise à jour prend en charge le AWS service APIs, elle l' AWS SAMCLIutilisera pour déployer vos modifications. Il en résulte une synchronisation rapide pour mettre à jour votre ressource dans le AWS Cloud.

b. Si votre ressource mise à jour ne prend pas en charge le AWS service APIs, elle AWS SAMCLI effectuera un CloudFormation déploiement. Cela met à jour l'intégralité de votre application dans le AWS Cloud. Bien que cela ne soit pas aussi rapide, cela vous évite d'avoir à lancer manuellement un déploiement.

Étant donné que la `sam sync` commande met automatiquement à jour votre application dans le AWS Cloud, elle est recommandée uniquement pour les environnements de développement. Lorsque vous exécutez `sam sync`, il vous sera demandé de confirmer :

```
**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
```

Personnalisez les modifications locales qui sont synchronisées avec AWS Cloud

Fournissez des options pour personnaliser les modifications locales synchronisées avec le AWS Cloud. Cela peut accélérer le temps nécessaire pour que vos modifications locales soient visibles dans le cloud en vue d'être testées et validées.

Par exemple, offrez la `--code` possibilité de synchroniser uniquement les modifications de code, telles que le code de AWS Lambda fonction. Pendant le développement, si vous vous concentrez spécifiquement sur le code Lambda, vos modifications seront rapidement transférées dans le cloud pour les tester et les valider. Voici un exemple :

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

Pour synchroniser uniquement les modifications de code pour une fonction ou une couche Lambda spécifique, utilisez l'option `--resource-id`. Voici un exemple :

```
$ sam sync --code --resource-id HelloWorldFunction --resource-id HelloWorldLayer
```

Préparer votre application dans le cloud pour les tests et la validation

La commande `sam sync` trouve automatiquement la méthode la plus rapide disponible pour mettre à jour votre application dans le AWS Cloud. Cela peut accélérer vos flux de travail de développement et de test dans le cloud. En utilisant le AWS service APIs, vous pouvez développer, synchroniser et tester rapidement les ressources prises en charge. Pour un exemple pratique, voir le module 6 - AWS SAM Accélérer dans The Complete AWS SAM Workshop.

Options pour la commande sam sync

Voici quelques-unes des principales options que vous pouvez utiliser pour modifier la commande `sam sync`. Pour obtenir la liste de toutes les options, consultez sam sync.

Effectuez un CloudFormation déploiement ponctuel

Utilisez l'option `--no-watch` pour désactiver la synchronisation automatique. Voici un exemple :

```
$ sam sync --no-watch
```

Ils AWS SAMCLI effectueront un CloudFormation déploiement unique. Cette commande regroupe les actions effectuées par les commandes `sam build` et `sam deploy`.

Ignorer le CloudFormation déploiement initial

Vous pouvez déterminer si un CloudFormation déploiement est requis à `sam sync` chaque exécution.

- Indiquez `--no-skip-deploy-sync` d'exiger un CloudFormation déploiement à `sam sync` chaque exécution. Cela garantit que votre infrastructure locale est synchronisée avec CloudFormation, évitant ainsi toute dérive. L'utilisation de cette option ajoute du temps à votre flux de travail de développement et de test.
- Indiquez `--skip-deploy-sync` pour rendre CloudFormation le déploiement facultatif. Il AWS SAMCLI comparera votre AWS SAM modèle local avec votre CloudFormation modèle déployé et ignorera le CloudFormation déploiement initial si aucune modification n'est détectée. Le fait de sauter CloudFormation le déploiement peut vous faire gagner du temps lors de la synchronisation des modifications locales apportées au. AWS Cloud

  Si aucune modification n'est détectée, le CloudFormation déploiement AWS SAMCLI sera tout de même effectué dans les scénarios suivants :

- Si 7 jours ou plus se sont écoulés depuis votre dernier CloudFormation déploiement.
- Si un grand nombre de modifications du code de fonction Lambda sont détectées, le CloudFormation déploiement est la méthode la plus rapide pour mettre à jour votre application.

Voici un exemple :

```
$ sam sync --skip-deploy-sync
```

Synchroniser une ressource à partir d'une pile imbriquée

Pour synchroniser une ressource à partir d'une pile imbriquée

1. Fournissez la pile racine en utilisant `--stack-name`.
2. Identifiez la ressource dans la pile imbriquée en utilisant le format suivant : *`nestedStackId/resourceId`*.
3. Fournissez la ressource dans la pile imbriquée en utilisant `--resource-id`.

   Voici un exemple :

   ```
   $ sam sync --code --stack-name sam-app --resource-id myNestedStack/
   HelloWorldFunction
   ```

Pour plus d'informations sur la création des applications imbriquées, consultez Réutilisez le code et les ressources à l'aide d'applications imbriquées dans AWS SAM.

Spécifiez une CloudFormation pile spécifique à mettre à jour

Pour spécifier une CloudFormation pile spécifique à mettre à jour, indiquez l'`--stack-name`option. Voici un exemple :

```
$ sam sync --stack-name dev-sam-app
```

Accélérez les temps de création en créant votre projet dans le dossier source

Pour les systèmes d'exécution et les méthodes de création pris en charge, vous pouvez utiliser l'option `--build-in-source` permettant de créer votre projet directement dans le dossier source.

Par défaut, il est AWS SAM CLI compilé dans un répertoire temporaire, ce qui implique de copier le code source et les fichiers de projet. Avec`--build-in-source`, les AWS SAM CLI builds se trouvent directement dans votre dossier source, ce qui accélère le processus de compilation en supprimant le besoin de copier des fichiers dans un répertoire temporaire.

Pour obtenir une liste des systèmes d'exécution ainsi que des méthodes de création pris en charge, consultez `--build-in-source`.

Spécifiez les fichiers et les dossiers qui ne lanceront pas une synchronisation

Utilisez l'option `--watch-exclude` pour spécifier tout fichier ou dossier qui ne lancera pas une synchronisation lors de la mise à jour. Pour plus d'informations sur cette option, consultez `--watch-exclude`.

Voici un exemple qui exclut le fichier `package-lock.json` associé à notre fonction `HelloWorldFunction` :

```
$ sam sync --watch --watch-exclude HelloWorldFunction=package-lock.json
```

Lorsque cette commande est exécutée, le processus de synchronisation AWS SAM CLI sera lancé. Cela inclut les éléments suivants :

- exécutez `sam build` pour créer vos fonctions et préparer votre application pour le déploiement.
- exécutez `sam deploy` pour déployer votre application.
- surveillez les changements apportées à votre application.

Lorsque nous modifierons le `package-lock.json` fichier, aucune synchronisation AWS SAM CLI ne sera lancée. Lorsqu'un autre fichier est mis à jour, une synchronisation AWS SAM CLI sera lancée, qui inclura le `package-lock.json` fichier.

Voici un exemple de spécification de fonction Lambda d'une pile enfant :

```
$ sam sync --watch --watch-exclude ChildStackA/MyFunction=database.sqlite3
```

Résolution des problèmes

Pour résoudre le problème AWS SAMCLI, voirRésolution des problèmes de la CLI AWS SAM.

Exemples

Utiliser sam sync pour mettre à jour l'application Hello World

Dans cet exemple, nous commençons par initialiser l'exemple d'application Hello World. Pour en savoir plus sur cette application, consultez Tutoriel : Déployer une application Hello World avec AWS SAM.

L'exécution de la commande `sam sync` lance le processus de création et de déploiement.

```
$ 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.
```

Une fois le déploiement terminé, nous modifions le code `HelloWorldFunction`. AWS SAMCLIDétecte ce changement et synchronise notre application avec le AWS Cloud. Étant donné que AWS le service est AWS Lambda compatible APIs, une synchronisation rapide est effectuée.

```
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.
```

Ensuite, nous modifions notre point de terminaison d'API dans le AWS SAM modèle de l'application. Nous modifions `/hello` pour `/helloworld`.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  HelloWorldFunction:
    ...
    Properties:
      ...
      Events:
        HelloWorld:
          Type: Api
          Properties:
            Path: /helloworld
            Method: get
```

Étant donné que la ressource Amazon API Gateway ne prend pas en charge l'API du AWS service, elle effectue AWS SAMCLI automatiquement un CloudFormation déploiement. Voici un exemple de résultat :

```
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.
```

En savoir plus

Pour obtenir une description de toutes les options `sam sync`, consulter sam sync.

Surveillez votre application sans serveur avec AWS SAM

Après avoir déployé votre application sans serveur, vous pouvez la surveiller pour obtenir des informations sur son fonctionnement et détecter les anomalies, ce qui peut vous aider à résoudre les problèmes. Cette section fournit des informations détaillées sur la surveillance de votre application sans serveur. Cela inclut des informations sur la façon de configurer Amazon CloudWatch pour qu'il vous avertisse lorsqu'il détecte des anomalies. Il fournit également des informations sur l'utilisation des journaux, notamment la mise en évidence des erreurs et des conseils pour l'affichage, le filtrage, la récupération et le suivi des journaux.

Rubriques

- Utilisation d' CloudWatch Application Insights pour surveiller vos applications AWS SAM sans serveur
- Utilisation des identifiants AWS SAM

Utilisation d' CloudWatch Application Insights pour surveiller vos applications AWS SAM sans serveur

Amazon CloudWatch Application Insights vous aide à surveiller les AWS ressources de vos applications afin d'identifier les problèmes potentiels. Il peut analyser les données relatives aux AWS ressources pour détecter les signes de problèmes et créer des tableaux de bord automatisés pour les visualiser. Vous pouvez configurer CloudWatch Application Insights pour l'utiliser avec vos AWS Serverless Application Model (AWS SAM) applications. Pour en savoir plus sur CloudWatch Application Insights, consultez Amazon CloudWatch Application Insights dans le guide de CloudWatch l'utilisateur Amazon.

Rubriques

- Configuration d' CloudWatch Application Insights avec AWS SAM
- Étapes suivantes

Configuration d' CloudWatch Application Insights avec AWS SAM

Configurez CloudWatch Application Insights pour vos AWS SAM applications via l'interface de ligne de AWS SAM commande (AWS SAMCLI) ou via vos AWS SAM modèles.

Configurer via la CLI AWS SAM

Lorsque vous initialisez votre application avecsam init, activez CloudWatch Application Insights via le flux interactif ou en utilisant l'--application-insightsoption.

Pour activer CloudWatch Application Insights via le flux AWS SAMCLI interactif, entrez **y** lorsque vous y êtes invité.

```
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]:
```

Pour activer CloudWatch Application Insights avec --application-insights cette option, procédez comme suit.

```
sam init --application-insights
```

Pour en savoir plus sur l'utilisation de la commande sam init, consultez sam init.

Configuration à l'aide AWS SAM de modèles

Activez CloudWatch Application Insights en définissant les `AWS::ApplicationInsights::Application` ressources `AWS::ResourceGroups::Group` et les ressources dans vos AWS SAM modèles.

```
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`— Crée un groupe pour organiser vos AWS ressources afin de gérer et d'automatiser des tâches sur un grand nombre de ressources à la fois. Ici, vous créez un groupe de ressources à utiliser avec CloudWatch Application Insights. Pour plus d'informations sur ce type de ressource, veuillez consulter `AWS::ResourceGroups::Group` dans le Guide de l'utilisateur AWS CloudFormation .
- `AWS::ApplicationInsights::Application`— Configure CloudWatch Application Insights pour le groupe de ressources. Pour plus d'informations sur ce type de ressource, veuillez consulter `AWS::ApplicationInsights::Application` dans le Guide de l'utilisateur AWS CloudFormation .

Les deux ressources sont automatiquement transmises au CloudFormation moment du déploiement de l'application. Vous pouvez utiliser la CloudFormation syntaxe de votre AWS SAM modèle pour configurer davantage CloudWatch Application Insights. Pour plus d'informations, consultez la section Utiliser CloudFormation des modèles dans le guide de CloudWatch l'utilisateur Amazon.

Lorsque vous utilisez la sam init --application-insights commande, ces deux ressources sont automatiquement générées dans votre AWS SAM modèle. Voici un exemple de modèle généré.

```
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
```

Étapes suivantes

Après avoir configuré CloudWatch Application Insights, utilisez-le sam build pour sam deploy créer et déployer votre application. Toutes les ressources prises en charge par CloudWatch Application Insights seront configurées pour la surveillance.

- Pour obtenir la liste des ressources prises en charge, consultez la section Journaux et métriques pris en charge dans le guide de CloudWatch l'utilisateur Amazon.
- Pour savoir comment accéder à CloudWatch Application Insights, consultez Access CloudWatch Application Insights dans le guide de CloudWatch l'utilisateur Amazon.

Utilisation des identifiants AWS SAM

Pour simplifier la résolution des problèmes, la CLI AWS SAM possède une commande appelée `sam logs`. Cette commande permet d'extraire les journaux générés par votre fonction Lambda depuis la ligne de commande.

> **Note**
>
> La `sam logs` commande fonctionne pour toutes les AWS Lambda fonctions, pas uniquement celles que vous déployez à l'aide AWS SAM.

Récupération des journaux par pile CloudFormation

Lorsque votre fonction fait partie d'une CloudFormation pile, vous pouvez récupérer les journaux en utilisant l'identifiant logique de la fonction :

```
sam logs -n HelloWorldFunction --stack-name mystack
```

Extraction des journaux par nom de fonction Lambda

Vous pouvez également extraire les journaux en utilisant le nom de la fonction :

```
sam logs -n mystack-HelloWorldFunction-1FJ8PD
```

Journaux détaillés

Ajoutez l'option `--tail` pour attendre les nouveaux journaux et les consulter au fur et à mesure qu'ils arrivent. Cela est utile pendant un déploiement ou lors de la résolution d'un problème de production.

```
sam logs -n HelloWorldFunction --stack-name mystack --tail
```

Affichage des journaux pour une plage de temps spécifique

Vous pouvez afficher les journaux pour une plage de temps spécifique à l'aide des options `-s` et `-e` :

```
sam logs -n HelloWorldFunction --stack-name mystack -s '10min ago' -e '2min ago'
```

Filtrage des journaux

Utilisez l'option `--filter` pour rechercher rapidement des journaux correspondant à des termes, expressions ou valeurs dans vos journaux d'événements.

```
sam logs -n HelloWorldFunction --stack-name mystack --filter "error"
```

Dans la réponse générée, la CLI AWS SAM soulignera toutes les occurrences du terme « error » (erreur), afin que vous puissiez facilement trouver le mot clé filtre dans la réponse générée par le journal.

Mise en surbrillance des erreurs

Lorsque votre fonction Lambda se bloque ou expire, la CLI AWS SAM met en surbrillance le message de délai d'attente en rouge. Cela permet de localiser aisément les exécutions spécifiques qui sont sur le point d'expirer au sein d'une diffusion importante de sortie de journal.

Impression JSON « pretty printing »

Si vos messages de journal impriment des chaînes JSON, la CLI AWS SAM imprimera automatiquement le fichier JSON en « Pretty-print » pour vous aider à le comprendre et à l'analyser visuellement.

AWS SAM référence

Cette section contient des documents AWS SAM de référence. Cela inclut des documents de AWS SAMCLI référence, tels que des informations de référence sur AWS SAMCLI les commandes et des AWS SAMCLI informations supplémentaires, telles que des informations de configuration, de contrôle de version et de dépannage. En outre, cette section inclut des informations de référence sur la AWS SAM spécification et le AWS SAM modèle, telles que des informations de référence sur les connecteurs, les référentiels d'images et les déploiements.

AWS SAM spécification et AWS SAM modèle

La AWS SAM spécification est une spécification open source sous licence Apache 2.0. La version actuelle de la AWS SAM spécification est disponible dans leAWS SAM modèle. AWS SAM La spécification est fournie avec une syntaxe abrégée simplifiée que vous utilisez pour définir les fonctions, les événements APIs, les configurations et les autorisations de votre application sans serveur.

Vous interagissez avec les AWS SAM spécifications par le biais du répertoire du projet d' AWS SAM application, qui comprend les dossiers et les fichiers créés lorsque vous exécutez la sam init commande. Ce répertoire inclut le AWS SAM modèle, un fichier important qui définit vos AWS ressources. Le AWS SAM modèle est une extension d'un AWS CloudFormation modèle. Pour la référence complète pour les modèles CloudFormation , veuillez consulter la rubrique Référence de modèles dans le Guide de l'utilisateur AWS CloudFormation .

Référence des commandes CLI AWS SAM

L'interface de ligne de AWS Serverless Application Model commande (AWS SAMCLI) est un outil de ligne de commande que vous pouvez utiliser avec des AWS SAM modèles et des intégrations tierces prises en charge pour créer et exécuter vos applications sans serveur.

Vous pouvez utiliser les commandes de la CLI AWS SAM pour développer, tester et déployer vos applications sans serveur dans le AWS Cloud. Voici quelques exemples de commandes de la CLI AWS SAM :

- `sam init` – Si vous utilisez la CLI AWS SAM pour la première fois, vous pouvez exécuter la commande `sam init` sans aucun paramètre pour créer une application Hello World. La

commande génère un AWS SAM modèle préconfiguré et un exemple de code d'application dans la langue de votre choix.

- `sam local invoke` et `sam local start-api` : utilisez ces commandes pour tester le code de votre application localement, avant de le déployer sur l' AWS Cloud.
- `sam logs` : utilisez cette commande pour récupérer les journaux générés par votre fonction Lambda. Cela peut vous aider à tester et à déboguer votre application après l'avoir déployée sur l' AWS Cloud.
- `sam package` : utilisez cette commande pour regrouper votre code d'application et vos dépendances dans un package de déploiement. Le package de déploiement est nécessaire pour charger votre application dans l' AWS Cloud.
- `sam deploy` : utilisez cette commande pour déployer votre application sans serveur sur l' AWS Cloud. Il crée les AWS ressources et définit les autorisations et les autres configurations définies dans le AWS SAM modèle.

Pour obtenir des instructions sur l'installation du AWS SAMCLI, voirInstaller la CLI AWS SAM.

Modèles de politique AWS SAM

Avec AWS SAM, vous pouvez choisir parmi une liste de modèles de politique pour définir les autorisations de votre AWS Lambda fonction sur les ressources utilisées par votre application. Pour obtenir la liste des modèles de politiques disponibles, reportez-vous àTableau de modèle de stratégie. Pour obtenir des informations générales sur les modèles de politiques AWS SAM, reportez-vous àModèles de politique AWS SAM.

Rubriques

- AWS SAM modèle
- Référence des commandes CLI AWS SAM
- Fichier de configuration CLI AWS SAM
- AWS SAM référence du connecteur
- Modèles de politique AWS SAM
- Référentiels d'images pour AWS SAM
- Télémétrie dans la CLI AWS SAM

- Configuration et gestion de l'accès aux ressources dans votre AWS SAM modèle

Référence des commandes CLI AWS SAM

Cette section inclut des informations de référence sur AWS SAMCLI les commandes. Cela inclut des détails sur l'utilisation, une liste complète des différentes options disponibles pour chaque commande et des informations supplémentaires. Le cas échéant, les informations supplémentaires incluent des détails tels que les arguments, les variables d'environnement et les événements. Consultez chaque commande pour plus de détails. Pour obtenir des instructions sur l'installation du AWS SAMCLI, voirInstaller la CLI AWS SAM.

Rubriques

- sam build
- sam delete
- sam deploy
- sam init
- sam list
- sam local callback
- sam local execution
- sam local generate-event
- sam local invoke
- sam local 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

sam build

Cette page fournit des informations de référence pour la AWS Serverless Application Model commande Command Line Interface (AWS SAMCLI)`sam build`.

- Pour une introduction à la AWS SAMCLI, voir Qu'est-ce que c'est AWS SAMCLI ?
- Pour obtenir de la documentation sur l'utilisation de la AWS SAMCLI `sam build` commande, consultezInitiation à la construction avec AWS SAM.

La commande `sam build` prépare une application pour les étapes suivantes du flux de travail du développeur, telles que le test local ou le déploiement sur AWS Cloud.

Usage

```
$ sam build <arguments> <options>
```

Arguments

ID de ressource

Facultatif. Indique AWS SAM de créer une seule ressource déclarée dans un AWS SAM modèle. Les artefacts de création pour la ressource spécifiée sont les seuls disponibles pour les commandes suivantes du flux de travail ; c'est-à-dire `sam package` et `sam deploy`.

Options

`--base-dir, -s DIRECTORY`

Résout les chemins relatifs au code source de la fonction ou de la couche par rapport à ce répertoire. Utilisez cette option si vous souhaitez modifier la façon dont les chemins d'accès relatifs aux dossiers de code source sont résolus. Par défaut, les chemins relatifs sont résolus selon l'emplacement du modèle AWS SAM .

Outre les ressources de l'application racine ou de la pile que vous créez, cette option applique également des applications ou des piles imbriquées.

Cette option s'applique aux types de ressources et propriétés suivants :

- Type de ressource : propriété `AWS::Serverless::Function` : `CodeUri`
- Type de ressource : attribut de ressource `AWS::Serverless::Function` : entrée `Metadata` : `DockerContext`
- Type de ressource : propriété `AWS::Serverless::LayerVersion` : `ContentUri`
- Type de ressource : propriété `AWS::Lambda::Function` : `Code`
- Type de ressource : propriété `AWS::Lambda::LayerVersion` : `Content`

`--beta-features | --no-beta-features`

Autorisez ou refusez les fonctionnalités bêta.

`--build-dir, -b` *`DIRECTORY`*

Le chemin d'accès à un répertoire dans lequel les artefacts créés sont stockés. Ce répertoire et tout son contenu sont supprimés avec cette option.

`--build-image` *`TEXT`*

L'URI de l'image de conteneur que vous souhaitez extraire pour la création. Par défaut, AWS SAM extrait l'image de conteneur depuis Amazon ECR Public. Utilisez cette option pour extraire l'image à partir d'un autre emplacement.

Vous pouvez spécifier cette option plusieurs fois. Chaque instance de cette option peut prendre une chaîne ou une paire valeur clé. Si vous spécifiez une chaîne, il s'agit de l'URI de l'image de conteneur à utiliser pour toutes les ressources de l'application. Par exemple, `sam build --use-container --build-image amazon/aws-sam-cli-build-image-python3.8`. Si vous spécifiez une paire valeur clé, la clé est le nom de la ressource et la valeur est l'URI de l'image de conteneur à utiliser pour cette ressource. Par exemple `sam build --use-container --build-image Function1=amazon/aws-sam-cli-build-image-python3.8`. Avec les paires valeur clé, vous pouvez spécifier différentes images de conteneur pour différentes ressources.

Cette option ne s'applique que si l'option `--use-container` est spécifiée, dans le cas contraire, une erreur se produit.

`--build-in-source | --no-build-in-source`

Procurez-vous `--build-in-source` pour créer votre projet directement dans le dossier source.

L'option `--build-in-source` prend en charge les systèmes d'exécution ainsi que les méthodes de création suivantes :

- exécutions : toute méthode d'exécution Node.js prise en charge par l'option `sam init --runtime`.
- Méthodes de construction : `Makefile`, `esbuild`.

L'option `--build-in-source` n'est pas compatible avec celles suivantes :

- `--hook-name`
- `--use-container`

Par défaut : `--no-build-in-source`

`--cached | --no-cached`

Activez ou désactivez les créations mises en cache. Utilisez cette option pour réutiliser les artefacts de build qui n'ont pas changé par rapport aux versions précédentes. AWS SAM évalue si vous avez modifié des fichiers dans le répertoire de votre projet. Par défaut, les créations ne sont pas mises en cache. Si l'`--no-cached`option est invoquée, elle remplace le `cached = true` paramètre défini dans samconfig.toml.

> **Note**
>
> AWS SAM n'évalue pas si vous avez modifié des modules tiers dont le projet dépend, si vous n'avez pas fourni de version spécifique. Par exemple, si votre fonction Python inclut un `requirements.txt` fichier contenant l'entrée `requests=1.x` et que la dernière version du module de requête passe de `1.1` à`1.2`, AWS SAM elle n'extrait pas la dernière version tant que vous n'avez pas exécuté une version non mise en cache.

`--cache-dir`

Le répertoire dans lequel les artefacts mis en cache sont stockés lorsque `--cached` est spécifié. Le répertoire mis en caché par défaut est `.aws-sam/cache`.

`--config-env` *`TEXT`*

Le nom d'environnement spécifiant les valeurs de paramètre par défaut dans le fichier de configuration à utiliser. La valeur par défaut est « par défaut ». Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--config-file` *PATH*

Le chemin d'accès et le nom du fichier de configuration contenant les valeurs de paramètres par défaut à utiliser. La valeur par défaut est « `samconfig.toml` » dans la racine du répertoire du projet. Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--container-env-var, -e` *TEXT*

Variables d'environnement à transmettre au conteneur de création. Vous pouvez spécifier cette option plusieurs fois. Chaque instance de cette option prend une paire clé-valeur, où la clé est la variable de ressource et d'environnement et la valeur est la valeur de la variable d'environnement. Par exemple : `--container-env-var Function1.GITHUB_TOKEN=TOKEN1 --container-env-var Function2.GITHUB_TOKEN=TOKEN2`.

Cette option ne s'applique que si l'option `--use-container` est spécifiée, dans le cas contraire, une erreur se produit.

`--container-env-var-file, -ef` *PATH*

Le chemin d'accès et le nom d'un fichier JSON contenant les valeurs des variables d'environnement de conteneur. Pour plus d'informations sur les fichier des variables d'environnement de conteneur, consultez Fichier de variables d'environnement du conteneur..

Cette option ne s'applique que si l'option `--use-container` est spécifiée, dans le cas contraire, une erreur se produit.

`--debug`

Active la journalisation de débogage pour imprimer les messages de débogage générés par la CLI AWS SAM et pour afficher les horodatages.

`--docker-network` *TEXT*

Spécifie le nom ou l'identifiant d'un réseau Docker existant auquel les conteneurs Docker Lambda doivent se connecter, avec le réseau de pont par défaut. Si cela n'est pas spécifié, les conteneurs Lambda se connectent uniquement au réseau Docker de pont par défaut.

`--exclude, -x`

Le nom de la ou des ressource(s) à exclure de `sam build`. Par exemple, si votre modèle contient `Function1`, `Function2` et `Function3` et vous exécutez `sam build --exclude Function2`, uniquement `Function1` et `Function3` seront créés.

`--help`

Affiche ce message, puis se ferme.

`--hook-name` *TEXT*

Nom du hook utilisé pour étendre la fonctionnalité de la CLI AWS SAM.

Valeurs acceptées : `terraform`.

`--manifest , -m` *PATH*

Le chemin d'accès à un fichier manifeste de dépendance personnalisé (par exemple, package.json) à utiliser à la place du fichier par défaut.

`--mount-symlinks`

Garantit le AWS SAMCLI montage permanent des liens symboliques présents dans les fichiers à créer ou à invoquer. Cela s'applique uniquement aux liens symboliques du répertoire de niveau supérieur (c'est-à-dire aux liens symboliques situés directement à la racine de la fonction). Par défaut, les liens symboliques ne sont pas montés, à l'exception de ceux nécessaires pour être utilisés `node_modules` dans `build-in-source` NodeJS.

`--no-use-container`

Option qui vous permet d'utiliser le kit d'outils IDE pour définir le comportement par défaut. Vous pouvez également l'utiliser `sam build --no-use-container` pour exécuter un build sur votre machine locale au lieu d'un conteneur docker.

`--parallel`

Active les builds parallèles. Utilisez cette option pour créer les fonctions et les couches de votre AWS SAM modèle en parallèle. Par défaut, les fonctions et les couches sont créées en séquence.

`--parameter-overrides`

(Facultatif) Chaîne contenant des remplacements de CloudFormation paramètres codés sous forme de paires clé-valeur. Utilise le même format que le AWS Command Line Interface (AWS CLI). Par exemple : « `ParameterKey=KeyPairName`, `ParameterValue=MyKey` `ParameterKey=InstanceType`, `ParameterValue=t1.micro` ». Cette option n'est pas compatible avec `--hook-name`.

`--profile` *TEXT*

Le profil spécifique de votre fichier d'informations d'identification qui obtient les AWS informations d'identification.

`--region TEXT`

Le Région AWS vers lequel déployer. Par exemple, us-east-1.

`--save-params`

Enregistrez les paramètres que vous fournissez sur la ligne de commande dans le fichier AWS SAM de configuration.

`--skip-prepare-infra`

Ignore l'étape de préparation si aucune modification d'infrastructure n'a été apportée. À utiliser avec l'option `--hook-name`.

`--skip-pull-image`

Spécifie si la commande doit ignorer la dernière image Docker pour l'exécution Lambda.

`--template-file, --template, -t PATH`

Le chemin et le nom du fichier AWS SAM modèle`[default: template.[yaml|yml]]`. Cette option n'est pas compatible avec `--hook-name`.

`--terraform-project-root-path`

Le chemin relatif ou absolu vers le répertoire de premier niveau contenant vos fichiers de configuration Terraform ou le code source de vos fonctions. Si ces fichiers se trouvent en dehors du répertoire contenant votre module racine Terraform, utilisez cette option pour spécifier son chemin absolu ou relatif. Cette option nécessite que `--hook-name` soit défini sur `terraform`.

`--use-container`, `-u`

Si les fonctions dépendent de packages avec des dépendances compilées nativement, utilisez cette option pour créer la fonction dans un conteneur Docker de type Lambda.

Exemple

Pour un exemple détaillé et une présentation détaillée de l'utilisation de la `sam build` sous-commande, reportez-vous à. Initiation à la construction avec AWS SAM

sam delete

Cette page fournit des informations de référence pour la AWS Serverless Application Model commande Command Line Interface (AWS SAMCLI)`sam delete`.

Pour une introduction à la AWS SAMCLI, voir Qu'est-ce que c'est AWS SAMCLI ?

La `sam delete` commande supprime une AWS SAM application en supprimant la CloudFormation pile, les artefacts empaquetés et déployés sur Amazon S3 et Amazon ECR, ainsi que le AWS SAM fichier modèle.

Cette commande vérifie également si une pile complémentaire Amazon ECR est déployée et, dans l'affirmative, invite l'utilisateur à supprimer cette pile et les référentiels Amazon ECR. Si `--no-prompts` est spécifié, alors les piles compagnons et les répertoires Amazon ECR sont supprimés par défaut.

Usage

```
$ sam delete <options>
```

Options

`--config-env` *TEXT*

Le nom d'environnement spécifiant les valeurs de paramètre par défaut dans le fichier de configuration à utiliser. La valeur par défaut est `default`. Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--config-file` *PATH*

Le chemin d'accès et le nom du fichier de configuration contenant les valeurs de paramètres par défaut à utiliser. La valeur par défaut est `samconfig.toml` à la racine du répertoire du projet. Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--debug`

Active la journalisation de débogage pour imprimer le message de débogage généré par la CLI AWS SAM et pour afficher les horodatages.

`--help`

Affiche ce message, puis se ferme.

`--no-prompts`

Spécifiez cette option pour AWS SAM fonctionner en mode non interactif. Le nom de la pile doit être fourni, soit avec l'option `--stack-name`, soit dans le fichier de configuration `toml`.

`--profile` *TEXT*

Le profil spécifique de votre fichier d'informations d'identification qui obtient les AWS informations d'identification.

`--region` *TEXT*

La AWS région dans laquelle le déploiement doit être effectué. Par exemple, us-east-1.

`--s3-bucket`

Le chemin du compartiment Amazon S3 que vous voulez supprimer.

`--s3-prefix`

Le préfixe du compartiment Amazon S3 que vous voulez supprimer.

`--save-params`

Enregistrez les paramètres que vous fournissez sur la ligne de commande dans le fichier AWS SAM de configuration.

`--stack-name` *TEXT*

Nom de la CloudFormation pile que vous souhaitez supprimer.

Exemples

La commande suivante supprime la pile`MY-STACK`.

```
$ sam delete --stack-name MY-STACK
```

La commande suivante supprime la pile `MY-STACK` et le compartiment `sam-s3-demo-bucket` S3 :

```
$ sam delete \
    --stack-name MyStack \
    --s3-bucket MySAMBucket
```

sam deploy

Cette page fournit des informations de référence pour la AWS Serverless Application Model commande Command Line Interface (AWS SAMCLI)`sam deploy`.

- Pour une introduction à la AWS SAMCLI, voir Qu'est-ce que c'est AWS SAMCLI ?

- Pour obtenir de la documentation sur l'utilisation de la AWS SAMCLI `sam deploy` commande, consultez[Présentation du déploiement avec AWS SAM](#).

La `sam deploy` commande déploie une application auprès de l'utilisateur AWS Cloud . AWS CloudFormation

Usage

```
$ <environment variables> sam deploy <options>
```

Variables d'environnement

`SAM_CLI_POLL_DELAY`

Définissez la variable d'`SAM_CLI_POLL_DELAY`environnement avec une valeur de secondes dans votre shell pour configurer la fréquence à laquelle la CLI AWS SAM vérifie l'état de la CloudFormation pile, ce qui est utile lorsque vous constatez une limitation provenant de. CloudFormation Cette variable d'environnement est utilisée pour interroger les appels `describe_stack` d'API, qui sont effectués pendant l'exécution`sam deploy`.

Voici un exemple de cette variable :

```
$ SAM_CLI_POLL_DELAY=5 sam deploy
```

Options

`--capabilities` *`LIST`*

Liste des fonctionnalités que vous devez spécifier pour autoriser la création CloudFormation de certaines piles. Certains modèles de pile peuvent inclure des ressources qui affectent les autorisations de votre compte Compte AWS, par exemple en créant de nouveaux utilisateurs Gestion des identités et des accès AWS (IAM). Pour ces piles, vous devez explicitement reconnaître leurs capacités en spécifiant cette option. Les seules valeurs valides sont `CAPABILITY_IAM` et `CAPABILITY_NAMED_IAM`. Si vous disposez de ressources IAM, vous pouvez spécifier l'une ou l'autre de ces capacités. Si vous disposez de ressources IAM avec des noms personnalisés, vous devez spécifier `CAPABILITY_NAMED_IAM`. Si vous ne spécifiez pas cette option, l'opération renvoie une erreur `InsufficientCapabilities`.

Lorsque vous déployez une application contenant des applications imbriquées, vous devez utiliser `CAPABILITY_AUTO_EXPAND` pour confirmer que l'application contient des applications imbriquées. Pour de plus amples informations, veuillez consulter Déploiement d'applications imbriquées.

`--config-env` *TEXT*

Le nom d'environnement spécifiant les valeurs de paramètre par défaut dans le fichier de configuration à utiliser. La valeur par défaut est `default`. Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--config-file` *PATH*

Le chemin d'accès et le nom du fichier de configuration contenant les valeurs de paramètres par défaut à utiliser. La valeur par défaut est `samconfig.toml` à la racine du répertoire du projet. Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--confirm-changeset | --no-confirm-changeset`

Invite à confirmer si la CLI AWS SAM déploie le jeu de modifications calculé.

`--debug`

Activez la journalisation de débogage pour imprimer le message de débogage généré par la CLI AWS SAM et pour afficher les horodatages.

`--disable-rollback | --no-disable-rollback`

Spécifiez si vous souhaitez annuler votre CloudFormation pile en cas d'erreur lors d'un déploiement. Par défaut, en cas d'erreur lors d'un déploiement, votre CloudFormation pile revient à son dernier état stable. Si vous spécifiez `--disable-rollback` et une erreur se produit pendant un déploiement, les ressources créées ou mises à jour avant l'erreur ne sont pas restaurées.

`--fail-on-empty-changeset | --no-fail-on-empty-changeset`

Indiquez s'il faut renvoyer un code de sortie différent de zéro s'il n'existe aucune modification à apporter à la pile. Le comportement par défaut consiste à renvoyer un code de sortie autre que zéro.

`--force-upload`

Spécifiez cette option pour télécharger des artefacts, même s’ils correspondent à des artefacts existants dans le compartiment Amazon S3. Les artefacts correspondants sont remplacés.

`--guided, -g`

Spécifiez cette option pour que la CLI AWS SAM utilise des invites pour vous guider tout au long du déploiement.

`--help`

Affichez ce message et quittez.

`--image-repositories` *TEXT*

Un mappage des fonctions vers l'URI de leur référentiel Amazon ECR. Référencez les fonctions par leur identifiant logique. Voici un exemple :

```
$ sam deploy --image-repositories Function1=123456789012.dkr.ecr.us-
east-1.amazonaws.com/my-repo
```

Vous pouvez spécifier cette option plusieurs fois au sein d'une même commande.

`--image-repository` *TEXT*

Le nom du référentiel Amazon ECR vers lequel cette commande charge l'image de la fonction. Cette option est requise pour les fonctions déclarées avec le type de package `Image`.

`--kms-key-id` *TEXT*

L'ID d'une clé AWS Key Management Service (AWS KMS) utilisée pour chiffrer les artefacts inactifs dans le compartiment Amazon S3. Si vous ne spécifiez pas cette option, AWS SAM utilise les clés de chiffrement gérées par Amazon S3.

`--metadata`

Une carte de métadonnées à attacher à tous les artefacts référencés dans le modèle.

`--no-execute-changeset`

Indique s'il faut appliquer le jeu de modifications. Spécifiez cette option si vous voulez afficher les modifications de la pile avant d'appliquer le jeu de modifications. Cette commande crée un jeu de modifications CloudFormation puis se termine sans appliquer le jeu de modifications. Pour appliquer le jeu de modifications, exécutez la même commande sans cette option.

`--no-progressbar`

N’affichez pas de barre de progression lors du téléchargement d’artefacts sur Amazon S3.

`--notification-arns LIST`

Liste des rubriques Amazon Simple Notification Service (Amazon SNS) ARNs associées à la CloudFormation pile.

`--on-failure [ROLLBACK | DELETE | DO_NOTHING]`

Spécifiez l'action à entreprendre lorsqu'une pile ne peut pas être créée.

Les options suivantes sont disponibles :

- `ROLLBACK` – Récupère la pile à un état antérieur connu et valide.
- `DELETE` – Récupère la pile à un état antérieur connu et valide, s'il en existe un. Sinon, supprime la pile.
- `DO_NOTHING` – Ne récupère ni ne supprime la pile. L'effet est le même que celui de `--disable-rollback`.

Le comportement par défaut est `ROLLBACK`.

> **Note**
>
> Vous pouvez spécifier l'`--disable-rollback`option ou l'`--on-failure`option, mais pas les deux.

`--parameter-overrides LIST`

Chaîne contenant des remplacements de CloudFormation paramètres codés sous forme de paires clé-valeur. Chaque remplacement utilise le format. `ParameterKey=name,ParameterValue=value` Les remplacements multiples sont séparés par des espaces. Voici deux exemples :

```
$ 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`

Le profil spécifique de votre fichier d'informations d'identification qui obtient les AWS informations d'identification.

`--region` *TEXT*

Le Région AWS vers lequel déployer. Par exemple, us-east-1.

`--resolve-image-repos`

Créez automatiquement des référentiels Amazon ECR à utiliser pour l'empaquetage et le déploiement de déploiements non guidés. Cette option s'applique uniquement aux fonctions et aux couches avec un `PackageType: Image` spécifié. Si vous spécifiez l'option `--guided`, la CLI AWS SAM ignore `--resolve-image-repos`.

> **Note**
>
> Si vous créez AWS SAM automatiquement des référentiels Amazon ECR pour des fonctions ou des couches avec cette option, et que vous supprimez ultérieurement ces fonctions ou couches de votre AWS SAM modèle, les référentiels Amazon ECR correspondants sont automatiquement supprimés.

`--resolve-s3`

Créez automatiquement un compartiment Amazon S3 à utiliser pour l'empaquetage et le déploiement de déploiements non guidés. Si vous spécifiez l'option `--guided`, l'interface CLI AWS SAM ignore `--resolve-s3`. Si vous spécifiez les deux options `--s3-bucket` et `--resolve-s3`, une erreur se produit.

`--role-arn` *TEXT*

Nom de ressource Amazon (ARN) d'un rôle IAM CloudFormation assumé lors de l'application de l'ensemble de modifications.

`--s3-bucket` *TEXT*

Le nom du compartiment Amazon S3 dans lequel cette commande télécharge votre CloudFormation modèle. Si le modèle a une taille supérieure à 51 200 octets, l'option `--s3-bucket` ou l'option `--resolve-s3` est obligatoire. Si vous spécifiez les deux options `--s3-bucket` et `--resolve-s3`, une erreur se produit.

`--s3-prefix` *TEXT*

Le préfixe ajouté aux noms des artefacts téléchargés vers le compartiment Amazon S3. Le nom du préfixe est un nom de chemin d'accès (nom de dossier) pour le compartiment Amazon S3.

`--save-params`

Enregistrez les paramètres que vous fournissez sur la ligne de commande dans le fichier AWS SAM de configuration.

`--signing-profiles` *`LIST`*

Liste des profils de signature pour signer les packages de déploiement. Cette option prend une liste de paires valeur clé, où la clé est le nom de la fonction ou de la couche à signer et la valeur est le profil de signature, avec un propriétaire de profil optionnel délimité par `:`. Par exemple, `FunctionNameToSign=SigningProfileName1 LayerNameToSign=SigningProfileName2:SigningProfileOwner`.

`--stack-name` *`TEXT`*

(Obligatoire) Le nom de la CloudFormation pile sur laquelle vous effectuez le déploiement. Si vous spécifiez une pile existante, la commande met à jour la pile. Si vous spécifiez une nouvelle pile, celle-ci est créée par la commande.

`--tags` *`LIST`*

Liste de balises à associer à la pile créée ou mise à jour. CloudFormation propage également ces balises aux ressources de la pile qui les supportent.

`--template-file, --template, -t` *`PATH`*

Le chemin et le nom du fichier où se trouve votre AWS SAM modèle.

> **Note**
>
> Si vous spécifiez cette option, AWS SAM déploie uniquement le modèle et les ressources locales vers lesquels il pointe.

`--use-json`

Sortie JSON pour le CloudFormation modèle. La sortie par défaut est YAML.

Exemple

Pour un exemple détaillé et une présentation détaillée de l'utilisation de la `sam deploy` sous-commande, reportez-vous à. Présentation du déploiement avec AWS SAM

sam init

Cette page fournit des informations de référence pour la AWS Serverless Application Model commande Command Line Interface (AWS SAMCLI)`sam init`.

- Pour une introduction à la AWS SAMCLI, voir Qu'est-ce que c'est AWS SAMCLI ?
- Pour obtenir de la documentation sur l'utilisation de la AWS SAMCLI `sam init` commande, consultezCréez votre application dans AWS SAM.

La commande `sam init` fournit des options pour initialiser une nouvelle application sans serveur.

Usage

```
$ sam init <options>
```

Options

`--app-template` *`TEXT`*

L'identificateur du modèle d'application gérée que vous souhaitez utiliser. En cas de doute, appelez `sam init` sans options pour un flux interactif.

Ce paramètre est obligatoire si `--no-interactive` est spécifié et que `--location` n'est pas fourni.

Ce paramètre est uniquement disponible dans la version 0.30.0 et ultérieures de la CLI AWS SAM. La spécification de ce paramètre avec une version antérieure une erreur se produit.

`--application-insights | --no-application-insights`

Activez la surveillance Amazon CloudWatch Application Insights pour votre application. Pour en savoir plus, veuillez consulter la section Utilisation d' CloudWatch Application Insights pour surveiller vos applications AWS SAM sans serveur.

L'option par défaut est `--no-application-insights`.

`--architecture, -a` *`[ x86_64 | arm64 ]`*

Architecture du jeu d'instructions pour les fonctions Lambda de l'application. Spécifiez l'un des `x86_64` ou `arm64`.

`--base-image` *[amazon/dotnet8-base | amazon/dotnet6-base | amazon/java25-base | amazon/java21-base | amazon/java17-base | amazon/java11-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/ruby3.4-base | amazon/ruby3.3-base | amazon/ruby3.2-base]*

L'image de base de l'application. Cette option s'applique uniquement lorsque le type de package est `Image`.

Ce paramètre est obligatoire si `--no-interactive` est spécifié, `--package-type` est spécifié comme `Image` et `--location` n'est pas spécifié.

`--config-env` *TEXT*

Le nom d'environnement spécifiant les valeurs de paramètre par défaut dans le fichier de configuration à utiliser. La valeur par défaut est « par défaut ». Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--config-file` *PATH*

Le chemin d'accès et le nom du fichier de configuration contenant les valeurs de paramètres par défaut à utiliser. La valeur par défaut est « samconfig.toml » dans la racine du répertoire du projet. Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--debug`

Active la journalisation de débogage pour imprimer les messages de débogage générés par la CLI AWS SAM et pour afficher les horodatages.

`--dependency-manager, -d` *[gradle | mod | maven | bundler | npm | cli-package | pip]*

Le gestionnaire de dépendances de l'exécution Lambda.

`--extra-content`

Remplacez tous les paramètres personnalisés dans la configuration du modèle `cookiecutter.json`, par exemple, `{"customParam1": "customValue1", "customParam2":"customValue2"}`

`--help, -h`

Affiche ce message, puis se ferme.

`--location, -l` *`TEXT`*

Le modèle ou l'emplacement de l'application (Git, Mercurial, HTTP/HTTPS, fichier .zip, chemin d'accès).

Ce paramètre est obligatoire si `--no-interactive` est spécifié et `--runtime`, `--name` et `--app-template` ne sont pas fournis.

Pour les référentiels Git, vous devez utiliser l'emplacement de la racine du référentiel.

Pour les chemins d'accès locaux, le modèle doit être dans un fichier .zip ou en format Cookiecutter.

`--name, -n` *`TEXT`*

Le nom du projet à générer en tant que répertoire.

Ce paramètre est obligatoire si `--no-interactive` est spécifié et `--location` n'est pas fourni.

`--no-input`

Désactive l'invite Cookiecutter et accepte les valeurs vcfdefault définies dans la configuration du modèle.

`--no-interactive`

Désactivez l'invite interactive pour les paramètres de démarrage et attendez si des valeurs requises sont manquantes.

`--output-dir, -o` *`PATH`*

L'emplacement où l'application initialisée est sortie.

`--package-type` *`[ Zip | Image ]`*

Le type de package de l'exemple d'application. `Zip` crée une archive de fichiers .zip, puis `Image` crée une image de conteneur.

`--runtime, -r` *[dotnet8 | dotnet6 | java25 | java21 | java17 | java11 | nodejs24.x | nodejs22.x | nodejs20.x | nodejs18.x | nodejs16.x | python3.14 | python3.13 | python3.12 | python3.11 | python3.10 | python3.9 | python3.8 | ruby3.4 | ruby3.3 | ruby3.2]*

L'exécution Lambda de l'application. Cette option s'applique uniquement lorsque le type de package est `Zip`.

Ce paramètre est obligatoire si `--no-interactive` est spécifié, `--package-type` est spécifié comme `Zip` et `--location` n'est pas spécifié.

`--save-params`

Enregistrez les paramètres que vous fournissez sur la ligne de commande dans le fichier AWS SAM de configuration.

`--tracing | --no-tracing`

Activez le AWS X-Ray suivi pour vos fonctions Lambda.

Exemple

Pour un exemple détaillé et une présentation détaillée de l'utilisation de la `sam init` sous-commande, reportez-vous à. Créez votre application dans AWS SAM

sam list

Cette page fournit des informations de référence pour la AWS Serverless Application Model commande Command Line Interface (AWS SAMCLI)`sam list`.

Pour une introduction à la AWS SAMCLI, voir Qu'est-ce que c'est AWS SAMCLI ?

La `sam list` commande génère des informations importantes sur les ressources de votre application sans serveur et sur son état. Utiliser sam list avant et après le déploiement pour faciliter le développement local et dans le cloud.

Usage

```
$ sam list <options> <subcommand>
```

Options

`--help, -h`

Affichez ce message et quittez.

Sous-commandes

`endpoints`

Affiche la liste des points de terminaison cloud et locaux de votre CloudFormation stack. Pour de plus amples informations, veuillez consulter sam list endpoints.

`resources`

Affiche les ressources de votre modèle AWS Serverless Application Model (AWS SAM) créées AWS CloudFormation lors du déploiement. Pour de plus amples informations, veuillez consulter sam list resources.

`stack-outputs`

Affiche les sorties de votre CloudFormation pile à partir d'un CloudFormation modèle AWS SAM or. Pour de plus amples informations, veuillez consulter sam list stack-outputs.

sam list endpoints

Cette page fournit des informations de référence pour la AWS Serverless Application Model `sam list endpoints` sous-commande Command Line Interface (AWS SAMCLI).

Pour une introduction à la AWS SAMCLI, voir Qu'est-ce que c'est AWS SAMCLI ?

La `sam list endpoints` sous-commande affiche une liste des points de terminaison cloud et locaux de votre CloudFormation stack. Vous pouvez interagir avec ces ressources à l'aide des commandes sam local et sam sync.

AWS Lambda et les types de ressources Amazon API Gateway sont pris en charge par cette commande.

> **Note**
>
> Les domaines personnalisés sont pris en charge lorsqu'ils sont configurés pour vos ressources Amazon API Gateway. Cette commande affichera le domaine personnalisé au lieu du point de terminaison par défaut.

Usage

```
$ sam list endpoints <options>
```

Options

`--config-env` *TEXT*

Le nom d'environnement spécifiant les valeurs de paramètre par défaut dans le fichier de configuration à utiliser.

Valeur par défaut : `default`

Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--config-file` *TEXT*

Le chemin d'accès et le nom du fichier de configuration contenant les valeurs de paramètres par défaut à utiliser.

Valeur par défaut : `samconfig.toml` dans le répertoire de travail actuel.

Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--debug`

Active la journalisation de débogage pour imprimer le message de débogage généré par la CLI AWS SAM avec les horodatages.

`--help, -h`

Affichez ce message et quittez.

`--output [json|table]`

Spécifiez le format de sortie des résultats.

Valeur par défaut : `table`

`--profile` *TEXT*

Sélectionnez un profil spécifique dans votre fichier d'informations d'identification pour obtenir des AWS informations d'identification.

`--region` *TEXT*

Définissez la AWS région du service. Par exemple, `us-east-1`.

`--save-params`

Enregistrez les paramètres que vous fournissez sur la ligne de commande dans le fichier AWS SAM de configuration.

`--stack-name` *TEXT*

Nom de la CloudFormation pile déployée. Le nom de la pile se trouve dans le fichier `samconfig.toml` de votre application ou dans le fichier de configuration désigné.

Lorsque cette option n'est pas spécifiée, les ressources locales définies dans votre modèle s'affichent.

`--template-file, --template, -t` *PATH*

AWS SAM fichier modèle.

Valeur par défaut : `template.[yaml|yml|json]`

Exemples

Affichez une sortie, au format json, des points de terminaison des ressources déployées à partir de votre CloudFormation pile nommée`test-stack`.

```
$ 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

Cette page fournit des informations de référence pour la AWS Serverless Application Model `sam list resources` sous-commande Command Line Interface (AWS SAMCLI).

Pour une introduction à la AWS SAMCLI, voir Qu'est-ce que c'est AWS SAMCLI ?

La `sam list resources` sous-commande affiche les ressources de votre modèle AWS Serverless Application Model (AWS SAM) créées AWS CloudFormation par la AWS SAM transformation lors du déploiement.

sam list resourcesÀ utiliser avec un AWS SAM modèle avant le déploiement pour voir les ressources qui seront créées. Entrez un nom de CloudFormation pile pour afficher une liste consolidée incluant les ressources déployées.

> **Note**
>
> Pour générer une liste de ressources à partir de votre AWS SAM modèle, une transformation locale de votre modèle est effectuée. Les ressources qui seront déployées sous certaines conditions, par exemple dans une région spécifique, sont incluses dans cette liste.

Usage

```
$ sam list resources <options>
```

Options

`--config-env` *TEXT*

Le nom d'environnement spécifiant les valeurs de paramètre par défaut dans le fichier de configuration à utiliser.

Valeur par défaut : `default`

Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--config-file` *TEXT*

Le chemin d'accès et le nom du fichier de configuration contenant les valeurs de paramètres par défaut à utiliser.

Valeur par défaut : `samconfig.toml` dans le répertoire de travail actuel.

Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--debug`

Active la journalisation de débogage pour imprimer le message de débogage généré par la CLI AWS SAM avec les horodatages.

`--help, -h`

Affichez ce message et quittez.

`--output [json|table]`

Spécifiez le format de sortie des résultats.

Valeur par défaut : `table`

`--profile` *TEXT*

Sélectionnez un profil spécifique dans votre fichier d'informations d'identification pour obtenir des AWS informations d'identification.

`--region` *TEXT*

Définissez la AWS région du service. Par exemple, `us-east-1`.

`--save-params`

Enregistrez les paramètres que vous fournissez sur la ligne de commande dans le fichier AWS SAM de configuration.

`--stack-name` *`TEXT`*

Nom de la CloudFormation pile déployée. Le nom de la pile se trouve dans le fichier `samconfig.toml` de votre application ou dans le fichier de configuration désigné.

Lorsqu'elle est fournie, la logique IDs des ressources de votre modèle sera mappée à leur entrée physique IDs correspondante. CloudFormation Pour en savoir plus sur les ressources physiques IDs, consultez la section Champs de ressources dans le guide de AWS CloudFormation l'utilisateur.

Lorsque cette option n'est pas spécifiée, les ressources locales définies dans votre modèle s'affichent.

`--template-file, --template, -t` *`PATH`*

AWS SAM fichier modèle.

Valeur par défaut : `template.[yaml|yml|json]`

Exemples

Affichez une sortie, sous forme de tableau, des ressources locales de votre AWS SAM modèle et des ressources déployées à partir de votre CloudFormation pile nommée`test-stack`. Exécuter à partir du même répertoire que votre fichier local.

```
$ 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
ServerlessRestApiDeploymentf5716dc08b                        -
-----------------------------------------------------------------------------------------------------------
```

sam list stack-outputs

Cette page fournit des informations de référence pour la AWS Serverless Application Model `sam list stack-outputs` sous-commande Command Line Interface (AWS SAMCLI).

Pour une introduction à la AWS SAMCLI, voir Qu'est-ce que c'est AWS SAMCLI ?

La `sam list stack-outputs` sous-commande affiche les sorties de votre AWS CloudFormation pile à partir d'un AWS Serverless Application Model (AWS SAM) ou d'un CloudFormation modèle. Pour plus d'informations sur `Outputs`, consultez Sorties dans le Guide de l'utilisateur AWS CloudFormation .

Usage

```
$ sam list stack-outputs <options>
```

Options

`--config-env` *`TEXT`*

Le nom d'environnement spécifiant les valeurs de paramètre par défaut dans le fichier de configuration à utiliser.

Valeur par défaut : `default`

Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--config-file` *`TEXT`*

Le chemin d'accès et le nom du fichier de configuration contenant les valeurs de paramètres par défaut à utiliser.

Valeur par défaut : `samconfig.toml` dans le répertoire de travail actuel.

Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--debug`

Active la journalisation de débogage pour imprimer le message de débogage généré par la CLI AWS SAM avec les horodatages.

`--help, -h`

Affichez ce message et quittez.

`--output [json|table]`

Spécifiez le format de sortie des résultats.

Valeur par défaut : `table`

`--profile` *TEXT*

Sélectionnez un profil spécifique dans votre fichier d'informations d'identification pour obtenir des AWS informations d'identification.

`--region` *TEXT*

Définissez la AWS région du service. Par exemple, `us-east-1`.

`--save-params`

Enregistrez les paramètres que vous fournissez sur la ligne de commande dans le fichier AWS SAM de configuration.

`--stack-name` *TEXT*

Nom de la CloudFormation pile déployée. Le nom de la pile se trouve dans le fichier `samconfig.toml` de votre application ou dans le fichier de configuration désigné.

Cette option est obligatoire.

Exemples

Affiche les sorties, sous forme de tableau, des ressources de votre CloudFormation pile nomméetest-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

Cette page fournit des informations de référence pour la AWS Serverless Application Model commande Command Line Interface (AWS SAMCLI)`sam local callback`.

Pour une introduction à la AWS SAMCLI, voir Qu'est-ce que c'est AWS SAMCLI ?

La `sam local callback` commande vous permet d'envoyer des rappels aux exécutions de fonctions durables locales pendant le développement et les tests.

Usage

```
$ sam local callback <subcommand> <options>
```

Options

`--help, -h`

Affichez ce message et quittez.

Sous-commandes

`succeed`

Envoyez un rappel de réussite à l'exécution d'une fonction durable. Pour de plus amples informations, veuillez consulter sam local callback succeed.

`fail`

Envoyez un rappel d'échec à une exécution de fonction durable. Pour de plus amples informations, veuillez consulter sam local callback fail.

`heartbeat`

Envoyez un rappel du rythme cardiaque à une exécution de fonction durable. Pour de plus amples informations, veuillez consulter sam local callback heartbeat.

sam local callback succeed

Envoyez un rappel de réussite à l'exécution d'une fonction durable.

Usage

```
$ sam local callback succeed CALLBACK_ID [OPTIONS]
```

Arguments obligatoires

`CALLBACK_ID`

Identifiant unique du rappel auquel envoyer une réponse positive.

Options

`-r, --result TEXT`

Charge utile du résultat de réussite sous forme de chaîne.

`--region TEXT`

Définissez la AWS région du service (par exemple,`us-east-1`).

`--profile TEXT`

Sélectionnez un profil spécifique dans votre fichier d'informations d'identification pour obtenir des AWS informations d'identification.

`--config-env TEXT`

Nom de l'environnement spécifiant les valeurs des paramètres par défaut dans le fichier de configuration. Valeur par défaut : `default`

`--config-file TEXT`

Fichier de configuration contenant les valeurs des paramètres par défaut. Valeur par défaut : `samconfig.toml`

`--save-params`

Enregistrez les paramètres fournis via la ligne de commande dans le fichier de configuration.

`--beta-features / --no-beta-features`

Activer/désactiver les fonctionnalités bêta.

`--debug`

Activez la journalisation du débogage pour imprimer les messages de débogage générés par la AWS SAM CLI et afficher les horodatages.

`-h, --help`

Affiche ce message, puis se ferme.

Exemples

Envoyer un rappel de réussite sans résultat :

```
$ sam local callback succeed my-callback-id
```

Envoyer un rappel de réussite avec le résultat :

```
$ sam local callback succeed my-callback-id --result 'Task completed successfully'
```

Envoyer un rappel de réussite avec une option courte :

```
$ sam local callback succeed my-callback-id -r 'Success result'
```

sam local callback fail

Envoyez un rappel d'échec à une exécution de fonction durable.

Usage

```
$ sam local callback fail CALLBACK_ID [OPTIONS]
```

Arguments obligatoires

`CALLBACK_ID`

Identifiant unique du rappel auquel envoyer une réponse d'échec.

Options

`--error-data TEXT`

Données d'erreur supplémentaires.

`--stack-trace TEXT`

Empilez les entrées de suivi. Cette option peut être spécifiée plusieurs fois.

`--error-type TEXT`

Type d'erreur.

`--error-message TEXT`

Message d'erreur détaillé.

`--region TEXT`

Définissez la AWS région du service (par exemple,`us-east-1`).

`--profile TEXT`

Sélectionnez un profil spécifique dans votre fichier d'informations d'identification pour obtenir des AWS informations d'identification.

`--config-env TEXT`

Nom de l'environnement spécifiant les valeurs des paramètres par défaut dans le fichier de configuration. Valeur par défaut : `default`

`--config-file TEXT`

Fichier de configuration contenant les valeurs des paramètres par défaut. Valeur par défaut : `samconfig.toml`

`--save-params`

Enregistrez les paramètres fournis via la ligne de commande dans le fichier de configuration.

`--beta-features / --no-beta-features`

Activer/désactiver les fonctionnalités bêta.

`--debug`

Activez la journalisation du débogage pour imprimer les messages de débogage générés par la AWS SAM CLI et afficher les horodatages.

`-h, --help`

Affiche ce message, puis se ferme.

Exemples

Envoyer un rappel en cas d'échec sans paramètres :

```
$ sam local callback fail my-callback-id
```

Envoyer un rappel d'échec avec un message d'erreur :

```
$ sam local callback fail my-callback-id --error-message 'Task failed'
```

Envoyer un rappel en cas d'échec avec tous les paramètres :

```
$ 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

Envoyez un rappel du rythme cardiaque à une exécution de fonction durable.

Usage

```
$ sam local callback heartbeat CALLBACK_ID [OPTIONS]
```

Arguments obligatoires

`CALLBACK_ID`

Identifiant unique du rappel auquel envoyer un battement de cœur.

Options

`--region TEXT`

Définissez la AWS région du service (par exemple,`us-east-1`).

`--profile TEXT`

Sélectionnez un profil spécifique dans votre fichier d'informations d'identification pour obtenir des AWS informations d'identification.

`--config-env TEXT`

Nom de l'environnement spécifiant les valeurs des paramètres par défaut dans le fichier de configuration. Valeur par défaut : `default`

`--config-file TEXT`

Fichier de configuration contenant les valeurs des paramètres par défaut. Valeur par défaut : `samconfig.toml`

`--save-params`

Enregistrez les paramètres fournis via la ligne de commande dans le fichier de configuration.

`--beta-features / --no-beta-features`

Activer/désactiver les fonctionnalités bêta.

`--debug`

Activez la journalisation du débogage pour imprimer les messages de débogage générés par la AWS SAM CLI et afficher les horodatages.

`-h, --help`

Affiche ce message, puis se ferme.

Exemples

Envoyer un rappel du rythme cardiaque :

```
$ sam local callback heartbeat my-callback-id
```

sam local execution

Cette page fournit des informations de référence pour la AWS Serverless Application Model commande Command Line Interface (AWS SAMCLI)`sam local execution`.

Pour une introduction à la AWS SAMCLI, voir Qu'est-ce que c'est AWS SAMCLI ?

La `sam local execution` commande vous permet de gérer et d'inspecter les exécutions de fonctions durables locales pendant le développement et les tests.

> **Note**
>
> Il est possible que ces commandes ne nécessitent pas l'accès aux AWS informations d'identification.

Usage

```
$ sam local execution <subcommand> <options>
```

Options

`--help, -h`

Affichez ce message et quittez.

Sous-commandes

`get`

Obtenez des informations sur l'exécution durable d'une fonction. Pour de plus amples informations, veuillez consulter sam local execution get.

`history`

Obtenez l'historique d'exécution d'une fonction durable. Pour de plus amples informations, veuillez consulter sam local execution history.

`stop`

Arrêtez l'exécution d'une fonction durable. Pour de plus amples informations, veuillez consulter sam local execution stop.

sam local execution get

Récupérez les détails de l'exécution d'une fonction durable spécifique.

> **Note**
>
> Il est possible que cette commande ne nécessite pas d'accès aux AWS informations d'identification.

Usage

```
$ sam local execution get DURABLE_EXECUTION_ARN [OPTIONS]
```

Arguments obligatoires

`DURABLE_EXECUTION_ARN`

Le nom de ressource Amazon (ARN) de la fonction durable à exécuter pour laquelle vous souhaitez obtenir des informations.

Options

`--format [summary|json]`

Format de la sortie. Valeur par défaut : `summary`

`--region TEXT`

Définissez la AWS région du service (par exemple,`us-east-1`).

`--profile TEXT`

Sélectionnez un profil spécifique dans votre fichier d'informations d'identification pour obtenir des AWS informations d'identification.

`--config-env TEXT`

Nom de l'environnement spécifiant les valeurs des paramètres par défaut dans le fichier de configuration. Valeur par défaut : `default`

`--config-file TEXT`

Fichier de configuration contenant les valeurs des paramètres par défaut. Valeur par défaut : `samconfig.toml`

`--save-params`

Enregistrez les paramètres fournis via la ligne de commande dans le fichier de configuration.

`--beta-features / --no-beta-features`

Activer/désactiver les fonctionnalités bêta.

`--debug`

Activez la journalisation du débogage pour imprimer les messages de débogage générés par la AWS SAM CLI et afficher les horodatages.

`-h, --help`

Affiche ce message, puis se ferme.

Exemples

Obtenez les détails de l'exécution avec le format de résumé par défaut :

```
$ sam local execution get arn:aws:lambda:us-east-1:123456789012:function:MyFunction:
$LATEST/durable-execution/my-execution-name/my-execution-id
```

Obtenez les détails de l'exécution au format JSON :

```
$ 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

Obtenez l'historique d'exécution d'une fonction durable.

Usage

```
$ sam local execution history EXECUTION_ID [OPTIONS]
```

Arguments obligatoires

`EXECUTION_ID`

Identifiant unique de l'exécution de la fonction durable pour laquelle il faut récupérer l'historique.

Options

`--format [table|json]`

Format de la sortie. Valeur par défaut : `table`

`--region TEXT`

Définissez la AWS région du service (par exemple,`us-east-1`).

`--profile TEXT`

Sélectionnez un profil spécifique dans votre fichier d'informations d'identification pour obtenir des AWS informations d'identification.

`--config-env TEXT`

Nom de l'environnement spécifiant les valeurs des paramètres par défaut dans le fichier de configuration. Valeur par défaut : `default`

`--config-file TEXT`

Fichier de configuration contenant les valeurs des paramètres par défaut. Valeur par défaut : `samconfig.toml`

`--save-params`

Enregistrez les paramètres fournis via la ligne de commande dans le fichier de configuration.

`--beta-features / --no-beta-features`

Activer/désactiver les fonctionnalités bêta.

`--debug`

Activez la journalisation du débogage pour imprimer les messages de débogage générés par la AWS SAM CLI et afficher les horodatages.

`-h, --help`

Affiche ce message, puis se ferme.

Exemples

Obtenez l'historique des exécutions sous forme de tableau :

```
$ sam local execution history my-execution-id
```

Obtenez l'historique des exécutions au format JSON :

```
$ sam local execution history my-execution-id --format json
```

sam local execution stop

Arrêtez l'exécution d'une fonction durable en cours d'exécution.

> **Note**
>
> Il est possible que cette commande ne nécessite pas d'accès aux AWS informations d'identification.

Usage

```
$ sam local execution stop DURABLE_EXECUTION_ARN [OPTIONS]
```

Arguments obligatoires

`DURABLE_EXECUTION_ARN`

L'Amazon Resource Name (ARN) de l'exécution de la fonction durable à arrêter.

Options

`--error-message TEXT`

Message d'erreur à associer à l'arrêt de l'exécution.

`--error-type TEXT`

Type d'erreur à associer à l'arrêt de l'exécution.

`--error-data TEXT`

Données d'erreur à associer à l'arrêt de l'exécution.

`--stack-trace TEXT`

Empilez les entrées de suivi. Cette option peut être spécifiée plusieurs fois.

`--region TEXT`

Définissez la AWS région du service (par exemple,`us-east-1`).

`--profile TEXT`

Sélectionnez un profil spécifique dans votre fichier d'informations d'identification pour obtenir des AWS informations d'identification.

`--config-env TEXT`

Nom de l'environnement spécifiant les valeurs des paramètres par défaut dans le fichier de configuration. Valeur par défaut : `default`

`--config-file TEXT`

Fichier de configuration contenant les valeurs des paramètres par défaut. Valeur par défaut : `samconfig.toml`

`--save-params`

Enregistrez les paramètres fournis via la ligne de commande dans le fichier de configuration.

`--beta-features / --no-beta-features`

Activer/désactiver les fonctionnalités bêta.

`--debug`

Activez la journalisation du débogage pour imprimer les messages de débogage générés par la AWS SAM CLI et afficher les horodatages.

`-h, --help`

Affiche ce message, puis se ferme.

Exemples

Arrêtez l'exécution sans informations détaillées sur l'erreur :

```
$ sam local execution stop arn:aws:lambda:us-east-1:123456789012:function:MyFunction:
$LATEST/durable-execution/my-execution-name/my-execution-id
```

Arrêtez l'exécution avec un message d'erreur et tapez :

```
$ 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"
```

Arrêtez l'exécution en indiquant tous les détails de l'erreur et en enregistrant la 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

Cette page fournit des informations de référence pour la AWS Serverless Application Model `sam local generate-event` sous-commande Command Line Interface (AWS SAMCLI).

- Pour une introduction à la AWS SAMCLI, voir Qu'est-ce que c'est AWS SAMCLI ?
- Pour obtenir de la documentation sur l'utilisation de la AWS SAMCLI `sam local generate-event` commande, consultezPrésentation des tests avec sam local generate-event.

La sous-commande `sam local generate-event` génère des échantillons d'évènement de charge utile pour Services AWS prise en charge.

Usage

```
$ sam local generate-event <options> <service> <event> <event-options>
```

Options

`--config-env TEXT`

Le nom d'environnement spécifiant les valeurs de paramètre par défaut dans le fichier de configuration à utiliser. La valeur par défaut est « par défaut ». Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--config-file PATH`

Le chemin d'accès et le nom du fichier de configuration contenant les valeurs de paramètres par défaut à utiliser. La valeur par défaut est `samconfig.toml` à la racine du répertoire du projet. Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--help`

Affiche ce message, puis se ferme.

Service

Pour obtenir la liste des services pris en charge, exécutez la procédure suivante :

```
$ sam local generate-event
```

Événement

Pour obtenir la liste des événements pris en charge pouvant être générés pour chaque service, exécutez la procédure suivante :

```
$ sam local generate-event <service>
```

Options d'événement

Pour obtenir une liste des options d'événement prises en charge que vous pouvez modifier, exécutez la procédure suivante :

```
$ sam local generate-event <service> <event> --help
```

Exemples

Pour des exemples d'utilisation de la `sam local generate-event` sous-commande, reportez-vous àGénérer des exemples d'événements.

sam local invoke

Cette page fournit des informations de référence pour la AWS Serverless Application Model `sam local invoke` sous-commande Command Line Interface (AWS SAMCLI).

- Pour une introduction à la AWS SAMCLI, voir Qu'est-ce que c'est AWS SAMCLI ?
- Pour obtenir de la documentation sur l'utilisation de la AWS SAMCLI `sam local invoke` sous-commande, consultezPrésentation des tests avec sam local invoke.

La `sam local invoke` sous-commande lance un appel unique d'une fonction localement. AWS Lambda

> **Note**
>
> Pour des fonctions durables, `sam local invoke` prend en charge l'exécution dynamique avec pointage automatique et rediffusion. Le conteneur continue de fonctionner pendant l'exécution d'une fonction durable afin de gérer la persistance et la reprise de l'état.

Usage

```
$ sam local invoke <arguments> <options>
```

> **Note**
>
> Si plusieurs fonctions sont définies dans votre AWS SAM modèle, indiquez l'ID logique de la fonction que vous souhaitez invoquer.

Arguments

ID de ressource

ID de la fonction Lambda à appeler.

Cet argument est facultatif. Si votre application contient une seule fonction Lambda, la AWS SAM CLI l'invoquera. Si votre application contient plusieurs fonctions, indiquez l'ID de la fonction à invoquer.

Valeurs valides : ID logique ou ARN de la ressource.

Options

`--add-host` *`LIST`*

Transmet un mappage de nom d'hôte à adresse IP au fichier hôte du conteneur Docker. Ce paramètre peut être transmis plusieurs fois.

Example

Exemple : `--add-host` *`example.com:127.0.0.1`*

`--beta-features | --no-beta-features`

Autorisez ou refusez les fonctionnalités bêta.

`--config-env` *`TEXT`*

Le nom d'environnement spécifiant les valeurs de paramètre par défaut dans le fichier de configuration à utiliser. La valeur par défaut est « par défaut ». Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--config-file` *`PATH`*

Le chemin d'accès et le nom du fichier de configuration contenant les valeurs de paramètres par défaut à utiliser. La valeur par défaut est « `samconfig.toml` » dans la racine du répertoire du projet. Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--container-env-vars`

(Facultatif) Transmettez les variables d'environnement au conteneur d'images de la fonction Lambda lors du débogage local.

`--container-host` *`TEXT`*

Hôte du conteneur Lambda émulé localement. La valeur par défaut est `localhost`. Si vous voulez exécuter la CLI AWS SAM dans un conteneur Docker sous macOS, vous pouvez spécifier `host.docker.internal`. Si vous souhaitez exécuter le conteneur sur un autre hôte AWS SAMCLI, vous pouvez spécifier l'adresse IP de l'hôte distant.

`--container-host-interface` *`TEXT`*

L'adresse IP de l'interface de réseau hôte à laquelle les ports de conteneur doivent se relier. La valeur par défaut est `127.0.0.1`. Utiliser `0.0.0.0` pour se lier à toutes les interfaces.

`--debug`

Active la journalisation de débogage pour imprimer les messages de débogage générés par la CLI AWS SAM et pour afficher les horodatages.

`--debug-args` *TEXT*

Arguments facultatifs à transmettre au débogueur.

`--debug-port, -d` *TEXT*

Si spécifié, cela démarre le conteneur de la fonction Lambda en mode débogage et expose ce port sur l'hôte local.

`--debugger-path` *TEXT*

Chemin d'accès de l'hôte vers un débogueur monté dans le conteneur Lambda.

`--docker-network` *TEXT*

Le nom ou l'identifiant d'un réseau Docker existant auquel les conteneurs Docker Lambda doivent se connecter, avec le réseau de pont par défaut. Si cela n'est pas spécifié, les conteneurs Lambda se connectent uniquement au réseau Docker de pont par défaut.

`--docker-volume-basedir, -v` *TEXT*

Emplacement du répertoire de base dans lequel se trouve le AWS SAM fichier. Si Docker est exécuté sur une machine distante, vous devez monter le chemin où se trouve le AWS SAM fichier sur la machine Docker et modifier cette valeur pour qu'elle corresponde à la machine distante.

`--durable-execution-name` *TEXT*

Nom de l'exécution durable (pour les fonctions durables uniquement)

`--env-vars, -n` *PATH*

Le fichier JSON contenant les valeurs des variables d'environnement de la fonction Lambda. Pour plus d'informations sur les fichiers de variables d'environnement, consultez Fichier de variable d'environnement.

`--event, -e` *PATH*

Le fichier JSON qui contient les données d'événement transmises à la fonction Lambda lorsqu'elle est appelée. Si vous ne spécifiez pas cette option, aucun événement n'est assumé. Pour entrer JSON à partir de `stdin`, vous devez transmettre la valeur '-'. Pour plus d'informations sur les

formats de messages d'événements provenant de différents AWS services, consultez la section Utilisation d'autres services dans le Guide du AWS Lambda développeur.

`--force-image-build`

Spécifie si la CLI AWS SAM doit recréer l'image utilisée pour invoquer les fonctions Lambda avec des couches.

`--help`

Affiche ce message, puis se ferme.

`--hook-name TEXT`

Nom du hook utilisé pour étendre la fonctionnalité de la CLI AWS SAM.

Valeurs acceptées : `terraform`.

`--invoke-image` *`TEXT`*

L'URI de l'image de conteneur que vous souhaitez utiliser pour l'appel de la fonction locale. Par défaut, AWS SAM extrait l'image du conteneur d'Amazon ECR Public (répertoriée dansRéférentiels d'images pour AWS SAM). Utilisez cette option pour extraire l'image à partir d'un autre emplacement.

Par exemple, `sam local invoke MyFunction --invoke-image amazon/aws-sam-cli-emulation-image-python3.8`.

`--layer-cache-basedir` *`DIRECTORY`*

Spécifie l'emplacement du répertoire de base dans lequel les couches utilisées par le modèle sont téléchargées.

`--log-file, -l` *`TEXT`*

Le fichier journal vers lequel envoyer les journaux d'exécution.

`--mount-symlinks`

Garantit le AWS SAMCLI montage permanent des liens symboliques présents dans les fichiers à créer ou à invoquer. Cela s'applique uniquement aux liens symboliques du répertoire de niveau supérieur (c'est-à-dire aux liens symboliques situés directement à la racine de la fonction). Par défaut, les liens symboliques ne sont pas montés, à l'exception de ceux nécessaires pour être utilisés `node_modules` dans `build-in-source` NodeJS.

`--no-event`

Appelle la fonction avec un événement vide.

`--no-memory-limit`

Supprime la limite de mémoire dans le conteneur lors de l'appel local, même lorsque la mémoire est configurée dans le AWS SAM modèle.

`--parameter-overrides`

Chaîne contenant des remplacements de CloudFormation paramètres codés sous forme de paires clé-valeur. Utilisez le même format que le AWS Command Line Interface (AWS CLI). Le AWS SAMCLI format est basé sur des mots clés et valeurs explicites, chaque remplacement étant séparé par un espace. Voici deux exemples :

- `--parameter-overrides ParameterKey=hello,ParameterValue=world`
- `--parameter-overrides ParameterKey=hello,ParameterValue=world ParameterKey=example1,ParameterValue=example2 ParameterKey=apple,ParameterValue=banana`

`--profile` *`TEXT`*

Le profil spécifique de votre fichier d'informations d'identification qui obtient les AWS informations d'identification.

`--region` *`TEXT`*

La AWS région dans laquelle le déploiement doit être effectué. Par exemple, us-east-1.

`--runtime` *`TEXT`*

Utilise le runtime spécifié pour appeler une fonction Lambda localement. Cela remplace le temps d'exécution défini dans le `template.yml` fichier. Cela permet également de tester les fonctions Lambda avec différents temps d'exécution sans modifier la configuration de la fonction d'origine.

`--save-params`

Enregistrez les paramètres que vous fournissez sur la ligne de commande dans le fichier AWS SAM de configuration.

`--shutdown`

Imite un événement d’arrêt une fois l’appel terminé, afin de tester la gestion des extensions du comportement d’arrêt.

`--skip-prepare-infra`

Ignore l'étape de préparation si aucune modification d'infrastructure n'a été apportée. À utiliser avec l'option `--hook-name`.

`--skip-pull-image`

Par défaut, la CLI AWS SAM vérifie le dernier environnement d'exécution à distance de Lambda et met automatiquement à jour votre image locale pour rester synchronisée.

Spécifiez cette option pour ne pas extraire la dernière image Docker de votre environnement d'exécution Lambda.

`--template, -t` *PATH*

Le fichier AWS SAM modèle.

Cette option n'est pas compatible avec `--hook-name`.

> **Note**
>
> Si vous spécifiez cette option, AWS SAM charge uniquement le modèle et les ressources locales vers lesquels il pointe.

`--tenant-id` *TEXT*

L'ID de locataire pour les fonctions Lambda à locataires multiples. Utilisé pour garantir l'isolation informatique entre les différents locataires. Obligatoire lors de l'appel de fonctions configurées avec le mode d'isolation des locataires.

`--terraform-plan-file`

Le chemin relatif ou absolu vers votre fichier de plan Terraform local lorsque vous utilisez la CLI AWS SAM avec Terraform Cloud. Cette option nécessite que `--hook-name` soit défini sur `terraform`.

Exemples

L'exemple suivant utilise un événement généré à des fins de test local en utilisant un `s3.json` événement pour appeler une fonction Lambda localement.

```
$ sam local invoke --event events/s3.json S3JsonLoggerFunction
```

L'exemple suivant teste la fonction à l' `HelloWorldFunction`aide de l'environnement d'exécution Python 3.11

```
$ sam local invoke --runtime python3.11 HelloWorldFunction
```

L'exemple suivant teste la fonction `HelloWorldFunction` avec un nom d'exécution durable

```
$ sam local invoke HelloWorldFunction --durable-execution-name my-execution
```

sam local start-api

Cette page fournit des informations de référence pour la AWS Serverless Application Model `sam local start-api` sous-commande Command Line Interface (AWS SAMCLI).

- Pour une introduction à la AWS SAMCLI, voir Qu'est-ce que c'est AWS SAMCLI ?
- Pour obtenir de la documentation sur l'utilisation de la AWS SAMCLI `sam local start-api` sous-commande, consultezPrésentation des tests avec sam local start-api.

La `sam local start-api` sous-commande exécute vos AWS Lambda fonctions localement pour les tester via un hôte de serveur HTTP local.

Usage

```
$ sam local start-api <options>
```

Options

`--add-host` *`LIST`*

Transmet un mappage de nom d'hôte à adresse IP au fichier hôte du conteneur Docker. Ce paramètre peut être transmis plusieurs fois.

Example

Exemple : `--add-host` *`example.com:127.0.0.1`*

`--beta-features | --no-beta-features`

Autorisez ou refusez les fonctionnalités bêta.

`--config-env` *TEXT*

Le nom d'environnement spécifiant les valeurs de paramètre par défaut dans le fichier de configuration à utiliser. La valeur par défaut est « par défaut ». Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--config-file` *PATH*

Le chemin d'accès et le nom du fichier de configuration contenant les valeurs de paramètres par défaut à utiliser. La valeur par défaut est « samconfig.toml » dans la racine du répertoire du projet. Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--container-env-vars`

Facultatif. Transmettez les variables d'environnement au conteneur d'images lors du débogage local.

`--container-host` *TEXT*

Hôte du conteneur Lambda émulé localement. La valeur par défaut est `localhost`. Si vous voulez exécuter la CLI AWS SAM dans un conteneur Docker sous macOS, vous pouvez spécifier `host.docker.internal`. Si vous souhaitez exécuter le conteneur sur un autre hôte AWS SAMCLI, vous pouvez spécifier l'adresse IP de l'hôte distant.

`--container-host-interface` *TEXT*

L'adresse IP de l'interface de réseau hôte à laquelle les ports de conteneur doivent se relier. La valeur par défaut est `127.0.0.1`. Utiliser `0.0.0.0` pour se lier à toutes les interfaces.

`--debug`

Active la journalisation de débogage pour imprimer le message de débogage généré par la CLI AWS SAM et pour afficher les horodatages.

`--debug-args` *TEXT*

Arguments facultatifs à transmettre au débogueur.

`--debug-function`

Facultatif. Spécifie la fonction Lambda pour appliquer les options de débogage lorsque `--warm-containers`est spécifié. Ce paramètre s'applique à `--debug-port`, `--debugger-path` et `--debug-args`.

`--debug-port, -d` *TEXT*

Lorsque spécifié, démarre le conteneur de fonction Lambda en mode débogage et expose ce port sur l'hôte local.

`--debugger-path` *TEXT*

Le chemin d'accès de l'hôte vers un débogueur qui sera monté dans le conteneur Lambda.

`--docker-network` *TEXT*

Nom ou identifiant d'un réseau Docker existant auquel les conteneurs Docker Lambda devraient se connecter, avec le réseau de pont par défaut. Si cela n'est pas spécifié, les conteneurs Lambda se connectent uniquement au réseau Docker de pont par défaut.

`--docker-volume-basedir, -v` *TEXT*

Emplacement du répertoire de base dans lequel se trouve le AWS SAM fichier. Si Docker est exécuté sur une machine distante, vous devez monter le chemin où se trouve le AWS SAM fichier sur la machine Docker et modifier cette valeur pour qu'elle corresponde à la machine distante.

`--env-vars, -n` *PATH*

Le fichier JSON contenant les valeurs des variables d'environnement de la fonction Lambda.

`--force-image-build`

Spécifie s'il AWS SAM CLI faut reconstruire l'image utilisée pour appeler des fonctions avec des couches.

`--help`

Affiche ce message, puis se ferme.

`--hook-name` *TEXT*

Nom du hook utilisé pour étendre la fonctionnalité de la CLI AWS SAM.

Valeurs acceptées : `terraform`.

`--host` *TEXT*

Le nom d'hôte local ou adresse IP à relier (par défaut : '127.0.0.1').

`--invoke-image` *TEXT*

L'URI de l'image de conteneur que vous souhaitez utiliser pour les fonctions Lambda. Par défaut, AWS SAM extrait l'image du conteneur depuis Amazon ECR Public. Utilisez cette option pour extraire l'image à partir d'un autre emplacement.

Vous pouvez spécifier cette option plusieurs fois. Chaque instance de cette option peut prendre une chaîne ou une paire valeur clé. Si vous spécifiez une chaîne, il s'agit de l'URI de l'image de conteneur à utiliser pour toutes les fonctions de l'application. Par exemple, `sam local start-api --invoke-image public.ecr.aws/sam/emu-python3.8`. Si vous spécifiez une paire valeur clé, la clé est le nom de la ressource et la valeur est l'URI de l'image de conteneur à utiliser pour cette ressource. Par exemple `sam local start-api --invoke-image public.ecr.aws/sam/emu-python3.8 --invoke-image Function1=amazon/aws-sam-cli-emulation-image-python3.8` . Avec les paires valeur clé, vous pouvez spécifier différentes images de conteneur pour différentes ressources.

`--layer-cache-basedir` *DIRECTORY*

Spécifie l'emplacement basedir dans lequel les couches utilisées par le modèle sont téléchargées.

`--log-file, -l` *TEXT*

Le fichier journal vers lequel envoyer les journaux d'exécution.

`--no-memory-limit`

Supprime la limite de mémoire dans le conteneur lors de l'appel local, même lorsque la mémoire est configurée dans le AWS SAM modèle.

`--parameter-overrides`

Chaîne contenant des remplacements de CloudFormation paramètres codés sous forme de paires clé-valeur. Utilisez le même format que le AWS Command Line Interface (AWS CLI). Le AWS SAMCLI format est basé sur des mots clés et valeurs explicites, chaque remplacement étant séparé par un espace. Voici deux exemples :

- `--parameter-overrides ParameterKey=hello,ParameterValue=world`
- `--parameter-overrides ParameterKey=hello,ParameterValue=world ParameterKey=example1,ParameterValue=example2 ParameterKey=apple,ParameterValue=banana`

`--port, -p` *INTEGER*

Le numéro de port local à écouter (par défaut : '3000').

`--profile` *TEXT*

Le profil spécifique de votre fichier d'informations d'identification qui obtient les AWS informations d'identification.

`--region` *TEXT*

La AWS région dans laquelle le déploiement doit être effectué. Par exemple, us-east-1.

`--save-params`

Enregistrez les paramètres que vous fournissez sur la ligne de commande dans le fichier AWS SAM de configuration.

`--shutdown`

Imite un événement d'arrêt une fois l'appel terminé, afin de tester la gestion des extensions du comportement d'arrêt.

`--skip-prepare-infra`

Ignore l'étape de préparation si aucune modification d'infrastructure n'a été apportée. À utiliser avec l'option `--hook-name`.

`--skip-pull-image`

Spécifie si l'interface de ligne de commande (CLI) doit ignorer la dernière image Docker pour l'exécution Lambda.

`--ssl-cert-file` *PATH*

Chemin d'accès au fichier de certificat SSL (par défaut : aucun). Lorsque vous utilisez cette option, elle doit également être utilisée. `--ssl-key-file`

`--ssl-key-file` *PATH*

Chemin d'accès au fichier clé SSL (par défaut : aucun). Lorsque vous utilisez cette option, elle doit également être utilisée. `--ssl-cert-file`

`--static-dir, -s` *TEXT*

Tous les fichiers d'actifs statiques (par exempleCSS/JavaScript/HTML) situés dans ce répertoire sont présentés à l'adresse`/`.

`--template, -t` *PATH*

Le fichier AWS SAM modèle.

> **Note**
> Si vous spécifiez cette option, AWS SAM charge uniquement le modèle et les ressources locales vers lesquels il pointe.

`--terraform-plan-file`

Le chemin relatif ou absolu vers votre fichier de plan Terraform local lorsque vous utilisez la CLI AWS SAM avec Terraform Cloud. Cette option nécessite que `--hook-name` soit défini sur `terraform`.

`--warm-containers` *`[EAGER | LAZY]`*

Facultatif. Spécifie comment la CLI AWS SAM gère les conteneurs pour chaque fonction.

Deux options sont disponibles :

`EAGER` : Les conteneurs pour toutes les fonctions sont chargés au démarrage et persistent entre les appels.

`LAZY` : Les conteneurs ne sont chargés que lorsque chaque fonction est appelée pour la première fois. Ces conteneurs persistent pour des appels supplémentaires.

Exemple

L'exemple suivant démarre un serveur local, ce qui vous permet de tester votre application via une API. Pour que cette commande fonctionne, l'application doit être installée et Docker doit être en cours d'exécution.

```
$ sam local start-api --port 3000
```

sam local start-lambda

Cette page fournit des informations de référence pour la AWS Serverless Application Model `sam local start-lambda` sous-commande Command Line Interface (AWS SAMCLI).

- Pour une introduction à la AWS SAMCLI, voir Qu'est-ce que c'est AWS SAMCLI ?
- Pour obtenir de la documentation sur l'utilisation de la AWS SAMCLI `sam local start-lambda` sous-commande, consultezPrésentation des tests avec sam local start-lambda.

La `sam local start-lambda` sous-commande démarre un point de terminaison local à AWS Lambdaémuler.

Usage

```
$ sam local start-lambda <options>
```

Options

`--add-host` *`LIST`*

Transmet un mappage de nom d'hôte à adresse IP au fichier hôte du conteneur Docker. Ce paramètre peut être transmis plusieurs fois.

Example

Exemple : `--add-host` *`example.com:127.0.0.1`*

`--beta-features | --no-beta-features`

Autorisez ou refusez les fonctionnalités bêta.

`--config-env` *`TEXT`*

Le nom d'environnement spécifiant les valeurs de paramètre par défaut dans le fichier de configuration à utiliser. La valeur par défaut est « par défaut ». Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--config-file` *`PATH`*

Le chemin d'accès et le nom du fichier de configuration contenant les valeurs de paramètres par défaut à utiliser. La valeur par défaut est « samconfig.toml » dans la racine du répertoire du projet. Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--container-env-vars`

Facultatif. Transmettez les variables d'environnement au conteneur d'images lors du débogage local.

`--container-host` *`TEXT`*

Hôte du conteneur Lambda émulé localement. La valeur par défaut est `localhost`. Si vous voulez exécuter la CLI AWS SAM dans un conteneur Docker sous macOS, vous pouvez spécifier

`host.docker.internal`. Si vous souhaitez exécuter le conteneur sur un autre hôte AWS SAMCLI, vous pouvez spécifier l'adresse IP de l'hôte distant.

`--container-host-interface` *TEXT*

L'adresse IP de l'interface de réseau hôte à laquelle les ports de conteneur doivent se relier. La valeur par défaut est `127.0.0.1`. Utiliser `0.0.0.0` pour se lier à toutes les interfaces.

`--debug`

Active la journalisation de débogage pour imprimer le message de débogage généré par la CLI AWS SAM et pour afficher les horodatages.

`--debug-args` *TEXT*

Arguments facultatifs à transmettre au débogueur.

`--debug-function`

Facultatif. Spécifie la fonction Lambda pour appliquer les options de débogage lorsque `--warm-containers`est spécifié. Ce paramètre s'applique à `--debug-port`, `--debugger-path` et `--debug-args`.

`--debug-port, -d` *TEXT*

Si spécifié, il démarre le conteneur de fonction Lambda en mode débogage et expose ce port sur l'hôte local.

`--debugger-path` *TEXT*

Le chemin d'accès de l'hôte vers un débogueur à monter dans le conteneur Lambda.

`--docker-network` *TEXT*

Le nom ou l'identifiant d'un réseau Docker existant auquel les conteneurs Docker Lambda doivent se connecter, avec le réseau de pont par défaut. Si ceci est spécifié, les conteneurs Lambda se connectent uniquement au réseau Docker de pont par défaut.

`--docker-volume-basedir, -v` *TEXT*

Emplacement du répertoire de base dans lequel se trouve le AWS SAM fichier. Si Docker est exécuté sur une machine distante, vous devez monter le chemin où se trouve le AWS SAM fichier sur la machine Docker et modifier cette valeur pour qu'elle corresponde à la machine distante.

`--env-vars, -n` *PATH*

Le fichier JSON contenant les valeurs des variables d'environnement de la fonction Lambda.

`--force-image-build`

Spécifiez s'il CLI faut reconstruire l'image utilisée pour appeler des fonctions avec des couches.

`--help`

Affiche ce message, puis se ferme.

`--hook-name` *TEXT*

Nom du hook utilisé pour étendre la fonctionnalité de la CLI AWS SAM.

Valeurs acceptées : `terraform`.

`--host` *TEXT*

Le nom d'hôte local ou adresse IP à relier (par défaut : '127.0.0.1').

`--invoke-image` *TEXT*

L'URI de l'image de conteneur que vous souhaitez utiliser pour l'appel de la fonction locale. Par défaut, AWS SAM extrait l'image du conteneur depuis Amazon ECR Public. Utilisez cette option pour extraire l'image à partir d'un autre emplacement.

Par exemple, `sam local start-lambda MyFunction --invoke-image amazon/aws-sam-cli-emulation-image-python3.8`.

`--layer-cache-basedir` *DIRECTORY*

Spécifie l'emplacement basedir dans lequel les couches utilisées par le modèle sont téléchargées.

`--log-file, -l` *TEXT*

Le fichier journal vers lequel envoyer les journaux d'exécution.

`--no-memory-limit`

Supprime la limite de mémoire dans le conteneur lors de l'appel local, même lorsque la mémoire est configurée dans le AWS SAM modèle.

`--parameter-overrides`

Chaîne contenant des remplacements de CloudFormation paramètres codés sous forme de paires clé-valeur. Utilisez le même format que le AWS Command Line Interface (AWS CLI). Le AWS SAMCLI format est basé sur des mots clés et valeurs explicites, chaque remplacement étant séparé par un espace. Voici deux exemples :

- `--parameter-overrides ParameterKey=hello,ParameterValue=world`

- `--parameter-overrides ParameterKey=hello,ParameterValue=world ParameterKey=example1,ParameterValue=example2 ParameterKey=apple,ParameterValue=banana`

`--port, -p` *`INTEGER`*

Numéro de port local à écouter (par défaut : '3001').

`--profile` *`TEXT`*

Le profil spécifique de votre fichier d'informations d'identification qui obtient les AWS informations d'identification.

`--region` *`TEXT`*

La AWS région dans laquelle le déploiement doit être effectué. Par exemple, us-east-1.

`--save-params`

Enregistrez les paramètres que vous fournissez sur la ligne de commande dans le fichier AWS SAM de configuration.

`--shutdown`

Imite un événement d'arrêt une fois l'appel terminé, afin de tester la gestion des extensions du comportement d'arrêt.

`--skip-prepare-infra`

Ignore l'étape de préparation si aucune modification d'infrastructure n'a été apportée. À utiliser avec l'option `--hook-name`.

`--skip-pull-image`

Spécifie s'il CLI faut ignorer l'extraction de la dernière image Docker vers le bas pour le runtime Lambda.

`--template, -t` *`PATH`*

Le fichier AWS SAM modèle.

> **Note**
>
> Si vous spécifiez cette option, AWS SAM charge uniquement le modèle et les ressources locales vers lesquels il pointe. Cette option n'est pas compatible avec `--hook-name`.

`--terraform-plan-file`

Le chemin relatif ou absolu vers votre fichier de plan Terraform local lorsque vous utilisez la CLI AWS SAM avec Terraform Cloud. Cette option nécessite que `--hook-name` soit défini sur `terraform`.

`--warm-containers` *`[EAGER | LAZY]`*

Facultatif. Spécifie comment la CLI AWS SAM gère les conteneurs pour chaque fonction.

Deux options sont disponibles :

- `EAGER` : Les conteneurs pour toutes les fonctions sont chargés au démarrage et persistent entre les appels.
- `LAZY` : Les conteneurs ne sont chargés que lorsque chaque fonction est appelée pour la première fois. Ces conteneurs persistent pour des appels supplémentaires.

Exemple

Pour un exemple détaillé et une présentation détaillée de l'utilisation de la `sam local start-lambda` sous-commande, reportez-vous à. Présentation des tests avec sam local start-lambda

sam logs

Cette page fournit des informations de référence pour la AWS Serverless Application Model commande Command Line Interface (AWS SAMCLI)`sam logs`.

Pour une introduction à la AWS SAMCLI, voir Qu'est-ce que c'est AWS SAMCLI ?

La `sam logs` commande récupère les journaux générés par vos AWS Lambda fonctions.

Usage

```
$ sam logs <options>
```

Options

`--config-env` *`TEXT`*

Le nom d'environnement spécifiant les valeurs de paramètre par défaut dans le fichier de configuration à utiliser. La valeur par défaut est « par défaut ». Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--config-file PATH`

Le chemin d'accès et le nom du fichier de configuration contenant les valeurs de paramètres par défaut à utiliser. La valeur par défaut est « samconfig.toml » dans la racine du répertoire du projet. Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--cw-log-group LIST`

Inclut les CloudWatch journaux des groupes de journaux que vous spécifiez. Si vous spécifiez cette option avec`name`, AWS SAM inclut les journaux des groupes de journaux spécifiés en plus des journaux des ressources nommées.

`--debug`

Active la journalisation de débogage pour imprimer le message de débogage généré par la CLI AWS SAM et pour afficher les horodatages.

`---end-time, e TEXT`

Récupère les journaux jusqu'à ce moment. L'heure peut être des valeurs relatives comme « il y a 5 minutes », « hier », ou un horodatage formaté comme « 2018-01-01 10:10:10 ».

`--filter TEXT`

Vous permet de spécifier une expression pour rechercher rapidement des journaux qui correspondent à des termes, des phrases ou des valeurs dans les journaux d'événements. Il peut s'agir d'un simple mot clé (par exemple, « erreur ») ou d'un modèle pris en charge par Amazon CloudWatch Logs. Pour connaître la syntaxe, consultez la documentation Amazon CloudWatch Logs.

`--help`

Affiche ce message, puis se ferme.

`--include-traces`

Inclut des traces X-Ray dans la sortie du journal.

`--name, -n TEXT`

Le nom de la ressource pour laquelle vous souhaitez récupérer des journaux. Si cette ressource fait partie d'une CloudFormation pile, il peut s'agir de l'ID logique de la ressource fonctionnelle dans le AWS SAM modèle CloudFormation/. Plusieurs noms peuvent être fournis en répétant à nouveau le paramètre. Si la ressource se trouve dans une pile imbriquée, le nom peut être précédé du nom de la pile imbriquée pour extraire les journaux de cette ressource (/).

NestedStackLogicalId ResourceLogicalId Si le nom de la ressource n'est pas fourni, la pile donnée sera analysée et les informations de journal seront extraites pour toutes les ressources prises en charge. Si vous ne spécifiez pas cette option, AWS SAM extrait les journaux de toutes les ressources de la pile que vous spécifiez. Les types de ressources suivants sont pris en charge :

- `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*

Spécifie le format de sortie pour les journaux. Pour imprimer des journaux formatés, spécifiez `text`. Pour imprimer les journaux au format JSON, spécifiez `json`.

`--profile` *TEXT*

Le profil spécifique de votre fichier d'informations d'identification qui obtient les AWS informations d'identification.

`--region` *TEXT*

AWS Région dans laquelle le déploiement doit être effectué. Par exemple, us-east-1.

`--save-params`

Enregistrez les paramètres que vous fournissez sur la ligne de commande dans le fichier AWS SAM de configuration.

`--stack-name` *TEXT*

Nom de la CloudFormation pile dont fait partie la ressource.

`--start-time, -s` *TEXT*

Récupère les journaux à partir de ce moment. L'heure peut être des valeurs relatives comme « il y a 5 minutes », « hier », ou un horodatage formaté comme « 2018-01-01 10:10:10 ». La valeur par défaut est « il y a 10 minutes ».

`--tail, -t`

Suit la sortie du journal. Cela ignore l'argument de fin d'heure et continue de récupérer les journaux à mesure qu'ils deviennent disponibles.

Exemples

Lorsque vos fonctions font partie d'une CloudFormation pile, vous pouvez récupérer les journaux en utilisant l'identifiant logique de la fonction lorsque vous spécifiez le nom de la pile.

```
$ sam logs -n HelloWorldFunction --stack-name myStack
```

Consultez les journaux pour une plage de temps spécifique à l'aide des options -s (--start-time) et -e (--end-time).

```
$ sam logs -n HelloWorldFunction --stack-name myStack -s '10min ago' -e '2min ago'
```

Vous pouvez également ajouter la `--tail` possibilité d'attendre les nouveaux journaux et de les voir à leur arrivée.

```
$ sam logs -n HelloWorldFunction --stack-name myStack --tail
```

Utilisez `--filter` cette option pour trouver rapidement les journaux correspondant aux termes, expressions ou valeurs de vos événements de journal.

```
$ sam logs -n HelloWorldFunction --stack-name myStack --filter "error"
```

Consultez les journaux d'une ressource dans une pile enfant.

```
$ sam logs --stack-name myStack -n childStack/HelloWorldFunction
```

Journaux de suivi de toutes les ressources prises en charge dans votre application.

```
$ sam logs --stack-name sam-app --tail
```

Récupérez les journaux d'une fonction Lambda et d'une API API Gateway spécifiques dans votre application.

```
$ sam logs --stack-name sam-app --name HelloWorldFunction --name HelloWorldRestApi
```

Récupérez les journaux de toutes les ressources prises en charge dans votre application, ainsi que des groupes de journaux spécifiés.

```
$ sam logs --cw-log-group /aws/lambda/myfunction-123 --cw-log-group /aws/lambda/
myfunction-456
```

sam package

L'interface de ligne de AWS Serverless Application Model commande (AWS SAM CLI) empaquette une AWS SAM application.

Cette commande crée un `.zip` fichier contenant votre code et vos dépendances, et télécharge le fichier sur Amazon Simple Storage Service (Amazon S3). AWS SAM active le chiffrement de tous les fichiers stockés dans Amazon S3. Il renvoie ensuite une copie de votre AWS SAM modèle, en remplaçant les références aux artefacts locaux par l'emplacement Amazon S3 où la commande a chargé les artefacts.

Par défaut, lorsque vous utilisez cette commande, la CLI AWS SAM suppose que le répertoire de travail actuel est le répertoire racine du projet. Le AWS SAMCLI premier essaie de localiser un fichier modèle créé à l'aide de la sam build commande, situé dans le `.aws-sam` sous-dossier et nommé`template.yaml`. Ensuite, la CLI AWS SAM tente de localiser un fichier modèle nommé `template.yaml` ou `template.yml` dans le répertoire de travail actuel. Si vous spécifiez l'`--template`option, AWS SAMCLI le comportement par défaut est remplacé et empaquetera uniquement ce AWS SAM modèle et les ressources locales vers lesquelles il pointe.

> **Note**
>
> sam deploy exécute maintenant implicitement la fonctionnalité de `sam package`. Vous pouvez utiliser la commande sam deploy directement pour empaqueter et déployer l'application.

Usage

```
$ sam package <arguments> <options>
```

Arguments

ID de ressource

ID de la fonction Lambda à empaqueter.

Cet argument est facultatif. Si votre application contient une seule fonction Lambda, la AWS SAM CLI l'empaquetera. Si votre application contient plusieurs fonctions, indiquez l'ID de la fonction pour empaqueter une seule fonction.

Valeurs valides : ID logique ou ARN de la ressource.

Options

`--config-env` *`TEXT`*

Le nom d'environnement spécifiant les valeurs de paramètre par défaut dans le fichier de configuration à utiliser. La valeur par défaut est « par défaut ». Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--config-file` *`PATH`*

Le chemin d'accès et le nom du fichier de configuration contenant les valeurs de paramètres par défaut à utiliser. La valeur par défaut est « samconfig.toml » dans la racine du répertoire du projet. Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--debug`

Active la journalisation de débogage pour imprimer le message de débogage généré par la CLI AWS SAM et pour afficher les horodatages.

`--force-upload`

Remplacer les fichiers existants dans le compartiment Amazon S3. Spécifiez cet indicateur pour télécharger des artefacts, même s'ils correspondent à des artefacts existants dans le compartiment Amazon S3.

`--help`

Affiche ce message, puis se ferme.

`--image-repository` *TEXT*

L'URI du référentiel Amazon Elastic Container Registry (Amazon ECR) vers lequel cette commande télécharge l'image de la fonction. Requis pour les fonctions déclarées avec le type de package `Image`.

`--kms-key-id` *TEXT*

L'ID d'une clé AWS Key Management Service (AWS KMS) utilisée pour chiffrer les artefacts inactifs dans le compartiment Amazon S3. Si cette option n'est pas spécifiée, AWS SAM utilise les clés de chiffrement gérées par Amazon S3.

`--metadata`

(Facultatif) Une carte de métadonnées à attacher à tous les artefacts référencés dans le modèle.

`--no-progressbar`

N'affichez pas de barre de progression lors du téléchargement d'artefacts sur Amazon S3.

`--output-template-file` *PATH*

Chemin d'accès au fichier dans lequel la commande écrit le modèle empaqueté. Si vous ne spécifiez pas de chemin d'accès, la commande écrit le modèle dans la sortie standard.

`--profile` *TEXT*

Le profil spécifique de votre fichier d'informations d'identification qui obtient les AWS informations d'identification.

`--region` *TEXT*

La AWS région dans laquelle le déploiement doit être effectué. Par exemple, us-east-1.

`--resolve-s3`

Créez automatiquement un compartiment Amazon S3 à utiliser pour l'empaquetage. Si vous précisez les deux options `--s3-bucket` et `--resolve-s3`, alors une erreur se produira.

`--s3-bucket` *TEXT*

Le nom du compartiment Amazon S3 dans lequel cette commande télécharge votre artefact. Si votre artefact est supérieur à 51 200 octets, l'option `--s3-bucket` ou l'`--resolve-s3`option est requise. Si vous précisez les deux options `--s3-bucket` et `--resolve-s3`, alors une erreur se produira.

`--s3-prefix TEXT`

Le préfixe ajouté aux noms des artefacts téléchargés vers le compartiment Amazon S3. Le nom du préfixe est un nom de chemin d'accès (nom de dossier) pour le compartiment Amazon S3. Cela s'applique uniquement aux fonctions déclarées avec le type de package `Zip`.

`--save-params`

Enregistrez les paramètres que vous fournissez sur la ligne de commande dans le fichier AWS SAM de configuration.

`--signing-profiles LIST`

(Facultatif) La liste des profils de signature pour signer les packages de déploiement. Ce paramètre prend une liste de paires valeur clé, où la clé est le nom de la fonction ou de la couche à signer et la valeur est le profil de signature, avec un propriétaire de profil facultatif délimité par `:`. Par exemple, `FunctionNameToSign=SigningProfileName1 LayerNameToSign=SigningProfileName2:SigningProfileOwner`.

`--template-file, --template, -t PATH`

Le chemin et le nom du fichier où se trouve votre AWS SAM modèle.

> **Note**
>
> Si vous spécifiez cette option, AWS SAM empaquette uniquement le modèle et les ressources locales vers lesquels il pointe.

`--use-json`

Sortie JSON pour le CloudFormation modèle. Par défaut, YAML est utilisé.

Exemple

L'exemple suivant crée et empaquette des artefacts pour une fonction Lambda et des CodeDeploy applications. Les artefacts sont chargés dans un compartiment Amazon S3. La sortie de la commande est un nouveau fichier nommé `package.yml`.

```
$ sam package \
```

```
  --template-file template.yml \
  --output-template-file package.yml \
  --s3-bucket amzn-s3-demo-bucket
```

sam pipeline bootstrap

Cette page fournit des informations de référence pour la AWS Serverless Application Model `sam local pipeline bootstrap` sous-commande Command Line Interface (AWS SAMCLI).

Pour une introduction à la AWS SAMCLI, voir Qu'est-ce que c'est AWS SAMCLI ?

La `sam pipeline bootstrap` sous-commande génère les ressources AWS d'infrastructure requises pour se connecter à votre système CI/CD. Cette étape doit être exécutée pour chaque étape de déploiement du pipeline, avant d'exécuter la commande sam pipeline init.

Cette sous-commande configure les ressources AWS d'infrastructure suivantes :

- Option de configuration des autorisations du pipeline via :
  - Un utilisateur IAM du pipeline avec un identifiant de clé d'accès et des informations d'identification d'accès par clé secrète à partager avec le CI/CD système.

    > **Note**
    >
    > Nous vous recommandons d'effectuer une rotation régulière des clés d'accès. Pour plus d'informations, consultez Rotation régulière des clés d'accès pour les cas d'utilisation nécessitant des informations d'identification dans le Guide de l'utilisateur IAM.

  - CI/CD Plateformes prises en charge via OIDC. Pour obtenir une présentation de l'utilisation d'OIDC avec un pipeline AWS SAM , consultez Comment utiliser l'authentification OIDC avec les pipelines AWS SAM.
- Rôle IAM d' CloudFormation exécution assumé par CloudFormation pour déployer l' AWS SAM application.
- Un compartiment Amazon S3 pour contenir les AWS SAM artefacts.
- Le cas échéant, un référentiel d'images Amazon ECR pour contenir des packages de déploiement Lambda de conteneur d'images (si vous disposez d'une ressource avec le type de package `Image`).

Usage

```
$ sam pipeline bootstrap <options>
```

Options

`--bitbucket-repo-uuid` *TEXT*

L'UUID du référentiel Bitbucket. Cette option est spécifique à l'utilisation d'OIDC avec Bitbucket pour les autorisations.

> **Note**
>
> Cette valeur se trouve à l'adresse https://bitbucket.org/*workspace*/*repository*/admin/addon/admin/pipelines/openid-connect

`--bucket` *TEXT*

L'ARN du compartiment Amazon S3 qui contient les AWS SAM artefacts.

`--cicd-provider` *TEXT*

La CI/CD plate-forme pour le AWS SAM pipeline.

`--cloudformation-execution-role` *TEXT*

L'ARN du rôle IAM à assumer CloudFormation lors du déploiement de la pile de l'application. Fournissez-le uniquement si vous voulez utiliser votre propre rôle. Sinon, la commande crée un nouveau rôle.

`--config-env` *TEXT*

Le nom d'environnement spécifiant les valeurs de paramètre par défaut dans le fichier de configuration à utiliser. La valeur par défaut est **`default`**. Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--config-file` *PATH*

Le chemin d'accès et le nom du fichier de configuration contenant les valeurs de paramètres par défaut à utiliser. La valeur par défaut est `samconfig.toml` à la racine du répertoire du projet. Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--confirm-changeset | --no-confirm-changeset`

Invitation à confirmer le déploiement de vos ressources.

`--create-image-repository | --no-create-image-repository`

Indiquez si vous souhaitez créer un référentiel d'images Amazon ECR si aucun n'est fourni. Le référentiel Amazon ECR contient les images de conteneur des fonctions Lambda ou des couches présentant un type de package `Image`. La valeur par défaut est `--no-create-image-repository`.

`--debug`

Active la journalisation de débogage, imprime les messages de débogage générés par la CLI AWS SAM et affiche les horodatages.

`--deployment-branch` *TEXT*

Nom de la branche à partir de laquelle les déploiements seront effectués. Cette option est spécifique à l'utilisation d' GitHub Actions OIDC pour les autorisations.

`--github-org` *TEXT*

L' GitHub organisation à laquelle appartient le référentiel. Si aucune organisation n'existe, saisissez le nom d'utilisateur du propriétaire du référentiel. Cette option est spécifique à l'utilisation d' GitHubActions OIDC pour les autorisations.

`--github-repo` *TEXT*

Nom du GitHub référentiel à partir duquel les déploiements seront effectués. Cette option est spécifique à l'utilisation d' GitHub Actions OIDC pour les autorisations.

`--gitlab-group` *TEXT*

Le GitLab groupe auquel appartient le référentiel. Cette option est spécifique à l'utilisation de l' GitLab OIDC pour les autorisations.

`--gitlab-project` *TEXT*

Le nom GitLab du projet. Cette option est spécifique à l'utilisation de l' GitLab OIDC pour les autorisations.

`--help, -h`

Affiche ce message, puis se ferme.

`--image-repository` *TEXT*

L'ARN d'un référentiel d'images Amazon ECR contenant les images de conteneur des fonctions Lambda ou des couches avec un type de package `Image`. S'il est fourni, l'option `--create-image-repository` est ignorée. S'il n'est pas fourni et que `--create-image-repository` est spécifié, la commande en crée un.

`--interactive | --no-interactive`

Désactivez l'invite interactive pour les paramètres d'amorçage et attendez si des paramètres requis sont manquants. La valeur par défaut est `--interactive`. Avec cette commande, `--stage` est le seul paramètre obligatoire.

> **Note**
>
> S'il `--no-interactive` est spécifié en même temps que`--use-oidc-provider`, tous les paramètres requis pour votre fournisseur OIDC doivent être inclus.

`--oidc-client-id` *TEXT*

L'ID client configuré pour être utilisé avec votre fournisseur OIDC.

`--oidc-provider` *[github-actions | gitlab | bitbucket-pipelines]*

Nom du CI/CD fournisseur qui sera utilisé pour les autorisations OIDC. GitLab, GitHub, et Bitbucket sont pris en charge.

`--oidc-provider-url` *TEXT*

URL du fournisseur OIDC. La valeur doit commencer par **`https://`**.

`--permissions-provider` *[oidc | iam]*

Choisissez un fournisseur d'autorisations pour assumer le rôle d'exécution du pipeline. La valeur par défaut est **`iam`**.

`--pipeline-execution-role` *TEXT*

L'ARN du rôle IAM devant être assumé par l'utilisateur de pipeline pour fonctionner sur cette étape. Fournissez-le uniquement si vous voulez utiliser votre propre rôle. S'il n'est pas fourni, cette commande crée un nouveau rôle.

`--pipeline-user` *TEXT*

Nom de ressource Amazon (ARN) de l'utilisateur IAM dont l'ID de clé d'accès et la clé d'accès secrète sont partagés avec le CI/CD système. Il est utilisé pour accorder à cet utilisateur IAM l'autorisation d'accéder au AWS compte correspondant. S'il n'est pas fourni, la commande crée un utilisateur IAM avec l'identifiant de la clé d'accès et les informations d'identification de clé d'accès secrète.

`--profile` *TEXT*

Le profil spécifique de votre fichier d'informations d'identification qui obtient les AWS informations d'identification.

`--region` *TEXT*

La AWS région dans laquelle le déploiement doit être effectué. Par exemple, `us-east-1`.

`--save-params`

Enregistrez les paramètres que vous fournissez sur la ligne de commande dans le fichier AWS SAM de configuration.

`--stage` *TEXT*

Le nom de l'étape de déploiement correspondante. Il est utilisé comme suffixe pour les ressources AWS d'infrastructure créées.

Résolution des problèmes

Error: Missing required parameter (Erreur : paramètre requis manquant)

Quand `--no-interactive` est spécifié avec `--use-oidc-provider` et qu'aucun des paramètres requis n'est fourni, ce message d'erreur s'affiche avec une description des paramètres manquants.

Exemple

L'exemple suivant crée les AWS ressources nécessaires à la création de votre CI/CD système, active la journalisation du débogage et imprime les messages de débogage générés par AWS SAMCLI : utilise un événement généré pour les tests locaux en utilisant un `s3.json` événement pour appeler une fonction Lambda localement

```
$ sam pipeline bootstrap --debug
```

sam pipeline init

Cette page fournit des informations de référence pour la AWS Serverless Application Model `sam pipeline init` sous-commande Command Line Interface (AWS SAMCLI).

Pour une introduction à la AWS SAMCLI, voir Qu'est-ce que c'est AWS SAMCLI ?

La `sam pipeline init` sous-commande génère un fichier de configuration de pipeline que votre CI/CD système peut utiliser pour déployer des applications sans serveur à l'aide de. AWS SAM

Avant d'utiliser sam pipeline init, vous devez amorcer les ressources nécessaires pour chaque étape du pipeline. Pour ce faire, exécutez sam pipeline init --bootstrap pour être guidé tout au long du processus de génération de fichiers d'installation et de configuration ou référez-vous aux ressources que vous avez précédemment créées à l'aide de la commande sam pipeline bootstrap.

Usage

```
$ sam pipeline init <options>
```

Options

`--bootstrap`

Activez le mode interactif qui guide l'utilisateur tout au long de la création des ressources AWS d'infrastructure nécessaires.

`--config-env` *TEXT*

Le nom d'environnement spécifiant les valeurs de paramètre par défaut dans le fichier de configuration à utiliser. La valeur par défaut est `default`. Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--config-file` *TEXT*

Le chemin d'accès et le nom du fichier de configuration contenant les valeurs de paramètres par défaut à utiliser. La valeur par défaut est `samconfig.toml` dans le répertoire racine du projet. Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--debug`

Active la journalisation de débogage pour imprimer les messages de débogage générés par la CLI AWS SAM et pour afficher les horodatages.

`--help, -h`

Affiche ce message, puis se ferme.

`--save-params`

Enregistrez les paramètres que vous fournissez sur la ligne de commande dans le fichier AWS SAM de configuration.

Exemple

L'exemple suivant vous montre comment utiliser l'`--bootstrap`option qui vous permet de passer par un mode interactif qui vous guide tout au long de la création des ressources d' AWS infrastructure nécessaires :

```
$ sam pipeline init --bootstrap
```

sam publish

Cette page fournit des informations de référence pour la AWS Serverless Application Model commande Command Line Interface (AWS SAMCLI)`sam publish`.

Pour une introduction à la AWS SAMCLI, voir Qu'est-ce que c'est AWS SAMCLI ?

La `sam publish` commande publie une AWS SAM application dans le AWS Serverless Application Repository. Cette commande prend un AWS SAM modèle empaqueté et publie l'application dans la AWS région spécifiée.

La `sam publish` commande s'attend à ce que le AWS SAM modèle inclue une `Metadata` section contenant les métadonnées d'application requises pour la publication. Dans la section `Metadata`, les propriétés `LicenseUrl` et `ReadmeUrl` doivent faire référence aux compartiments Amazon Simple Storage Service (Amazon S3) et non pas aux fichiers locaux. Pour plus d'informations sur la `Metadata` section du AWS SAM modèle, consultezPublier votre candidature à l'aide du AWS SAMCLI.

Par défaut, `sam publish` crée l'application comme privée. Avant que d'autres comptes AWS soient autorisés à visualiser et à déployer l'application, vous devez la partager. Pour plus d'informations sur le partage de l'application, consultez Exemples de stratégies basées sur les ressources AWS Serverless Application Repository dans le Guide du développeur AWS Serverless Application Repository .

> **Note**
>
> Actuellement `sam publish` ne prend pas en charge la publication d'applications imbriquées qui sont spécifiées localement. Si votre application contient des applications imbriquées, vous devez les publier séparément dans l'application parent AWS Serverless Application Repository avant de publier.

Usage

```
$ sam publish <options>
```

Options

`--config-env` *TEXT*

Le nom d'environnement spécifiant les valeurs de paramètre par défaut dans le fichier de configuration à utiliser. La valeur par défaut est « par défaut ». Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--config-file` *PATH*

Le chemin d'accès et le nom du fichier de configuration contenant les valeurs de paramètres par défaut à utiliser. La valeur par défaut est « `samconfig.toml` » dans la racine du répertoire du projet. Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--debug`

Active la journalisation de débogage pour imprimer les messages de débogage générés par la CLI AWS SAM et pour afficher les horodatages.

`--help`

Affiche ce message, puis se ferme.

`--profile` *TEXT*

Le profil spécifique de votre fichier d'informations d'identification qui obtient les AWS informations d'identification.

`--region` *TEXT*

La AWS région dans laquelle le déploiement doit être effectué. Par exemple, us-east-1.

`--save-params`

Enregistrez les paramètres que vous fournissez sur la ligne de commande dans le fichier AWS SAM de configuration.

`--semantic-version` *TEXT*

(Facultatif) Utilisez cette option pour fournir une version sémantique de l'application qui remplace la propriété `SemanticVersion` dans la section `Metadata` du fichier modèle. Pour obtenir la spécification de gestion sémantique des versions, veuillez consulter le site web de la Gestion sémantique des versions.

`--template, -t` *PATH*

Le chemin du fichier AWS SAM modèle`[default: template.[yaml|yml]]`.

Exemples

Pour publier une application, procédez comme suit :

```
$ sam publish --template packaged.yaml --region us-east-1
```

sam remote callback

Cette page fournit des informations de référence pour la AWS Serverless Application Model commande Command Line Interface (AWS SAMCLI)`sam remote callback`.

Pour une introduction à la AWS SAMCLI, voir Qu'est-ce que c'est AWS SAMCLI ?

La `sam remote callback` commande vous permet d'envoyer des rappels à des exécutions de fonctions durables à distance exécutées dans AWS Lambda.

> **Note**
> Ces commandes nécessitent l'accès aux AWS informations d'identification.

Usage

```
$ sam remote callback <subcommand> <options>
```

Options

`--help, -h`

Affichez ce message et quittez.

Sous-commandes

`succeed`

Envoyez un rappel réussi à une exécution de fonction durable à distance. Pour de plus amples informations, veuillez consulter sam remote callback succeed.

`fail`

Envoie un échec de rappel à l'exécution d'une fonction durable à distance. Pour de plus amples informations, veuillez consulter sam remote callback fail.

`heartbeat`

Envoie un rythme cardiaque de rappel à une exécution de fonction durable à distance. Pour de plus amples informations, veuillez consulter sam remote callback heartbeat.

sam remote callback succeed

Envoyez un rappel réussi à une exécution de fonction durable à distance.

> **Note**
>
> Cette commande nécessite l'accès aux AWS informations d'identification.

Usage

```
$ sam remote callback succeed CALLBACK_ID [OPTIONS]
```

Arguments obligatoires

`CALLBACK_ID`

Identifiant unique du rappel auquel envoyer une réponse positive.

Options

`-r, --result TEXT`

Charge utile du résultat de réussite sous forme de chaîne.

`--region TEXT`

Définissez la AWS région du service (par exemple,`us-east-1`).

`--profile TEXT`

Sélectionnez un profil spécifique dans votre fichier d'informations d'identification pour obtenir des AWS informations d'identification.

`--config-env TEXT`

Nom de l'environnement spécifiant les valeurs des paramètres par défaut dans le fichier de configuration. Valeur par défaut : `default`

`--config-file TEXT`

Fichier de configuration contenant les valeurs des paramètres par défaut. Valeur par défaut : `samconfig.toml`

`--save-params`

Enregistrez les paramètres fournis via la ligne de commande dans le fichier de configuration.

`--beta-features / --no-beta-features`

Activer/désactiver les fonctionnalités bêta.

`--debug`

Activez la journalisation du débogage pour imprimer les messages de débogage générés par la AWS SAM CLI et afficher les horodatages.

`-h, --help`

Affiche ce message, puis se ferme.

Exemples

Envoyer un rappel de réussite sans résultat :

```
$ sam remote callback succeed my-callback-id
```

Envoyer un rappel de réussite avec le résultat :

```
$ sam remote callback succeed my-callback-id --result 'Task completed successfully'
```

sam remote callback fail

Envoie un échec de rappel à l'exécution d'une fonction durable à distance.

> **Note**
>
> Cette commande nécessite l'accès aux AWS informations d'identification.

Usage

```
$ sam remote callback fail CALLBACK_ID [OPTIONS]
```

Arguments obligatoires

`CALLBACK_ID`

Identifiant unique du rappel auquel envoyer une réponse d'échec.

Options

`--error-data TEXT`

Données d'erreur supplémentaires.

`--stack-trace TEXT`

Empilez les entrées de suivi. Cette option peut être spécifiée plusieurs fois.

`--error-type TEXT`

Type d'erreur.

`--error-message TEXT`

Message d'erreur détaillé.

`--region TEXT`

Définissez la AWS région du service (par exemple,`us-east-1`).

`--profile TEXT`

Sélectionnez un profil spécifique dans votre fichier d'informations d'identification pour obtenir des AWS informations d'identification.

`--config-env TEXT`

Nom de l'environnement spécifiant les valeurs des paramètres par défaut dans le fichier de configuration. Valeur par défaut : `default`

`--config-file TEXT`

Fichier de configuration contenant les valeurs des paramètres par défaut. Valeur par défaut : `samconfig.toml`

`--save-params`

Enregistrez les paramètres fournis via la ligne de commande dans le fichier de configuration.

`--beta-features / --no-beta-features`

Activer/désactiver les fonctionnalités bêta.

`--debug`

Activez la journalisation du débogage pour imprimer les messages de débogage générés par la AWS SAM CLI et afficher les horodatages.

`-h, --help`

Affiche ce message, puis se ferme.

Exemples

Envoyer un rappel en cas d'échec sans paramètres :

```
$ sam remote callback fail my-callback-id
```

Envoyer un rappel d'échec avec un message d'erreur :

```
$ sam remote callback fail my-callback-id --error-message 'Task failed'
```

Envoyer un rappel en cas d'échec avec tous les paramètres :

```
$ 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

Envoie un rythme cardiaque de rappel à une exécution de fonction durable à distance.

> **Note**
>
> Cette commande nécessite l'accès aux AWS informations d'identification.

Usage

```
$ sam remote callback heartbeat CALLBACK_ID [OPTIONS]
```

Arguments obligatoires

`CALLBACK_ID`

Identifiant unique du rappel auquel envoyer un battement de cœur.

Options

`--region TEXT`

Définissez la AWS région du service (par exemple,`us-east-1`).

`--profile TEXT`

Sélectionnez un profil spécifique dans votre fichier d'informations d'identification pour obtenir des AWS informations d'identification.

`--config-env TEXT`

Nom de l'environnement spécifiant les valeurs des paramètres par défaut dans le fichier de configuration. Valeur par défaut : `default`

`--config-file TEXT`

Fichier de configuration contenant les valeurs des paramètres par défaut. Valeur par défaut : `samconfig.toml`

`--save-params`

Enregistrez les paramètres fournis via la ligne de commande dans le fichier de configuration.

`--beta-features / --no-beta-features`

Activer/désactiver les fonctionnalités bêta.

`--debug`

Activez la journalisation du débogage pour imprimer les messages de débogage générés par la AWS SAM CLI et afficher les horodatages.

`-h, --help`

Affiche ce message, puis se ferme.

Exemples

Envoyer un rappel du rythme cardiaque :

```
$ sam remote callback heartbeat my-callback-id
```

sam remote execution

Cette page fournit des informations de référence pour la AWS Serverless Application Model commande Command Line Interface (AWS SAMCLI)`sam remote execution`.

Pour une introduction à la AWS SAMCLI, voir Qu'est-ce que c'est AWS SAMCLI ?

La `sam remote execution` commande vous permet de gérer et d'inspecter les exécutions de fonctions durables à distance exécutées dans AWS Lambda.

> **Note**
>
> Ces commandes nécessitent l'accès aux AWS informations d'identification.

Usage

```
$ sam remote execution <subcommand> <options>
```

Options

`--help, -h`

Affichez ce message et quittez.

Sous-commandes

`get`

Découvrez les détails d'une exécution durable. Pour de plus amples informations, veuillez consulter sam remote execution get.

`history`

Obtenez l'historique d'exécution d'une fonction durable. Pour de plus amples informations, veuillez consulter sam remote execution history.

`stop`

Arrêtez l'exécution d'une fonction durable. Pour de plus amples informations, veuillez consulter sam remote execution stop.

sam remote execution get

Récupérez les détails de l'exécution d'une fonction durable spécifique à partir de AWS Lambda.

L'ARN d'exécution doit être au format renvoyé par Invoke ou se trouver dans les journaux d'exécution.

> **Note**
>
> Cette commande nécessite l'accès aux AWS informations d'identification.

Usage

```
$ sam remote execution get DURABLE_EXECUTION_ARN [OPTIONS]
```

Arguments obligatoires

`DURABLE_EXECUTION_ARN`

Le nom de ressource Amazon (ARN) de la fonction durable à exécuter pour laquelle vous souhaitez obtenir des informations.

Options

`--format [summary|json]`

Format de la sortie. Valeur par défaut : `summary`

`--region TEXT`

Définissez la AWS région du service (par exemple,`us-east-1`).

`--profile TEXT`

Sélectionnez un profil spécifique dans votre fichier d'informations d'identification pour obtenir des AWS informations d'identification.

`--config-env TEXT`

Nom de l'environnement spécifiant les valeurs des paramètres par défaut dans le fichier de configuration. Valeur par défaut : `default`

`--config-file TEXT`

Fichier de configuration contenant les valeurs des paramètres par défaut. Valeur par défaut : `samconfig.toml`

`--save-params`

Enregistrez les paramètres fournis via la ligne de commande dans le fichier de configuration.

`--beta-features / --no-beta-features`

Activer/désactiver les fonctionnalités bêta.

`--debug`

Activez la journalisation du débogage pour imprimer les messages de débogage générés par la AWS SAM CLI et afficher les horodatages.

`-h, --help`

Affiche ce message, puis se ferme.

Exemples

Obtenez les détails de l'exécution avec le format de résumé par défaut :

```
$ 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'
```

Obtenez les détails de l'exécution au format JSON :

```
$ 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**
>
> Vous devez vous assurer que les caractères de contrôle de l'ARN d'exécution $ sont correctement échappés lorsque vous utilisez des commandes shell.

sam remote execution history

Récupérez l'historique d'exécution d'une fonction durable spécifique à partir de AWS Lambda.

L'ARN d'exécution doit être au format renvoyé par StartDurableExecution ou trouvé dans les journaux d'exécution.

> **Note**
>
> Cette commande nécessite l'accès aux AWS informations d'identification.

Usage

```
$ sam remote execution history DURABLE_EXECUTION_ARN [OPTIONS]
```

Arguments obligatoires

`DURABLE_EXECUTION_ARN`

Le nom de ressource Amazon (ARN) de la fonction durable dont vous souhaitez récupérer l'historique.

Options

`--format [table|json]`

Format de la sortie. Valeur par défaut : `table`

`--region TEXT`

Définissez la AWS région du service (par exemple,`us-east-1`).

`--profile TEXT`

Sélectionnez un profil spécifique dans votre fichier d'informations d'identification pour obtenir des AWS informations d'identification.

`--config-env TEXT`

Nom de l'environnement spécifiant les valeurs des paramètres par défaut dans le fichier de configuration. Valeur par défaut : `default`

`--config-file TEXT`

Fichier de configuration contenant les valeurs des paramètres par défaut. Valeur par défaut : `samconfig.toml`

`--save-params`

Enregistrez les paramètres fournis via la ligne de commande dans le fichier de configuration.

`--beta-features / --no-beta-features`

Activer/désactiver les fonctionnalités bêta.

`--debug`

Activez la journalisation du débogage pour imprimer les messages de débogage générés par la AWS SAM CLI et afficher les horodatages.

`-h, --help`

Affiche ce message, puis se ferme.

Exemples

Obtenez l'historique des exécutions avec le format de tableau par défaut :

```
$ sam remote execution history arn:aws:lambda:us-
east-1:123456789012:function:MyFunction:$LATEST/durable-execution/my-execution-name/my-
execution-id
```

Obtenez l'historique des exécutions au format JSON :

```
$ 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

Arrêtez l'exécution d'une fonction durable en cours d'exécution dans AWS Lambda.

L'ARN d'exécution doit être au format renvoyé StartDurableExecution ou trouvé dans les journaux d'exécution.

> **Note**
>
> Cette commande nécessite l'accès aux AWS informations d'identification.

Usage

```
$ sam remote execution stop DURABLE_EXECUTION_ARN [OPTIONS]
```

Arguments obligatoires

`DURABLE_EXECUTION_ARN`

L'Amazon Resource Name (ARN) de l'exécution de la fonction durable à arrêter.

Options

`--error-message TEXT`

Message d'erreur à associer à l'arrêt de l'exécution.

`--error-type TEXT`

Type d'erreur à associer à l'arrêt de l'exécution.

`--error-data TEXT`

Données d'erreur à associer à l'exécution arrêtée.

`--stack-trace TEXT`

Empilez les entrées de suivi. Cette option peut être spécifiée plusieurs fois.

`--region TEXT`

Définissez la AWS région du service (par exemple,`us-east-1`).

`--profile TEXT`

Sélectionnez un profil spécifique dans votre fichier d'informations d'identification pour obtenir des AWS informations d'identification.

`--config-env TEXT`

Nom de l'environnement spécifiant les valeurs des paramètres par défaut dans le fichier de configuration. Valeur par défaut : `default`

`--config-file TEXT`

Fichier de configuration contenant les valeurs des paramètres par défaut. Valeur par défaut : `samconfig.toml`

`--save-params`

Enregistrez les paramètres fournis via la ligne de commande dans le fichier de configuration.

`--beta-features / --no-beta-features`

Activer/désactiver les fonctionnalités bêta.

`--debug`

Activez la journalisation du débogage pour imprimer les messages de débogage générés par la AWS SAM CLI et afficher les horodatages.

`-h, --help`

Affiche ce message, puis se ferme.

Exemples

Arrêtez l'exécution sans détails sur l'erreur :

```
$ sam remote execution stop arn:aws:lambda:us-east-1:123456789012:function:MyFunction:
$LATEST/durable-execution/my-execution-name/my-execution-id
```

Arrêtez l'exécution avec un message d'erreur et tapez :

```
$ 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"
```

Arrêtez l'exécution en indiquant tous les détails de l'erreur et en enregistrant la 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

Cette page fournit des informations de référence pour la AWS Serverless Application Model commande Command Line Interface (AWS SAMCLI)`sam remote invoke`.

- Pour une introduction à la AWS SAMCLI, voir Qu'est-ce que c'est AWS SAMCLI ?
- Pour obtenir de la documentation sur l'utilisation de la AWS SAMCLI `sam remote invoke` commande, consultezPrésentation des tests dans le cloud avec sam remote invoke.

La `sam remote invoke` commande invoque les ressources prises en charge dans le AWS Cloud.

Usage

```
$ sam remote invoke <arguments> <options>
```

Arguments

ID de ressource

L'ID de la ressource prise en charge à invoquer.

Cet argument accepte les valeurs suivantes :

- Amazon Resource Name (ARN) — L'ARN de la ressource.

> **Tip**
>
> `sam list stack-outputs --stack-name <stack-name>`À utiliser pour obtenir l'ARN de vos ressources.

- ID logique — Identifiant logique de la ressource. Vous devez également fournir le nom de la AWS CloudFormation pile à l'aide de l'`--stack-name`option.
- Identifiant physique — Identifiant physique de la ressource. Cet ID est créé lorsque vous déployez une ressource à l'aide de CloudFormation.

> **Tip**
>
> `sam list resources --stack-name <stack-name>`À utiliser pour obtenir l'identifiant physique de vos ressources.

Lorsque vous fournissez un ARN ou un identifiant physique :

Si vous fournissez un ARN ou un identifiant physique, ne fournissez pas de nom de pile. Lorsque le nom de la pile est fourni à l'aide de l'`--stack-name`option, ou lorsque le nom de la pile est défini dans votre fichier de configuration, votre identifiant de ressource AWS SAM CLI sera automatiquement traité comme une valeur d'identifiant logique issue de la CloudFormation pile.

Lorsque vous ne fournissez pas d'ID de ressource :

Si vous ne fournissez pas d'ID de ressource, mais que vous fournissez un nom de pile avec l'`--stack-name`option, la AWS SAM CLI essaiera d'invoquer automatiquement une ressource dans votre CloudFormation pile en utilisant la logique suivante :

1. Ils AWS SAM CLI identifieront les types de ressources dans l'ordre suivant et passeront à l'étape suivante une fois que le type de ressource aura été trouvé dans votre pile :
   a. Lambda
   b. Step Functions
   c. Amazon SQS
   d. Kinesis Data Streams

2. Si le type de ressource ne contient qu'une seule ressource dans votre pile, il l' AWS SAM CLIinvoquera. Si plusieurs ressources du type de ressource existent dans votre pile, une erreur AWS SAM CLI sera renvoyée.

Voici des exemples de ce qu'ils AWS SAM CLI feront :

- Pile contenant deux fonctions Lambda et une file d'attente Amazon SQS : elle localisera AWS SAM CLI le type de ressource Lambda et renverra une erreur puisque la pile contient plusieurs fonctions Lambda.
- Pile contenant une fonction Lambda et deux applications Amazon Kinesis Data Streams : elle AWS SAM CLI localisera la fonction Lambda et l'invoquera, car la pile contient une seule ressource Lambda.
- Pile contenant une seule file d'attente Amazon SQS et deux applications Kinesis Data Streams. Elle AWS SAM CLI localisera la file d'attente Amazon SQS et l'invoquera, car la pile contient une seule file d'attente Amazon SQS.

Options

`--beta-features | --no-beta-features`

Autorisez ou refusez les fonctionnalités bêta.

`--config-env` *`TEXT`*

Spécifiez l'environnement à utiliser à partir de votre fichier de configuration CLI AWS SAM.

Par défaut : `default`

`--config-file` *`FILENAME`*

Spécifiez le nom de votre compartiment et le nom du fichier de configuration.

Pour plus d'informations sur les fichiers de configuration, consultez Configuration de la CLI AWS SAM.

Par défaut : `samconfig.toml` à la racine du répertoire de votre projet.

`--debug`

Activez la journalisation des débogues. Cela imprime les messages de débogage et les horodatages générés par la CLI AWS SAM.

`--event, -e TEXT`

L'événement à envoyer à la ressource cible.

`--event-file FILENAME`

Le chemin d'accès à un fichier contenant l'événement à envoyer à la ressource cible.

`--help, -h`

Affichez le message d'aide, puis fermez.

`--output [ text | json ]`

Exportez les résultats de votre invocation dans un format de sortie spécifique.

`json`— Les métadonnées de la demande et la réponse aux ressources sont renvoyées dans la structure JSON. La réponse contient la sortie complète du SDK.

`text`— Les métadonnées de la demande sont renvoyées sous forme de texte. La réponse à la ressource est renvoyée dans le format de sortie de la ressource invoquée.

`--parameter`

Paramètres supplémentaires Boto3 que vous pouvez transmettre à la ressource invoquée.

Amazon Kinesis Data Streams

Les paramètres supplémentaires suivants peuvent être utilisés pour placer un enregistrement dans le flux de données Kinesis :

- `ExplicitHashKey='string'`
- `PartitionKey='string'`
- `SequenceNumberForOrdering='string'`
- `StreamARN='string'`

Pour une description de chaque paramètre, consultez Kinesis.client.PUT_RECORD.

AWS Lambda

Les paramètres supplémentaires suivants peuvent être utilisés pour appeler une ressource Lambda et recevoir une réponse mise en mémoire tampon :

- `ClientContext='base64-encoded string'`
- `InvocationType='[ DryRun | Event | RequestResponse ]'`

- `LogType='[ None | Tail ]'`
- `Qualifier='string'`

Les paramètres supplémentaires suivants peuvent être utilisés pour appeler une ressource Lambda avec diffusion des réponses :

- `ClientContext='base64-encoded string'`
- `InvocationType='[ DryRun | RequestResponse ]'`
- `LogType='[ None | Tail ]'`
- `Qualifier='string'`

Pour obtenir une description de chaque paramètre, consultez les rubriques suivantes :

- Lambda avec réponse mise en mémoire tampon — Lambda.Client.Invoke
- Lambda avec diffusion de réponses — Lambda.Client.Invoke_With_Response_Stream

Amazon Simple Queue Service (Amazon SQS)

Les paramètres supplémentaires suivants peuvent être utilisés pour envoyer un message à une file d'attente Amazon SQS :

- `DelaySeconds=integer`
- `MessageAttributes='json string'`
- `MessageDeduplicationId='string'`
- `MessageGroupId='string'`
- `MessageSystemAttributes='json string'`

Pour une description de chaque paramètre, consultez SQS.Client.Send_Message.

AWS Step Functions

Les paramètres supplémentaires suivants peuvent être utilisés pour démarrer l'exécution d'une machine à états :

- `name='string'`
- `traceHeader='string'`

Pour une description de chaque paramètre, consultez SFN.Client.Start_Execution.

`--profile TEXT`

Le profil spécifique de votre fichier d'informations d'identification pour obtenir des AWS informations d'identification.

`--region` *TEXT*

Le Région AWS de la ressource. Par exemple, `us-east-1`.

`--stack-name` *TEXT*

Nom de la CloudFormation pile à laquelle appartient la ressource.

`--tenant-id` *TEXT*

L'ID de locataire pour les fonctions Lambda à locataires multiples. Utilisé pour garantir l'isolation informatique entre les différents locataires. Obligatoire lors de l'appel de fonctions configurées avec le mode d'isolation des locataires.

`--test-event-name` *NAME*

Nom de l'événement de test partageable à transmettre à votre fonction Lambda.

> **Note**
>
> Cette option ne prend en charge que les fonctions Lambda.

Exemple

L'exemple suivant invoque les ressources prises en charge dans le AWS Cloud et active la journalisation du débogage, qui imprime les messages de débogage et les horodatages générés par : AWS SAMCLI

```
$ sam remote invoke--debug
```

sam remote test-event

Cette page fournit des informations de référence pour la AWS Serverless Application Model commande Command Line Interface (AWS SAMCLI)`sam remote test-event`.

- Pour une introduction à la AWS SAMCLI, voir Qu'est-ce que c'est AWS SAMCLI ?
- Pour obtenir de la documentation sur l'utilisation de la AWS SAMCLI `sam remote test-event` commande, consultezPrésentation des tests dans le cloud avec sam remote test-event.

La `sam remote test-event` commande interagit avec les événements de test partageables dans le registre des EventBridge schémas Amazon.

Usage

```
$ sam remote test-event <options> <subcommand>
```

Options

`--help, -h`

Affichez le message d'aide, puis fermez.

Sous-commandes

`delete`

Supprimez un événement de test partageable du registre des EventBridge schémas. Pour plus d'informations de référence, consultez sam remote test-event delete.

`get`

Obtenez un événement de test partageable à partir du registre des EventBridge schémas. Pour plus d'informations de référence, consultez sam remote test-event get.

`list`

Répertoriez les événements de test partageables existants pour une AWS Lambda fonction. Pour plus d'informations de référence, consultez sam remote test-event list.

`put`

Enregistrez un événement depuis un fichier local dans le registre des EventBridge schémas. Pour plus d'informations de référence, consultez sam remote test-event put.

sam remote test-event delete

Cette page fournit des informations de référence pour la AWS Serverless Application Model `sam remote test-event delete` sous-commande Command Line Interface (AWS SAMCLI).

- Pour une introduction à la AWS SAMCLI, voir Qu'est-ce que c'est AWS SAMCLI ?
- Pour obtenir de la documentation sur l'utilisation de la AWS SAMCLI `sam remote test-event` commande, consultezPrésentation des tests dans le cloud avec sam remote test-event.

La `sam remote test-event delete` sous-commande supprime un événement de test partageable du registre des schémas Amazon EventBridge .

Usage

```
$ sam remote test-event delete <arguments> <options>
```

Arguments

ID de ressource

ID de la AWS Lambda fonction associée à l'événement de test partageable.

Si vous fournissez un identifiant logique, vous devez également fournir une valeur pour la AWS CloudFormation pile associée à la fonction Lambda à l'aide de l'`--stack-name`option.

Valeurs valides : ID logique ou ARN de la ressource.

Options

`--config-env` *TEXT*

Le nom d'environnement spécifiant les valeurs de paramètre par défaut dans le fichier de configuration à utiliser. La valeur par défaut est « par défaut ». Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--config-file` *PATH*

Le chemin d'accès et le nom du fichier de configuration contenant les valeurs de paramètres par défaut à utiliser. La valeur par défaut est « `samconfig.toml` » dans la racine du répertoire du projet. Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--help, -h`

Affichez le message d'aide, puis fermez.

`--name` *TEXT*

Nom de l'événement de test partageable à supprimer.

`--stack-name` *TEXT*

Nom de la CloudFormation pile associée à la fonction Lambda.

Cette option est obligatoire si vous fournissez l'ID logique de la fonction Lambda en tant qu'argument.

sam remote test-event get

Cette page fournit des informations de référence pour la AWS Serverless Application Model `sam remote test-event get` sous-commande Command Line Interface (AWS SAMCLI).

- Pour une introduction à la AWS SAMCLI, voir Qu'est-ce que c'est AWS SAMCLI ?
- Pour obtenir de la documentation sur l'utilisation de la AWS SAMCLI `sam remote test-event` commande, consultezPrésentation des tests dans le cloud avec sam remote test-event.

La `sam remote test-event get` sous-commande obtient un événement de test partageable depuis le registre des EventBridge schémas Amazon.

Usage

```
$ sam remote test-event get <arguments> <options>
```

Arguments

ID de ressource

L'ID de la AWS Lambda fonction associée à l'événement de test partageable à obtenir.

Si vous fournissez un identifiant logique, vous devez également fournir une valeur pour la AWS CloudFormation pile associée à la fonction Lambda à l'aide de l'`--stack-name`option.

Valeurs valides : ID logique ou ARN de la ressource.

Options

`--config-env` *TEXT*

Le nom d'environnement spécifiant les valeurs de paramètre par défaut dans le fichier de configuration à utiliser. La valeur par défaut est « par défaut ». Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--config-file PATH`

Le chemin d'accès et le nom du fichier de configuration contenant les valeurs de paramètres par défaut à utiliser. La valeur par défaut est « `samconfig.toml` » dans la racine du répertoire du projet. Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--help, -h`

Affichez le message d'aide, puis fermez.

`--name TEXT`

Nom de l'événement de test partageable à obtenir.

`--output-file FILENAME`

Chemin et nom du fichier dans lesquels enregistrer l'événement sur votre ordinateur local.

Si vous ne fournissez pas cette option, le contenu de l'événement de test partageable AWS SAM CLI sera affiché sur votre console.

`--stack-name TEXT`

Nom de la CloudFormation pile associée à la fonction Lambda.

Cette option est obligatoire si vous fournissez l'ID logique de la fonction Lambda en tant qu'argument.

sam remote test-event list

Cette page fournit des informations de référence pour la AWS Serverless Application Model `sam remote test-event list` sous-commande Command Line Interface (AWS SAMCLI).

- Pour une introduction à la AWS SAMCLI, voir Qu'est-ce que c'est AWS SAMCLI ?
- Pour obtenir de la documentation sur l'utilisation de la AWS SAMCLI `sam remote test-event` commande, consultezPrésentation des tests dans le cloud avec sam remote test-event.

La `sam remote test-event list` sous-commande répertorie les événements de test partageables existants pour une AWS Lambda fonction spécifique à partir du registre des EventBridge schémas Amazon.

Usage

```
$ sam remote test-event list <arguments> <options>
```

Arguments

ID de ressource

ID de la fonction Lambda associée aux événements de test partageables.

Si vous fournissez un identifiant logique, vous devez également fournir une valeur pour la AWS CloudFormation pile associée à la fonction Lambda à l'aide de l'`--stack-name`option.

Valeurs valides : ID logique ou ARN de la ressource.

Options

`--config-env` *TEXT*

Le nom d'environnement spécifiant les valeurs de paramètre par défaut dans le fichier de configuration à utiliser. La valeur par défaut est « par défaut ». Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--config-file` *PATH*

Le chemin d'accès et le nom du fichier de configuration contenant les valeurs de paramètres par défaut à utiliser. La valeur par défaut est « `samconfig.toml` » dans la racine du répertoire du projet. Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--help, -h`

Affichez le message d'aide, puis fermez.

`--stack-name` *TEXT*

Nom de la CloudFormation pile associée à la fonction Lambda.

Cette option est obligatoire si vous fournissez l'ID logique de la fonction Lambda en tant qu'argument.

Exemples

Pour des exemples d'utilisation de cette commande, reportez-vous àListe d'événements de test partageables.

sam remote test-event put

Cette page fournit des informations de référence pour la AWS Serverless Application Model `sam remote test-event put` sous-commande Command Line Interface (AWS SAMCLI).

- Pour une introduction à la AWS SAMCLI, voir Qu'est-ce que c'est AWS SAMCLI ?
- Pour obtenir de la documentation sur l'utilisation de la AWS SAMCLI `sam remote test-event` commande, consultezPrésentation des tests dans le cloud avec sam remote test-event.

La `sam remote test-event put` sous-commande enregistre un événement de test partageable depuis votre machine locale dans le registre des EventBridge schémas Amazon.

Usage

```
$ sam remote test-event put <arguments> <options>
```

Arguments

ID de ressource

ID de la AWS Lambda fonction associée à l'événement de test partageable.

Si vous fournissez un identifiant logique, vous devez également fournir une valeur pour la AWS CloudFormation pile associée à la fonction Lambda à l'aide de l'`--stack-name`option.

Valeurs valides : ID logique ou ARN de la ressource.

Options

`--config-env` *`TEXT`*

Le nom d’environnement spécifiant les valeurs de paramètre par défaut dans le fichier de configuration à utiliser. La valeur par défaut est « par défaut ». Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--config-file` *PATH*

Le chemin d'accès et le nom du fichier de configuration contenant les valeurs de paramètres par défaut à utiliser. La valeur par défaut est « `samconfig.toml` » dans la racine du répertoire du projet. Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--file` *FILENAME*

Chemin et nom du fichier de l'événement sur votre ordinateur local.

Indiquez `-` comme valeur de nom de fichier à lire à partir de `stdin`.

Cette option est obligatoire.

`--force`, `-f`

Remplacez un événement de test partageable portant le même nom.

`--help, -h`

Affichez le message d'aide, puis fermez.

`--name` *TEXT*

Nom sous lequel enregistrer l'événement de test partageable.

Si un événement de test partageable portant le même nom existe dans le registre du EventBridge schéma, il ne le AWS SAM CLI remplacera pas. Pour le remplacer, ajoutez l'option `--force`.

`--output-file` *FILENAME*

Chemin et nom du fichier dans lesquels enregistrer l'événement sur votre ordinateur local.

Si vous ne fournissez pas cette option, le contenu de l'événement de test partageable AWS SAM CLI sera affiché sur votre console.

`--stack-name` *TEXT*

Nom de la CloudFormation pile associée à la fonction Lambda.

Cette option est obligatoire si vous fournissez l'ID logique de la fonction Lambda en tant qu'argument.

Exemple

Pour un exemple d'utilisation de cette commande, reportez-vous àEnregistrement d'événements de test partageables.

sam sync

Cette page fournit des informations de référence pour la AWS Serverless Application Model commande Command Line Interface (AWS SAMCLI)`sam sync`.

- Pour une introduction à la AWS SAMCLI, voir Qu'est-ce que c'est AWS SAMCLI ?
- Pour obtenir de la documentation sur l'utilisation du AWS SAMCLI, voirAWS SAM CLI.

La commande `sam sync` synchronise les modifications apportées de l'application locale à AWS Cloud.

Usage

```
$ sam sync <options>
```

Options

`--base-dir, -s` *`DIRECTORY`*

Résout les chemins relatifs au code source de la fonction ou de la couche par rapport à ce répertoire. Utilisez cette option pour modifier la façon dont les chemins d'accès relatifs aux dossiers de code source sont résolus. Par défaut, les chemins relatifs sont résolus en fonction de l'emplacement du AWS SAM modèle.

Outre les ressources de l'application ou de la pile racine que vous créez, cette option s'applique également aux applications ou piles imbriquées. En outre, cette option s'applique aux types de ressources et propriétés suivants :

- Type de ressource : propriété `AWS::Serverless::Function` : `CodeUri`
- Type de ressource : attribut de ressource `AWS::Serverless::Function` : entrée `Metadata` : `DockerContext`
- Type de ressource : propriété `AWS::Serverless::LayerVersion` : `ContentUri`
- Type de ressource : propriété `AWS::Lambda::Function` : `Code`

- Type de ressource : propriété `AWS::Lambda::LayerVersion` : `Content`

`--build-image` *`TEXT`*

L'URI de l'image de conteneur que vous souhaitez utiliser lors de la création de votre application. AWS SAM Utilise par défaut l'URI du référentiel d'images de conteneurs provenant d'Amazon Elastic Container Registry (Amazon ECR) Public. Spécifiez cette option pour utiliser une autre image.

Vous pouvez utiliser cette option plusieurs fois au sein d'une même commande. Chaque option accepte une chaîne de caractères ou une paire clé-valeur.

- Chaîne de caractères : si vous spécifiez une chaîne de caractères, il s'agit de l'URI de l'image de conteneur qui sera utilisée par toutes les ressources de l'application. En voici un exemple :

```
$ sam sync --build-image amazon/aws-sam-cli-build-image-python3.8
```

- Paire clé-valeur : spécifiez le nom de la ressource comme clé et l'URI de l'image du conteneur à utiliser avec cette ressource comme valeur. Utilisez ce format pour spécifier un URI d'image de conteneur différent pour chaque ressource de votre application. Voici un exemple :

```
$ sam sync --build-image Function1=amazon/aws-sam-cli-build-image-python3.8
```

Cette option ne s'applique que si l'option `--use-container` est spécifiée, dans le cas contraire, une erreur se produit.

`--build-in-source | --no-build-in-source`

Fournit `--build-in-source` pour créer votre projet directement dans le dossier source.

L'option `--build-in-source` prend en charge les systèmes d'exécution ainsi que les méthodes de création suivantes :

- exécutions : toute méthode d'exécution Node.js prise en charge par l'option `sam init --runtime`.
- Méthodes de construction : `Makefile`, `esbuild`.

L'option `--build-in-source` n'est pas compatible avec celles suivantes :

- `--use-container`

Par défaut : `--no-build-in-source`

`--capabilities LIST`

Liste des fonctionnalités que vous spécifiez pour CloudFormation autoriser la création de certaines piles. Certains modèles de pile peuvent inclure des ressources susceptibles d'affecter les autorisations de votre Compte AWS. Par exemple, en créant de nouveaux utilisateurs Gestion des identités et des accès AWS (IAM). Spécifiez cette option pour remplacer les valeurs par défaut. Les valeurs valides sont les suivantes :

- CAPABILITY_IAM
- CAPABILITY_NAMED_IAM
- CAPABILITY_RESOURCE_POLICY
- CAPABILITY_AUTO_EXPAND

Par défaut : `CAPABILITY_NAMED_IAM` et `CAPABILITY_AUTO_EXPAND`

`--code`

Par défaut, AWS SAM synchronise toutes les ressources de votre application. Spécifiez cette option pour synchroniser uniquement les ressources de code, qui comprennent les éléments suivants :

- `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`

Pour synchroniser les ressources de code, AWS SAM utilise APIs directement le AWS service, au lieu de le déployer via AWS CloudFormation. Pour mettre à jour votre CloudFormation stack, exécutez sam sync --watch ousam deploy.

`--config-env` *TEXT*

Le nom d'environnement spécifiant les valeurs de paramètre par défaut dans le fichier de configuration à utiliser. La valeur par défaut est « par défaut ». Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--config-file` *PATH*

Le chemin d'accès et le nom du fichier de configuration contenant les valeurs de paramètres par défaut à utiliser. La valeur par défaut est « `samconfig.toml` » dans la racine du répertoire du projet. Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--dependency-layer | --no-dependency-layer`

Spécifie s'il faut séparer les dépendances des fonctions individuelles dans une autre couche pour accélérer le processus de synchronisation.

Par défaut : `--dependency-layer`

`--image-repository` *TEXT*

Le nom du référentiel Amazon Elastic Container Registry (Amazon ECR) vers lequel cette commande télécharge l'image de la fonction. Requis pour les fonctions déclarées avec le type de package `Image`.

`--image-repositories` *TEXT*

Un mappage des fonctions vers l'URI de leur référentiel Amazon ECR. Référencez les fonctions par leur identifiant logique. Voici un exemple :

```
$ sam sync --image-repositories Function1=123456789012.dkr.ecr.us-
east-1.amazonaws.com/my-repo
```

Vous pouvez spécifier cette option plusieurs fois au sein d'une même commande.

`--kms-key-id` *TEXT*

L'ID d'une clé AWS Key Management Service (AWS KMS) utilisée pour chiffrer les artefacts inactifs dans le compartiment Amazon S3. Si vous ne spécifiez pas cette option, AWS SAM utilise les clés de chiffrement gérées par Amazon S3.

`--metadata`

Un mappage de métadonnées à attacher à tous les artefacts que vous référencez dans votre modèle.

`--notification-arns` *`LIST`*

Liste des rubriques Amazon Simple Notification Service (Amazon SNS) ARNs associées à la CloudFormation pile.

`--no-use-container`

Option qui vous permet d'utiliser le kit d'outils IDE pour définir le comportement par défaut.

`--parameter-overrides`

Chaîne contenant des remplacements de CloudFormation paramètres codés sous forme de paires clé-valeur. Utilisez le même format que le AWS Command Line Interface (AWS CLI). Le AWS SAMCLI format est basé sur des mots clés et valeurs explicites, chaque remplacement étant séparé par un espace. Voici deux exemples :

- `--parameter-overrides ParameterKey=hello,ParameterValue=world`
- `--parameter-overrides ParameterKey=hello,ParameterValue=world ParameterKey=example1,ParameterValue=example2 ParameterKey=apple,ParameterValue=banana`

`--resource` *`TEXT`*

Spécifie le type de ressource à synchroniser. Pour synchroniser plusieurs ressources, vous pouvez spécifier cette option plusieurs fois. Cette option est prise en charge avec l'option `--code`. La valeur doit correspondre à l'une des ressources répertoriées sous `--code`. Par exemple, `--resource AWS::Serverless::Function --resource AWS::Serverless::LayerVersion`.

`--resource-id` *`TEXT`*

Spécifie l'identifiant de la ressource à synchroniser. Pour synchroniser plusieurs ressources, vous pouvez spécifier cette option plusieurs fois. Cette option est prise en charge avec l'option `--code`. Par exemple, `--resource-id Function1 --resource-id Function2`.

`--role-arn` *`TEXT`*

Nom de ressource Amazon (ARN) d'un rôle IAM CloudFormation assumé lors de l'application de l'ensemble de modifications.

`--s3-bucket` *TEXT*

Nom du compartiment Amazon Simple Storage Service (Amazon S3) dans lequel cette commande télécharge CloudFormation votre modèle. Si le modèle est supérieur à 51 200 octets, le `--s3-bucket` ou l'option `--resolve-s3` sont obligatoires. Si vous spécifiez les deux options `--s3-bucket` et `--resolve-s3`, une erreur se produit.

`--s3-prefix` *TEXT*

Le préfixe ajouté aux noms des artefacts que vous chargez dans le compartiment Amazon S3. Le nom du préfixe est un nom de chemin d'accès (nom de dossier) pour le compartiment Amazon S3. Cela s'applique uniquement aux fonctions déclarées avec le type de package `Zip`.

`--save-params`

Enregistre les paramètres que vous fournissez sur la ligne de commande dans le fichier AWS SAM de configuration.

`--skip-deploy-sync | --no-skip-deploy-sync`

Spécifie `--skip-deploy-sync` pour ignorer la synchronisation initiale de l'infrastructure si elle n'est pas requise. Le AWS SAMCLI comparera votre AWS SAM modèle local avec le CloudFormation modèle déployé et n'effectuera un déploiement que si une modification est détectée.

Spécifie `--no-skip-deploy-sync` d'effectuer un CloudFormation déploiement à `sam sync` chaque exécution.

Pour en savoir plus, veuillez consulter la section Ignorer le CloudFormation déploiement initial.

Par défaut : `--skip-deploy-sync`

`--stack-name` *TEXT*

Le nom de la CloudFormation pile de votre application.

Cette option est obligatoire.

`--tags` *LIST*

Liste de balises à associer à la pile créée ou mise à jour. CloudFormation propage également ces balises aux ressources de la pile qui les supportent.

`--template-file, --template, -t` *PATH*

Le chemin et le nom du fichier où se trouve votre AWS SAM modèle.

> **Note**
>
> Si vous spécifiez cette option, AWS SAM déploie uniquement le modèle et les ressources locales vers lesquels il pointe.

`--use-container, -u`

Si vos fonctions dépendent de packages dont les dépendances sont compilées de manière native, utilisez cette option pour créer votre fonction dans un Docker conteneur AWS Lambda similaire.

> **Note**
>
> Actuellement, cette option n'est pas compatible avec `--dependency-layer`. Si vous utilisez `--use-container` avec `--dependency-layer`, la CLI AWS SAM vous en informe et continue avec `--no-dependency-layer`.

`--watch`

Lance un processus qui surveille les modifications apportées par votre application locale et les synchronise automatiquement avec le AWS Cloud. Par défaut, lorsque vous spécifiez cette option, toutes les ressources de votre application sont AWS SAM synchronisées au fur et à mesure que vous les mettez à jour. Avec cette option, AWS SAM effectue un CloudFormation déploiement initial. AWS SAM Utilise ensuite le AWS service APIs pour mettre à jour les ressources du code. AWS SAM utilise CloudFormation pour mettre à jour les ressources de l'infrastructure lorsque vous mettez à jour votre AWS SAM modèle.

`--watch-exclude` *TEXT*

Empêche un fichier ou un dossier d'être observé en cas de modification de fichier. Pour utiliser cette option, `--watch` doit également être fournie.

Cette option reçoit une paire clé-valeur :

- clé : ID logique d'une fonction Lambda dans votre application.
- valeur : nom du fichier ou du dossier associé à exclure.

Lorsque vous mettez à jour des fichiers ou des dossiers spécifiés avec l'`--watch-excludeoption`, aucune synchronisation ne AWS SAM CLI sera lancée. Toutefois, lorsqu'une

mise à jour d'autres fichiers ou dossiers initie une synchronisation, ceux-ci ou sont inclus dans cette synchronisation.

Vous pouvez fournir cette option plusieurs fois dans une seule commande.

Exemples

Pour des exemples d'utilisation de cette commande, reportez-vous àOptions pour la commande sam sync.

sam traces

Cette page fournit des informations de référence pour la AWS Serverless Application Model commande Command Line Interface (AWS SAMCLI)`sam traces`.

Pour une introduction à la AWS SAMCLI, voir Qu'est-ce que c'est AWS SAMCLI ?

La `sam traces` commande récupère les AWS X-Ray traces Compte AWS dans votre fichier. Région AWS

Usage

```
$ sam traces <options>
```

Options

`--config-env` *`TEXT`*

Le nom d'environnement spécifiant les valeurs de paramètre par défaut dans le fichier de configuration à utiliser. La valeur par défaut est « par défaut ». Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--config-file` *`PATH`*

Le chemin d'accès et le nom du fichier de configuration contenant les valeurs de paramètres par défaut à utiliser. La valeur par défaut est « `samconfig.toml` » dans la racine du répertoire du projet. Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--end-time` *TEXT*

Récupère les traces jusqu'à ce moment. L'heure peut être une valeur relative comme « il y a 5 minutes », « demain » ou un horodatage formaté comme « 2018-01-01 10:10:10 ».

`--output` *TEXT*

Spécifie le format de sortie pour les journaux. Pour imprimer des journaux formatés, spécifiez `text`. Pour imprimer les journaux au format JSON, spécifiez `json`.

`--save-params`

Enregistrez les paramètres que vous fournissez sur la ligne de commande dans le fichier AWS SAM de configuration.

`--start-time` *TEXT*

Récupère les traces en ce moment. L'heure peut être une valeur relative comme « il y a 5 minutes », « hier » ou un horodatage formaté comme « 2018-01-01 10:10:10 ». La valeur par défaut est « il y a 10 minutes ».

`--tail`

Suit la sortie de la trace. Cette méthode ignore l'argument des heures de fin de fin et continue d'afficher les traces au fur et à mesure qu'elles deviennent disponibles.

`--trace-id` *TEXT*

Identifiant unique pour une trace X-Ray.

Exemples

Exécutez la commande suivante pour récupérer les traces de X-Ray par identifiant.

```
$ sam traces --trace-id tracing-id-1 --trace-id tracing-id-2
```

Exécutez la commande suivante pour suivre les traces X-Ray au fur et à mesure qu'elles deviennent disponibles.

```
$ sam traces --tail
```

sam validate

Cette page fournit des informations de référence pour la AWS Serverless Application Model commande Command Line Interface (AWS SAMCLI)`sam validate`.

Pour une introduction à la AWS SAMCLI, voir Qu'est-ce que c'est AWS SAMCLI ?

La `sam validate` commande vérifie si un fichier AWS SAM modèle est valide.

Usage

```
$ sam validate <options>
```

Options

`--config-env` *TEXT*

Le nom d'environnement spécifiant les valeurs de paramètre par défaut dans le fichier de configuration à utiliser. La valeur par défaut est « par défaut ». Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--config-file` *PATH*

Le chemin d'accès et le nom du fichier de configuration contenant les valeurs de paramètres par défaut à utiliser. La valeur par défaut est « samconfig.toml » dans la racine du répertoire du projet. Pour plus d'informations sur les fichiers de configuration, consultez Fichier de configuration CLI AWS SAM.

`--debug`

Active la journalisation de débogage pour imprimer le message de débogage généré par la CLI AWS SAM et pour afficher les horodatages.

`--lint`

Exécutez la validation linting sur le modèle via cfn-lint. Créez un fichier de configuration `cfnlintrc` pour spécifier des paramètres supplémentaires. Pour plus d'informations, consultez cfn-lint dans le référentiel.CloudFormation GitHub

`--profile` *TEXT*

Le profil spécifique de votre fichier d'informations d'identification qui obtient les AWS informations d'identification.

`--region TEXT`

AWS Région dans laquelle le déploiement doit être effectué. Par exemple, us-east-1.

`--save-params`

Enregistrez les paramètres que vous fournissez sur la ligne de commande dans le fichier AWS SAM de configuration.

`--template-file, --template, -t PATH`

Le fichier AWS SAM modèle. La valeur par défaut est `template.[yaml|yml]`.

Si votre modèle se trouve dans votre répertoire de travail actuel et qu'il est nommé `template.[yaml|yml|json]`, cette option n'est pas obligatoire.

Si vous venez d'exécuter sam build, cette option n'est pas obligatoire.

Exemple

Pour un exemple d'utilisation de cette commande pour valider un modèle, reportez-vous àValider les fichiers AWS SAM modèles.

Pour un exemple d'utilisation de cette commande avec cfn-lint, reportez-vous à. Validez vos AWS SAM applications avec CloudFormation Linter

AWS SAMCLIgestion

Cette section contient des informations sur les moyens de gérer et de personnaliser votre version du AWS SAMCLI. Cela inclut des informations sur la manière dont vous pouvez configurer les valeurs des paramètres de AWS SAMCLI commande à l'aide d'un fichier de configuration au niveau du projet. Il inclut également des informations sur la gestion des différentes versions de votre compte AWS SAMCLI, la définition des AWS informations d'identification AWS SAM permettant de passer des appels aux AWS services en votre nom et les différentes méthodes de personnalisation AWS SAM. Cette section se termine par une section sur le AWS SAM dépannage général.

Rubriques

- Fichier de configuration CLI AWS SAM
- Gestion des versions CLI AWS SAM

- Configuration des AWS informations d'identification
- Télémétrie dans la CLI AWS SAM
- Résolution des problèmes de la CLI AWS SAM

Fichier de configuration CLI AWS SAM

L'interface de ligne de AWS Serverless Application Model commande (AWS SAMCLI) prend en charge un fichier de configuration au niveau du projet que vous pouvez utiliser pour configurer les valeurs des paramètres de AWS SAMCLI commande.

Pour obtenir de la documentation sur la création et l'utilisation de fichiers de configuration, veuillez consulter la section Configuration de la CLI AWS SAM.

Rubriques

- Paramètres du fichier de configuration par défaut
- Formats de fichiers de configuration pris en charge
- Spécifier un fichier de configuration
- Principes de base relatifs au fichier de configuration
- Règles de valeur des paramètres
- Ordre de priorité de configuration
- Création et modification de fichiers de configuration

Paramètres du fichier de configuration par défaut

AWS SAM utilise les paramètres de fichier de configuration par défaut suivants :

- Nom : `samconfig`.
- Emplacement : à la racine de votre projet. Il s'agit du même emplacement que celui de votre fichier `template.yaml`.
- Format : `TOML`. Pour en savoir plus, veuillez consulter la section TOML dans la documentation TOML.

Voici un exemple de structure de projet qui inclut le nom et l'emplacement du fichier de configuration par défaut :

```
sam-app
### README.md
### __init__.py
### events
### hello_world
### samconfig.toml
### template.yaml
### tests
```

Voici un exemple de fichier de `samconfig.toml` :

```
...
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
```

Formats de fichiers de configuration pris en charge

Les formats `TOML` et `[YAML|YML]` sont pris en charge. Consultez la syntaxe de base suivante :

TOML

```
version = 0.1
[environment]
[environment.command]
[environment.command.parameters]
option = parameter value
```

YAML

```
version: 0.1
environment:
  command:
    parameters:
      option: parameter value
```

Spécifier un fichier de configuration

Par défaut, la CLI AWS SAM recherche un fichier de configuration dans l'ordre suivant :

1. Fichier de configuration personnalisé : si vous utilisez l'option `--config-file` pour spécifier un nom et un emplacement de fichier, la CLI AWS SAM recherche d'abord ce fichier.
2. Fichier **`samconfig.toml`** par défaut : il s'agit du nom et du format du fichier de configuration par défaut, situé à la racine de votre projet. Si vous ne spécifiez pas de fichier de configuration personnalisé, la CLI AWS SAM recherche ensuite ce fichier.
3. Fichier **`samconfig.[yaml|yml]`** : si le fichier `samconfig.toml` n'existe pas à la racine de votre projet, la CLI AWS SAM recherche ce fichier.

Voici un exemple de spécification d'un fichier de configuration personnalisé à l'aide de l'option `--config-file` :

```
$ sam deploy --config-file myconfig.yaml
```

> **Note**
>
> Le `--config-file` paramètre doit être relatif à l'emplacement du fichier AWS SAM modèle, car il AWS SAMCLI doit déterminer le contexte dans lequel la configuration est appliquée.
> Le `samconfig.toml` fichier gère les paramètres de configuration de votre version de AWS

SAMCLI, et la CLI recherche le `samconfig.toml` fichier (ou le paramètre du fichier de configuration remplacé) dans le dossier relatif du `template.yaml` fichier.

Principes de base relatifs au fichier de configuration

Environnement

Un environnement est un identifiant nommé qui contient un ensemble unique de paramètres de configuration. Vous pouvez avoir plusieurs environnements dans une seule AWS SAM application.

Le nom de l'environnement par défaut est `default`.

Utilisez l' AWS SAMCLI`--config-env`option pour spécifier l'environnement à utiliser.

Commande

La commande est la commande CLI AWS SAM pour laquelle les valeurs des paramètres sont spécifiées.

Pour spécifier les valeurs des paramètres pour toutes les commandes, utilisez l'identificateur `global`.

Lorsque vous faites référence à une commande CLI AWS SAM, remplacez les espaces (–) et les traits d'union ( ) par des traits de soulignement (_). Voir les exemples suivants :

- `build`
- `local_invoke`
- `local_start_api`

Parameters

Les paramètres sont spécifiés sous forme de paires clé-valeur.

- La clé est le nom de l'option de commande CLI AWS SAM.
- La valeur est celle à spécifier.

Lorsque vous spécifiez des clés, utilisez le nom complet de l'option de commande et remplacez les traits d'union (–) par des traits de soulignement (_). Voici quelques exemples :

- `region`
- `stack_name`
- `template_file`

Règles de valeur des paramètres

TOML

- Les valeurs booléennes peuvent être `true` ou `false`. Par exemple, `confirm_changeset = true`.
- Pour les valeurs de chaîne de caractères, utilisez des guillemets (`""`). Par exemple, `region = "us-west-2"`.
- Pour les valeurs de liste, utilisez des guillemets (`""`) et séparez chaque valeur par une espace ( ). Par exemple : `capabilities = "CAPABILITY_IAM CAPABILITY_NAMED_IAM"`.
- Pour la valeur contenant une liste de paires clé-valeur, les paires sont délimitées par des espaces ( ), et la valeur de chaque paire est entourée de guillemets (`\" \"`) encodés. Par exemple, `tags = "project=\"my-application\" stage=\"production\""`.
- Pour les valeurs de paramètres qui peuvent être spécifiés plusieurs fois, la valeur est un tableau d'arguments. Par exemple : `image_repositories = ["my-function-1=image-repo-1", "my-function-2=image-repo-2"]`.

YAML

- Les valeurs booléennes peuvent être `true` ou `false`. Par exemple, `confirm_changeset: true`.
- Pour les entrées contenant une seule valeur de chaîne de caractères, les guillemets (`""`) sont facultatifs. Par exemple, `region: us-west-2`. Cela inclut les entrées qui contiennent plusieurs paires clé-valeur fournies sous la forme d'une seule chaîne de caractères. Voici un exemple :

```
$ sam deploy --tags "foo=bar hello=world"
```

```
default:
  deploy:
    parameters:
      tags: foo=bar hello=world
```

- Pour les entrées qui contiennent une liste de valeurs ou pour les entrées qui peuvent être utilisées plusieurs fois au sein d'une même commande, spécifiez-les sous forme de liste de chaînes de caractères.

  Voici un exemple :

```
$ sam remote invoke --parameter "InvocationType=Event" --parameter "LogType=None"
```

```
default:
  remote_invoke:
    parameters:
        parameter:
        - InvocationType=Event
        - LogType=None
```

Ordre de priorité de configuration

Lors de la configuration des valeurs, l'ordre de priorité est le suivant :

- Les valeurs des paramètres que vous fournissez à la ligne de commande ont la priorité sur les valeurs correspondantes dans le fichier de configuration et la section `Parameters` du fichier modèle.
- Si l'option `--parameter-overrides` est utilisée sur la ligne de commande ou dans votre fichier de configuration avec la clé `parameter_overrides`, ses valeurs ont la priorité sur les valeurs de la section `Parameters` du fichier modèle.
- Dans votre fichier de configuration, les entrées fournies pour une commande spécifique sont prioritaires sur les entrées globales. Dans l'exemple suivant, la commande `sam deploy` utilisera le nom de pile `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
```

Création et modification de fichiers de configuration

Création de fichiers de configuration

Lorsque vous créez une application à l'aide de `sam init`, un fichier `samconfig.toml` par défaut est créé. Vous pouvez également créer manuellement votre fichier de configuration.

Modification des fichiers de configuration

Vous pouvez modifier manuellement vos fichiers de configuration. De plus, lors de tout flux interactif de la CLI AWS SAM, les valeurs configurées seront affichées entre crochets (`[ ]`). Si vous modifiez ces valeurs, la CLI AWS SAM mettra à jour votre fichier de configuration.

Vous trouverez ci-dessous un exemple du flux interactif utilisant la commande `sam deploy --guided` :

```
$ 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
```

Lorsque vous modifiez votre fichier de configuration, la CLI AWS SAM traite les valeurs globales de la manière suivante :

- Si la valeur du paramètre existe dans la section `global` de votre fichier de configuration, la CLI AWS SAM n'écrit pas la valeur dans la section de commande spécifique.
- Si la valeur du paramètre existe à la fois dans les sections de commande `global` et dans des sections de commande spécifiques, la CLI AWS SAM supprime l'entrée spécifique au profit de la valeur globale.

Gestion des versions CLI AWS SAM

Gérez les versions de votre interface de ligne de AWS Serverless Application Model commande (AWS SAMCLI) par le biais de la mise à niveau, de la rétrogradation et de la désinstallation. Vous pouvez également télécharger et installer la compilation nocturne de la CLI AWS SAM.

Rubriques

- Mise à niveau de la CLI AWS SAM en cours
- Désinstallation de la CLI AWS SAM
- Passer de l'utilisation de Homebrew pour gérer la CLI AWS SAM
- Gestion de la compilation nocturne de la CLI AWS SAM
- Installation de la CLI AWS SAM dans un environnement virtuel à l'aide de pip
- Gestion de la CLI AWS SAM avec Homebrew
- Résolution des problèmes

Mise à niveau de la CLI AWS SAM en cours

Linux

Pour mettre à niveau la CLI AWS SAM sous Linux, suivez les instructions d'installation de Installation de la CLI AWS SAM, mais ajoutez l'option `--update` à la commande d'installation, comme suit :

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

macOS

Le AWS SAMCLI doit être mis à niveau en utilisant la même méthode que celle utilisée pour l'installer. Nous vous recommandons d'utiliser le programme d'installation du package pour installer et mettre à niveau le AWS SAMCLI.

Pour mettre à niveau la CLI AWS SAMà l'aide du programme d'installation de packages, installez la dernière version du package. Pour obtenir des instructions, veuillez consulter Installation de la CLI AWS SAM.

Windows

Pour effectuer la mise à niveau AWS SAMCLI, répétez les étapes d'installation de Installer la CLI AWS SAM Windows.

Désinstallation de la CLI AWS SAM

Linux

Pour désinstaller la CLI AWS SAM sous Linux, vous devez supprimer le lien symbolique et le répertoire d'installation en exécutant les commandes suivantes :

1. Localisez le lien symbolique et les chemins d'installation.

   - Utilisez la commande which pour trouver le lien symbolique.

     ```
     which sam
     ```

     La sortie indique le chemin où se trouvent AWS SAM les fichiers binaires, par exemple :

     ```
     /usr/local/bin/sam
     ```

   - Utilisez la commande ls pour rechercher le répertoire vers lequel pointe le lien symbolique.

```
ls -l /usr/local/bin/sam
```

Dans l'exemple suivant, le répertoire d'installation est `/usr/local/aws-sam-cli`.

```
 lrwxrwxrwx 1 ec2-user ec2-user 49 Oct 22 09:49 /usr/local/bin/sam -> /usr/local/
aws-sam-cli/current/bin/sam
```

2. Supprimez le lien symbolique.

```
sudo rm /usr/local/bin/sam
```

3. Supprimez le répertoire d'installation.

```
sudo rm -rf /usr/local/aws-sam-cli
```

macOS

Désinstallez la CLI AWS SAM par la même méthode que celle utilisée pour l'installer. Nous vous recommandons d'utiliser le programme d'installation du package pour installer le AWS SAMCLI.

Si vous avez installé la CLI AWS SAMà l'aide du programme d'installation de packages, suivez ces étapes pour la désinstaller.

Pour désinstaller la CLI AWS SAM

1. Supprimez le programme CLI AWS SAM en modifiant et en exécutant ce qui suit :

```
$ sudo rm -rf /path-to/aws-sam-cli
```

   a. *`sudo`*— Si votre utilisateur dispose d'autorisations d'écriture sur l'endroit où le AWS SAMCLI programme est installé, cela n'sudoest pas obligatoire. Dans le cas contraire, la valeur sudo est obligatoire.
   b. *`/path-to`*— Chemin d'accès à l'endroit où vous avez installé le AWS SAMCLI programme. L'emplacement par défaut est `/usr/local`.
2. AWS SAMCLI`$PATH`Supprimez-le en modifiant et en exécutant ce qui suit :

```
$ sudo rm -rf /path-to-symlink-directory/sam
```

a. *sudo*— Si votre utilisateur dispose d'autorisations d'écriture pour`$PATH`, ce n'`sudo`est pas obligatoire. Dans le cas contraire, la valeur sudo est obligatoire.

b. *path-to-symlink-directory*— Votre variable d'`$PATH`environnement. L'emplacement par défaut est `/usr/local/bin`.

3. Procédez comme suit pour vérifier que la CLI AWS SAM est bien désinstallée :

```
$ sam --version
command not found: sam
```

Windows

Pour désinstaller la CLI AWS SAM en utilisant les paramètres Windows, procédez aux étapes suivantes :

1. Dans le menu Démarrer, recherchez « Ajouter ou supprimer des programmes ».
2. Sélectionnez le résultat nommé Interface de ligne de commande AWS SAM et cliquez sur Uninstall (Désinstaller) pour lancer le programme de désinstallation.
3. Confirmez que vous souhaitez désinstaller le AWS SAMCLI.

Passer de l'utilisation de Homebrew pour gérer la CLI AWS SAM

Si vous avez l'habitude Homebrew d'installer et de mettre à niveau le AWS SAMCLI, nous vous recommandons d'utiliser une méthode AWS compatible. Suivez ces instructions pour passer à une méthode prise en charge.

Pour passer de l'utilisation de Homebrew

1. Suivez les instructions décrites dans Désinstaller une CLI Homebrew installée AWS SAM pour désinstaller la version gérée par Homebrew.
2. Suivez les instructions décrites dans Installer la CLI AWS SAM pour installer la CLI AWS SAM à l'aide d'une méthode prise en charge.

Gestion de la compilation nocturne de la CLI AWS SAM

Vous pouvez télécharger et installer la compilation nocturne de la CLI AWS SAM. Celle-ci contient une version préliminaire du code de la CLI AWS SAM qui peut être moins stable que la version de

production. Une fois installée, vous pouvez utiliser la compilation nocturne à l'aide de la commande `sam-nightly`. Vous pouvez installer à la fois les versions de production et de compilation nocturne de la CLI AWS SAM et les utiliser en même temps.

> **Note**
>
> La compilation nocturne ne contient pas de version préliminaire de l'image de compilation. Ainsi, la création de votre application sans serveur à l'aide de l'option --use-container utilise la dernière version de production de l'image de compilation.

Installation de la compilation nocturne de la CLI AWS SAM

Pour installer la compilation nocturne de la CLI AWS SAM, suivez ces instructions.

Linux

Vous pouvez installer la version de compilation nocturne de la CLI AWS SAM sur la plateforme Linux x86_64à l'aide du programme d'installation de packages.

Pour installer la compilation nocturne de la CLI AWS SAM

1. Téléchargez le programme d'installation du sam-cli-nightlypackage depuis le aws-sam-cli GitHubréférentiel.
2. Suivez les étapes d'installation de la CLI AWS SAM pour installer le package de compilation nocturne.

macOS

Vous pouvez installer la version de compilation nocturne de la CLI AWS SAM sur macOS à l'aide du programme d'installation du package de compilation nocturne.

Pour installer la compilation nocturne de la CLI AWS SAM

1. Téléchargez le programme d'installation du package pour votre plate-forme depuis sam-cli-nightlyle aws-sam-cli GitHubréférentiel.
2. Suivez les étapes d'installation de la CLI AWS SAM pour installer le package de compilation nocturne.

Windows

La version de compilation nocturne de la CLI AWS SAM est disponible via le lien de téléchargement suivant : compilation nocturne de la CLI AWS SAM. Pour installer la compilation nocturne sur Windows, exécutez les mêmes étapes que celles décrites dans Installer la CLI AWS SAM, mais utilisez plutôt le lien de téléchargement de compilation nocturne à la place.

Pour vérifier que vous avez installé la version de compilation nocturne, exécutez la commande sam-nightly --version. La sortie de cette commande est sous la forme `1.X.Y.dev<YYYYMMDDHHmm>`, par exemple :

```
SAM CLI, version 1.20.0.dev202103151200
```

Passer de Homebrew au package d'installation

Si vous utilisez Homebrew pour installer et mettre à niveau la compilation nocturne de la CLI AWS SAM et souhaitez opter pour l'utilisation du programme d'installation de package, procédez comme suit.

Pour passer de Homebrew au package d'installation

1. Désinstallez la compilation nocturne de la CLI AWS SAM installée par Homebrew.

   ```
   $ brew uninstall aws-sam-cli-nightly
   ```

2. Procédez comme suit pour vérifier que la compilation nocturne de la CLI AWS SAM est bien désinstallée :

   ```
   $ sam-nightly --version
   zsh: command not found: sam-nightly
   ```

3. Suivez les étapes décrites dans la section précédente pour installer la compilation nocturne de la CLI AWS SAM.

Installation de la CLI AWS SAM dans un environnement virtuel à l'aide de pip

Nous vous recommandons d'utiliser le programme d'installation du package natif pour installer le AWS SAMCLI. Si vous devez utiliser pip, nous vous recommandons d'installer la CLI AWS SAM dans un environnement virtuel. Cela garantit un environnement d'installation propre et un environnement isolé en cas d'erreurs.

> **Note**
> À compter du 24 octobre 2023, AWS SAM CLI le support pourPython 3.7. Pour en savoir plus, veuillez consulter la section Fin de la prise en charge de Python 3.7 par l'CLIAWS SAM.

Pour installer la CLI AWS SAM dans un environnement virtuel

1. À partir du répertoire de départ de votre choix, créez un environnement virtuel et nommez-le.

   Linux / macOS

   ```
   $ mkdir project
   $ cd project
   $ python3 -m venv venv
   ```

   Windows

   ```
   > mkdir project
   > cd project
   > py -3 -m venv venv
   ```

2. Activer l'environnement virtuel

   Linux / macOS

   ```
   $ . venv/bin/activate
   ```

   L'invite change pour vous indiquer que votre environnement virtuel est actif.

   ```
   (venv) $
   ```

   Windows

   ```
   > venv\Scripts\activate
   ```

   L'invite change pour vous indiquer que votre environnement virtuel est actif.

   ```
   (venv) >
   ```

3. Installez la CLI AWS SAM dans votre environnement virtuel.

```
(venv) $ pip install --upgrade aws-sam-cli
```

4. Vérifiez que la CLI AWS SAM est installée correctement.

```
(venv) $ sam --version
SAM CLI, version 1.94.0
```

5. La commande `deactivate` vous permet de quitter l'environnement virtuel. Réactivez l'environnement chaque fois que vous démarrez une nouvelle session.

Gestion de la CLI AWS SAM avec Homebrew

> **Note**
>
> À compter de septembre 2023, le programme d'Homebrewinstallation AWS géré pour le AWS SAMCLI (`aws/tap/aws-sam-cli`) ne AWS sera plus maintenu. Pour continuer à utiliser Homebrew, vous pouvez utiliser le programme d'installation géré par la communauté (`aws-sam-cli`). À partir de septembre 2023, toute commande Homebrew faisant référence à `aws/tap/aws-sam-cli` sera redirigée vers `aws-sam-cli`.
> Nous vous recommandons d'utiliser nos méthodes d'installation et de mise à niveau.

Installation de la CLI AWS SAMà l'aide de Homebrew

> **Note**
>
> Ces instructions utilisent le programme d' AWS SAMCLIHomebrewinstallation géré par la communauté. Pour assistance complémentaire, consultez le référentiel homebrew-core.

Pour installer la CLI AWS SAM

1. Exécutez les commandes suivantes :

```
$ brew install aws-sam-cli
```

2. Vérifiez l'installation :

```
$ sam --version
```

Une fois l'installation réussie du AWS SAMCLI, vous devriez voir le résultat suivant :

```
SAM CLI, version 1.94.0
```

Mise à niveau de la CLI AWS SAMà l'aide de Homebrew

Pour mettre à niveau la CLI AWS SAMà l'aide de Homebrew, exécutez la commande suivante :

```
$ brew upgrade aws-sam-cli
```

Désinstaller une CLI Homebrew installée AWS SAM

Si la CLI AWS SAM a été installée à l'aide de Homebrew, procédez comme suit pour la désinstaller.

Pour désinstaller la CLI AWS SAM

1. Exécutez les commandes suivantes :

   ```
   $ brew uninstall aws-sam-cli
   ```

2. Procédez comme suit pour vérifier que la CLI AWS SAM est bien désinstallée :

   ```
   $ sam --version
   command not found: sam
   ```

Passer au programme d'installation géré par la communauté Homebrew

Si vous utilisez le programme d'Homebrewinstallation AWS géré (`aws/tap/aws-sam-cli`) et que vous préférez continuer à l'utiliserHomebrew, nous vous recommandons de passer au programme d'Homebrewinstallation géré par la communauté (`aws-sam-cli`).

Pour passer dans une seule commande, exécutez la commande suivante :

```
$ brew uninstall aws-sam-cli && brew untap aws/tap && brew cleanup aws/tap && brew
 update && brew install aws-sam-cli
```

Suivez ces instructions pour exécuter chaque commande individuellement.

Pour passer au programme d'installation géré par la communauté Homebrew

1. Désinstallez la Homebrew version AWS gérée de AWS SAMCLI :

```
$ brew uninstall aws-sam-cli
```

2. Vérifiez que la CLI AWS SAM a été désinstallée :

```
$ which sam
sam not found
```

3. Supprimez le AWS SAMCLI robinet AWS géré :

```
$ brew untap aws/tap
```

   Si vous recevez un message d'erreur comme celui-ci, ajoutez l'option `--force` et réessayez.

```
Error: Refusing to untap aws/tap because it contains the following installed
 formulae or casks:
aws-sam-cli-nightly
```

4. Supprimez les fichiers mis en cache pour le programme d'installation AWS géré :

```
$ brew cleanup aws/tap
```

5. Mettre à jour Homebrew et toutes les formules :

```
$ brew update
```

6. Installez la version gérée par la communauté de AWS SAMCLI :

```
$ brew install aws-sam-cli
```

7. Vérifiez que la CLI AWS SAM est correctement installée :

```
$ sam --version
SAM CLI, version 1.94.0
```

Résolution des problèmes

Si vous rencontrez des erreurs lors de l'installation ou de l'utilisation du AWS SAMCLI, consultezRésolution des problèmes de la CLI AWS SAM.

Configuration des AWS informations d'identification

L'interface de ligne de AWS SAM commande (CLI) vous oblige à définir des AWS informations d'identification afin qu'elle puisse effectuer des appels aux AWS services en votre nom. Par exemple, il AWS SAMCLI passe des appels à Amazon S3 et CloudFormation.

Vous avez peut-être déjà défini des AWS informations d'identification pour utiliser des AWS outils, tels que l'un des AWS SDKs ou des AWS CLI. Si ce n'est pas le cas, cette rubrique présente les approches recommandées pour définir les AWS informations d'identification.

Pour définir les AWS informations d'identification, vous devez disposer de l'ID de clé d'accès et de votre clé d'accès secrète pour l'utilisateur IAM que vous souhaitez configurer. Pour plus d'informations sur les clés d'accès IDs et les clés d'accès secrètes, consultez la section Gestion des clés d'accès pour les utilisateurs IAM dans le guide de l'utilisateur IAM.

Ensuite, déterminez si vous l'avez AWS CLI installé. Puis suivez les instructions de configuration dans l'une des sections suivantes.

À l'aide du AWS CLI

Si vous l'avez AWS CLI installé, utilisez la `aws configure` commande et suivez les instructions :

```
$ 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]:
```

Pour obtenir des informations sur la commande `aws configure`, voir Configuration rapide de la AWS CLI dans le Guide de l'utilisateur AWS Command Line Interface .

Ne pas utiliser le AWS CLI

Si vous ne l'avez pas AWS CLI installé, vous pouvez créer un fichier d'informations d'identification ou définir des variables d'environnement :

- Fichier d'informations d'identification : vous pouvez définir les informations d'identification dans le fichier AWS d'informations d'identification de votre système local. Ce fichier doit être situé dans l'un des emplacements suivants :
  - `~/.aws/credentials` sous Linux ou macOS
  - `C:\Users\USERNAME\.aws\credentials` sous Windows

  Ce fichier doit contenir des lignes au format suivant :

  ```
  [default]
  aws_access_key_id = your_access_key_id
  aws_secret_access_key = your_secret_access_key
  ```

- Environment variables (Variables d'environnement) – Vous pouvez définir les variables d'environnement de `AWS_ACCESS_KEY_ID` et `AWS_SECRET_ACCESS_KEY`.

  Pour définir ces variables sous Linux ou macOS, utilisez la commande Exportation :

  ```
  export AWS_ACCESS_KEY_ID=your_access_key_id
  export AWS_SECRET_ACCESS_KEY=your_secret_access_key
  ```

  Pour définir ces variables sous Windows, utilisez la commande set (définir) :

  ```
  set AWS_ACCESS_KEY_ID=your_access_key_id
  set AWS_SECRET_ACCESS_KEY=your_secret_access_key
  ```

Télémétrie dans la CLI AWS SAM

Chez AWS, nous développons et lançons des services en fonction de ce que nous apprenons des interactions avec les clients. Nous utilisons les commentaires des clients pour améliorer notre produit. La télémétrie est une information supplémentaire qui nous aide à mieux comprendre les besoins de nos clients, diagnostiquer les problèmes et fournir des fonctions permettant d'améliorer l'expérience client.

L'interface de ligne de AWS SAM commande (CLI) collecte des données télémétriques, telles que des mesures d'utilisation génériques, des informations sur le système et l'environnement, ainsi que

des erreurs. Pour plus de détails sur les types de télémétrie collectés, consultez Type d'informations à collecter.

La CLI AWS SAMne collecte pas les informations personnelles, telles que les noms d'utilisateur ou les adresses e-mail. Elle n'extrait pas non plus les informations sensibles au niveau du projet.

Les clients contrôlent si la télémétrie est activée et peuvent modifier leurs paramètres à tout moment. Si la télémétrie reste activée, la CLI AWS SAM envoie des données de télémétrie en arrière-plan sans nécessiter d'interaction supplémentaire avec le client.

Désactivation de la télémétrie pour une session

Dans les systèmes d'exploitation macOS et Linux, vous pouvez désactiver la télémétrie pour une seule session. Pour désactiver la télémétrie de votre session en cours, exécutez la commande suivante pour définir la variable d'environnement `SAM_CLI_TELEMETRY` sur `false`. Répétez la commande pour chaque nouveau terminal ou chaque nouvelle session.

```
export SAM_CLI_TELEMETRY=0
```

Désactivation de la télémétrie pour votre profil dans toutes les sessions

Exécutez les commandes suivantes pour désactiver la télémétrie pour toutes les sessions lorsque vous exécutez la CLI AWS SAM sur votre système d'exploitation.

Pour désactiver la télémétrie sous Linux

1. Exécuter :

   ```
   echo "export SAM_CLI_TELEMETRY=0" >>~/.profile
   ```

2. Exécuter :

   ```
   source ~/.profile
   ```

Pour désactiver la télémétrie sous macOS

1. Exécuter :

   ```
   echo "export SAM_CLI_TELEMETRY=0" >>~/.profile
   ```

2. Exécuter :

```
source ~/.profile
```

Pour désactiver la télémétrie sous Windows

Vous pouvez définir temporairement la variable d'environnement pendant toute la durée de vie de la fenêtre du terminal à l'aide de la commande suivante :

Si vous utilisez l'invite de commande :

```
set SAM_CLI_TELEMETRY=0
```

Si vous utilisez PowerShell :

```
$env:SAM_CLI_TELEMETRY=0
```

Pour définir définitivement la variable d'environnement soit dans l'invite de commande PowerShell, soit à l'aide de la commande suivante :

```
setx SAM_CLI_TELEMETRY 0
```

> **Note**
>
> Les modifications n'entreront en vigueur qu'après la fermeture et la réouverture du terminal.

Type d’informations à collecter

- Informations d'utilisation : les commandes et sous-commandes génériques exécutées par les clients.
- Erreurs et informations de diagnostic : le statut et la durée des commandes exécutées par les clients, y compris les codes de sortie, les noms d'exception internes et les échecs lors de la connexion à Docker.
- Informations sur le système et l'environnement : version de Python, système d'exploitation (Windows, Linux ou macOS), environnement dans lequel il AWS SAMCLI s'exécute (par exemple

AWS CodeBuild, un kit d'outils AWS IDE ou un terminal) et valeurs de hachage des attributs d'utilisation.

En savoir plus

Les données de télémétrie AWS SAMCLI collectées sont conformes aux politiques de confidentialité des AWS données. Pour plus d'informations, consultez les ressources suivantes :

- AWS Modalités du service
- Questions fréquentes (FAQ) relatives à la confidentialité des données

Résolution des problèmes de la CLI AWS SAM

Cette section explique comment résoudre les messages d'erreur lors de l'utilisation, de l'installation et de la gestion de l'interface de ligne de AWS Serverless Application Model commande (AWS SAMCLI).

Rubriques

- Résolution des problèmes
- Messages d'erreur
- Messages d'avertissement

Résolution des problèmes

Pour obtenir des conseils de résolution des problèmes liés à AWS SAMCLI, voirRésolution des erreurs d'installation.

Messages d'erreur

Erreur Curl : « curl : (6) Impossible de résoudre : … »

Lorsque vous essayez d'appeler le point de terminaison API Gateway, vous voyez l'erreur suivante :

```
curl: (6) Could not resolve: endpointdomain (Domain name not found)
```

Cela signifie que vous avez tenté d'envoyer une demande à un domaine non valide. Cela peut se produire si votre application sans serveur n'a pas réussi à se déployer, ou si vous avez une faute de

frappe dans votre commande curl. Vérifiez que l'application a été déployée avec succès à l'aide de la CloudFormation console ou du AWS CLI, et vérifiez que votre curl commande est correcte.

Erreur : impossible de trouver les informations exactes sur la ressource avec le nom de pile donné

Lorsque vous exécutez la commande `sam remote invoke` sur une application qui contient une seule ressource de fonction Lambda, l'erreur suivante s'affiche :

```
Error: Can't find exact resource information with given <stack-name>. Please provide
 full resource ARN or --stack-name to resolve the ambiguity.
```

Cause possible : vous n'avez pas fourni l'option **`--stack-name`**.

Si aucun ARN de fonction n'est fourni en tant qu'argument, la `sam remote invoke` commande nécessite que l'option `--stack-name` soit fournie.

Solution : fournissez l'option**`--stack-name`**.

Voici un exemple :

```
$ 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\"}"}%
```

Erreur : impossible de trouver les informations sur les ressources à partir du nom de la pile

Lorsque vous exécutez la commande `sam remote invoke` et que vous transmettez un ARN de fonction Lambda en tant qu'argument, l'erreur suivante s'affiche :

```
Error: Can't find resource information from stack name (<stack-name>) and resource id
 (<function-id>)
```

Cause possible : la valeur du nom de la pile est définie dans votre fichier **`samconfig.toml`**.

La CLI AWS SAM vérifie d'abord si votre fichier `samconfig.toml` contient un nom de pile. S'il est spécifié, l'argument est transmis en tant que valeur d'identifiant logique.

Solution : transmettez plutôt l'ID logique de la fonction.

Vous pouvez transmettre l'ID logique de la fonction en tant qu'argument au lieu de l'ARN de la fonction.

Solution : supprimez la valeur du nom de la pile de votre fichier de configuration.

Vous pouvez supprimer la valeur du nom de la pile de votre fichier de configuration. Cela empêche la CLI AWS SAM de transmettre l'ARN de votre fonction en tant que valeur d'identifiant logique.

Exécutez `sam build` après avoir modifié votre fichier de configuration.

Erreur : Impossible de créer les ressources gérées : impossible de localiser les informations d'identification

Lorsque vous exécutez la commande sam deploy, vous voyez l'erreur suivante :

```
Error: Failed to create managed resources: Unable to locate credentials
```

Cela signifie que vous n'avez pas configuré AWS d'informations d'identification permettant de AWS SAMCLI passer des appels AWS de service. Pour résoudre ce problème, vous devez configurer les AWS informations d'identification. Pour de plus amples informations, veuillez consulter Configuration des AWS informations d'identification.

Erreur : FileNotFoundError sous Windows

Lorsque vous exécutez des commandes AWS SAMCLI sous Windows, le message d'erreur suivant peut s'afficher :

```
Error: FileNotFoundError
```

Cause possible : ils AWS SAMCLI peuvent interagir avec des chemins de fichiers qui dépassent la limite de chemin maximale de Windows.

Solution : pour résoudre ce problème, le nouveau comportement des longs chemins doit être activé. Pour ce faire, consultez la section Activer les longs chemins dans Windows 10, version 1607 et versions ultérieures dans la documentation de développement d'applications Microsoft Windows.

Erreur : résolveur de dépendances de pip…

Exemple de texte d'erreur :

```
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.
```

Cause possible : si vous utilisez pip pour l'installation de packages, les dépendances entre les packages peuvent entrer en conflit.

Chaque version du package `aws-sam-cli` dépend d'une version du package `aws-sam-translator`. Par exemple, `aws-sam-cli` v1.58.0 peut dépendre de v1.51.0 `aws-sam-translator`.

Si vous installez la CLI CLI AWS SAMà l'aide de pip, puis installez un autre package dépendant d'une version plus récente de `aws-sam-translator`, les événements suivants se produiront :

- La version la plus récente de `aws-sam-translator` sera installée.
- La version actuelle de `aws-sam-cli` et la version plus récente de `aws-sam-translator` peuvent ne pas être compatibles.
- Lorsque vous utilisez le AWS SAMCLI, l'erreur du résolveur de dépendances se produit.

Solutions :

1. Utilisez le programme d'installation de packages natif de la CLI AWS SAM.
   a. Désinstallez la CLI AWS SAMà l'aide de pip. Pour obtenir des instructions, veuillez consulter Désinstallation de la CLI AWS SAM.
   b. Installez la CLI AWS SAMà l'aide du programme d'installation de packages natif. Pour obtenir des instructions, veuillez consulter Installer la CLI AWS SAM.

c. Si nécessaire, mettez à niveau la CLI AWS SAMà l'aide du programme d'installation de packages natif. Pour obtenir des instructions, veuillez consulter Mise à niveau de la CLI AWS SAM en cours.

2. Si vous devez utiliser pip, nous vous recommandons d'installer la CLI AWS SAM dans un environnement virtuel. Cela garantit un environnement d'installation propre et un environnement isolé en cas d'erreurs. Pour obtenir des instructions, veuillez consulter Installation de la CLI AWS SAM dans un environnement virtuel à l'aide de pip.

Erreur : aucune commande « télécommande » de ce type

Lorsque vous exécutez la commande `sam remote invoke`, vous voyez l'erreur suivante :

```
$ 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'.
```

Cause possible : votre version de la CLI AWS SAM est obsolète.

La AWS SAMCLI `sam remote invoke` commande a été publiée avec AWS SAMCLI la version 1.88.0. Vous pouvez vérifier votre version en exécutant la commande `sam --version`.

Solution : mettez à niveau votre CLI AWS SAM vers la dernière version.

Pour obtenir des instructions, veuillez consulter Mise à niveau de la CLI AWS SAM en cours.

Erreur : l'exécution de projets AWS SAM en local nécessiteDocker. L'avez-vous installé ?

Lorsque vous exécutez la commande sam local start-api, vous voyez l'erreur suivante :

```
Error: Running AWS SAM projects locally requires Docker. Have you got it installed?
```

Cela signifie que vous n'avez pas correctement installé Docker. Docker est requis pour tester votre application localement. Pour résoudre ce problème, suivez les instructions d'installation de Docker pour votre hôte de développement. Pour de plus amples informations, veuillez consulter Installation de Docker.

Erreur : Contraintes de sécurité non satisfaites

Lors de l'exécution de sam deploy --guided, vous devrez répondre à la question `Function may not have authorization defined, Is this okay? [y/N]`. Si vous répondez à cette question par non **N** (réponse par défaut), vous voyez apparaître l'erreur suivante :

```
Error: Security Constraints Not Satisfied
```

L'invite vous informe que l'application que vous êtes sur le point de déployer peut comporter une API Amazon API Gateway accessible au public configurée sans autorisation. En répondant **N** à cette question, vous dites que ce n'est pas OK.

Pour résoudre le problème, vous disposez des options suivantes :

- Configurez votre application avec l'autorisation. Pour plus d'informations sur la configuration de l'autorisation, consultez Contrôlez l'accès aux API avec votre AWS SAM modèle.
- Si votre intention est de disposer d'un point de terminaison d'API accessible au public sans autorisation, redémarrez votre déploiement et répondez à cette question par **Y** pour indiquer que vous êtes d'accord avec le déploiement.

message : jeton d'authentification manquant

Lorsque vous essayez d'appeler le point de terminaison API Gateway, vous voyez l'erreur suivante :

```
{"message":"Missing Authentication Token"}
```

Cela signifie que vous avez tenté d'envoyer une demande au domaine correct, mais que l'URI n'est pas reconnaissable. Pour résoudre ce problème, vérifiez l'URL complète et mettez à jour la commande curl avec l'URL correcte.

Messages d'avertissement

Avertissement :... AWS ne maintiendra plus le Homebrew programme d'installation pendant AWS SAM ...

Lors de l'installation de la CLI AWS SAMà l'aide de Homebrew, le message d'avertissement suivant s'affiche :

```
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 ...
```

Cause potentielle : AWS ne plus maintenir le Homebrew support.

À compter de septembre 2023, l'Homebrewinstallateur ne AWS sera plus maintenu pour le AWS SAMCLI.

Solution : utilisez une méthode d'installation AWS prise en charge.

- Vous trouverez les méthodes d'installation AWS prises en charge à l'adresseInstaller la CLI AWS SAM.

Solution : pour continuer à utiliser Homebrew, utilisez le programme d'installation géré par la communauté.

- Vous pouvez utiliser le programme d'installation Homebrew géré par la communauté à votre discrétion. Pour obtenir des instructions, veuillez consulter Gestion de la CLI AWS SAM avec Homebrew.

AWS SAM référence du connecteur

Cette section contient des informations de référence pour le type de ressource du connecteur AWS Serverless Application Model (AWS SAM). Pour obtenir une présentation des connecteurs, veuillez consulter Gestion des autorisations relatives aux ressources à l'aide de AWS SAM connecteurs.

Types de ressources source et de destination pris en charge pour les connecteurs

Le type de ressource `AWS::Serverless::Connector` prend en charge un certain nombre de connexions entre les ressources source et de destination. Lorsque vous configurez des connecteurs dans votre AWS SAM modèle, utilisez le tableau suivant pour référencer les connexions prises en charge et les propriétés qui doivent être définies pour chaque type de ressource source et de destination. Pour plus d'informations sur la configuration de connecteurs dans votre modèle, consultez AWS::Serverless::Connector.

Pour les ressources source et de destination, lorsqu'elles sont définies dans le même modèle, utilisez la propriété `Id`. Sinon, un élément `Qualifier` peut être ajouté pour réduire la portée de la ressource

que vous avez définie. Quand les ressources ne se trouvent pas dans le même modèle, utilisez une combinaison d'autres propriétés.

Pour demander de nouvelles connexions, soumettez un nouveau problème dans le serverless-application-model AWS GitHubréférentiel.

Source type (Type de source)	Type de destination	Permissions	Propriétés de source	Propriétés de destination
`AWS::ApiGateway::RestApi`	`AWS::Lambda::Function`	`Write`	`Id` ou `Qualifier` , `ResourceId` et `Type`	`Id` ou `Arn` et `Type`
`AWS::ApiGateway::RestApi`	`AWS::Serverless::Function`	`Write`	`Id` ou `Qualifier` , `ResourceId` et `Type`	`Id` ou `Arn` et `Type`
`AWS::ApiGatewayV2::Api`	`AWS::Lambda::Function`	`Write`	`Id` ou `Qualifier` , `ResourceId` et `Type`	`Id` ou `Arn` et `Type`
`AWS::ApiGatewayV2::Api`	`AWS::Serverless::Function`	`Write`	`Id` ou `Qualifier` , `ResourceId` et `Type`	`Id` ou `Arn` et `Type`
`AWS::AppSync::DataSource`	`AWS::DynamoDB::Table`	`Read`	`Id` ou `RoleName` et `Type`	`Id` ou `Arn` et `Type`
`AWS::AppSync::DataSource`	`AWS::DynamoDB::Table`	`Write`	`Id` ou `RoleName` et `Type`	`Id` ou `Arn` et `Type`

Source type (Type de source)	Type de destination	Permissions	Propriétés de source	Propriétés de destination
AWS::AppSync::DataSource	AWS::Events::EventBus	Write	Id ou RoleName et Type	Id ou Arn et Type
AWS::AppSync::DataSource	AWS::Lambda::Function	Write	Id ou RoleName et Type	Id ou Arn et Type
AWS::AppSync::DataSource	AWS::Serverless::Function	Write	Id ou RoleName et Type	Id ou Arn et Type
AWS::AppSync::DataSource	AWS::Serverless::SimpleTable	Read	Id ou RoleName et Type	Id ou Arn et Type
AWS::AppSync::DataSource	AWS::Serverless::SimpleTable	Write	Id ou RoleName et Type	Id ou Arn et Type
AWS::AppSync::GraphQLApi	AWS::Lambda::Function	Write	Id ou ResourceId et Type	Id ou Arn et Type
AWS::AppSync::GraphQLApi	AWS::Serverless::Function	Write	Id ou ResourceId et Type	Id ou Arn et Type
AWS::DynamoDB::Table	AWS::Lambda::Function	Read	Id ou Arn et Type	Id ou RoleName et Type
AWS::DynamoDB::Table	AWS::Serverless::Function	Read	Id ou Arn et Type	Id ou RoleName et Type

Source type (Type de source)	Type de destination	Permissions	Propriétés de source	Propriétés de destination
`AWS::Events::Rule`	`AWS::Events::EventBus`	`Write`	`Id` ou `RoleName` et `Type`	`Id` ou `Arn` et `Type`
`AWS::Events::Rule`	`AWS::Lambda::Function`	`Write`	`Id` ou `Arn` et `Type`	`Id` ou `Arn` et `Type`
`AWS::Events::Rule`	`AWS::Serverless::Function`	`Write`	`Id` ou `Arn` et `Type`	`Id` ou `Arn` et `Type`
`AWS::Events::Rule`	`AWS::Serverless::StateMachine`	`Write`	`Id` ou `RoleName` et `Type`	`Id` ou `Arn` et `Type`
`AWS::Events::Rule`	`AWS::SNS::Topic`	`Write`	`Id` ou `Arn` et `Type`	`Id` ou `Arn` et `Type`
`AWS::Events::Rule`	`AWS::SQS::Queue`	`Write`	`Id` ou `Arn` et `Type`	`Id` ou `Arn`, `QueueUrl` et `Type`
`AWS::Events::Rule`	`AWS::StepFunctions::StateMachine`	`Write`	`Id` ou `RoleName` et `Type`	`Id` ou `Arn` et `Type`
`AWS::Lambda::Function`	`AWS::DynamoDB::Table`	`Read`, `Write`	`Id` ou `RoleName` et `Type`	`Id` ou `Arn` et `Type`

Source type (Type de source)	Type de destination	Permissions	Propriétés de source	Propriétés de destination
AWS::Lambda::Function	AWS::Events::EventBus	Write	Id ou RoleName et Type	Id ou Arn et Type
AWS::Lambda::Function	AWS::Lambda::Function	Write	Id ou RoleName et Type	Id ou Arn et Type
AWS::Lambda::Function	AWS::Location::PlaceIndex	Read	Id ou RoleName et Type	Id ou Arn et Type
AWS::Lambda::Function	AWS::S3::Bucket	Read, Write	Id ou RoleName et Type	Id ou Arn et Type
AWS::Lambda::Function	AWS::Serverless::Function	Write	Id ou RoleName et Type	Id ou Arn et Type
AWS::Lambda::Function	AWS::Serverless::SimpleTable	Read, Write	Id ou RoleName et Type	Id ou Arn et Type
AWS::Lambda::Function	AWS::Serverless::StateMachine	Read, Write	Id ou RoleName et Type	Id ou Arn, Name et Type
AWS::Lambda::Function	AWS::SNS::Topic	Write	Id ou RoleName et Type	Id ou Arn et Type

Source type (Type de source)	Type de destination	Permissions	Propriétés de source	Propriétés de destination
AWS::Lambda::Function	AWS::SQS::Queue	Read, Write	Id ou RoleName et Type	Id ou Arn et Type
AWS::Lambda::Function	AWS::StepFunctions::StateMachine	Read, Write	Id ou RoleName et Type	Id ou Arn, Name et Type
AWS::S3::Bucket	AWS::Lambda::Function	Write	Id ou Arn et Type	Id ou Arn et Type
AWS::S3::Bucket	AWS::Serverless::Function	Write	Id ou Arn et Type	Id ou Arn et Type
AWS::Serverless::Api	AWS::Lambda::Function	Write	Id ou Qualifier , ResourceId et Type	Id ou Arn et Type
AWS::Serverless::Api	AWS::Serverless::Function	Write	Id ou Qualifier , ResourceId et Type	Id ou Arn et Type
AWS::Serverless::Function	AWS::DynamoDB::Table	Read, Write	Id ou RoleName et Type	Id ou Arn et Type
AWS::Serverless::Function	AWS::Events::EventBus	Write	Id ou RoleName et Type	Id ou Arn et Type

Source type (Type de source)	Type de destination	Permissions	Propriétés de source	Propriétés de destination
AWS::Serverless::Function	AWS::Lambda::Function	Write	Id ou RoleName et Type	Id ou Arn et Type
AWS::Serverless::Function	AWS::S3::Bucket	Read, Write	Id ou RoleName et Type	Id ou Arn et Type
AWS::Serverless::Function	AWS::Serverless::Function	Write	Id ou RoleName et Type	Id ou Arn et Type
AWS::Serverless::Function	AWS::Serverless::SimpleTable	Read, Write	Id ou RoleName et Type	Id ou Arn et Type
AWS::Serverless::Function	AWS::Serverless::StateMachine	Read, Write	Id ou RoleName et Type	Id ou Arn, Name et Type
AWS::Serverless::Function	AWS::SNS::Topic	Write	Id ou RoleName et Type	Id ou Arn et Type
AWS::Serverless::Function	AWS::SQS::Queue	Read, Write	Id ou RoleName et Type	Id ou Arn et Type
AWS::Serverless::Function	AWS::StepFunctions::StateMachine	Read, Write	Id ou RoleName et Type	Id ou Arn, Name et Type

Source type (Type de source)	Type de destination	Permissions	Propriétés de source	Propriétés de destination
AWS::Serverless::HttpApi	AWS::Lambda::Function	Write	Id ou Qualifier , ResourceId et Type	Id ou Arn et Type
AWS::Serverless::HttpApi	AWS::Serverless::Function	Write	Id ou Qualifier , ResourceId et Type	Id ou Arn et Type
AWS::Serverless::SimpleTable	AWS::Lambda::Function	Read	Id ou Arn et Type	Id ou RoleName et Type
AWS::Serverless::SimpleTable	AWS::Serverless::Function	Read	Id ou Arn et Type	Id ou RoleName et Type
AWS::Serverless::StateMachine	AWS::DynamoDB::Table	Read, Write	Id ou RoleName et Type	Id ou Arn et Type
AWS::Serverless::StateMachine	AWS::Events::EventBus	Write	Id ou RoleName et Type	Id ou Arn et Type
AWS::Serverless::StateMachine	AWS::Lambda::Function	Write	Id ou RoleName et Type	Id ou Arn et Type

Source type (Type de source)	Type de destination	Permissions	Propriétés de source	Propriétés de destination
AWS::Serverless::StateMachine	AWS::S3::Bucket	Read, Write	Id ou RoleName et Type	Id ou Arn et Type
AWS::Serverless::StateMachine	AWS::Serverless::Function	Write	Id ou RoleName et Type	Id ou Arn et Type
AWS::Serverless::StateMachine	AWS::Serverless::SimpleTable	Read, Write	Id ou RoleName et Type	Id ou Arn et Type
AWS::Serverless::StateMachine	AWS::Serverless::StateMachine	Read, Write	Id ou RoleName et Type	Id ou Arn, Name et Type
AWS::Serverless::StateMachine	AWS::SNS::Topic	Write	Id ou RoleName et Type	Id ou Arn et Type
AWS::Serverless::StateMachine	AWS::SQS::Queue	Write	Id ou RoleName et Type	Id ou Arn et Type
AWS::Serverless::StateMachine	AWS::StepFunctions::StateMachine	Read, Write	Id ou RoleName et Type	Id ou Arn, Name et Type

Source type (Type de source)	Type de destination	Permissions	Propriétés de source	Propriétés de destination
AWS::SNS::Topic	AWS::Lambda::Function	Write	Id ou Arn et Type	Id ou Arn et Type
AWS::SNS::Topic	AWS::Serverless::Function	Write	Id ou Arn et Type	Id ou Arn et Type
AWS::SNS::Topic	AWS::SQS::Queue	Write	Id ou Arn et Type	Id ou Arn, QueueUrl et Type
AWS::SQS::Queue	AWS::Lambda::Function	Read, Write	Id ou Arn et Type	Id ou RoleName et Type
AWS::SQS::Queue	AWS::Serverless::Function	Read, Write	Id ou Arn et Type	Id ou RoleName et Type
AWS::StepFunctions::StateMachine	AWS::DynamoDB::Table	Read, Write	Id ou RoleName et Type	Id ou Arn et Type
AWS::StepFunctions::StateMachine	AWS::Events::EventBus	Write	Id ou RoleName et Type	Id ou Arn et Type
AWS::StepFunctions::StateMachine	AWS::Lambda::Function	Write	Id ou RoleName et Type	Id ou Arn et Type

Source type (Type de source)	Type de destination	Permissions	Propriétés de source	Propriétés de destination
AWS::StepFunctions::StateMachine	AWS::S3::Bucket	Read, Write	Id ou RoleName et Type	Id ou Arn et Type
AWS::StepFunctions::StateMachine	AWS::Serverless::Function	Write	Id ou RoleName et Type	Id ou Arn et Type
AWS::StepFunctions::StateMachine	AWS::Serverless::SimpleTable	Read, Write	Id ou RoleName et Type	Id ou Arn et Type
AWS::StepFunctions::StateMachine	AWS::Serverless::StateMachine	Read, Write	Id ou RoleName et Type	Id ou Arn, Name et Type
AWS::StepFunctions::StateMachine	AWS::SNS::Topic	Write	Id ou RoleName et Type	Id ou Arn et Type
AWS::StepFunctions::StateMachine	AWS::SQS::Queue	Write	Id ou RoleName et Type	Id ou Arn et Type
AWS::StepFunctions::StateMachine	AWS::StepFunctions::StateMachine	Read, Write	Id ou RoleName et Type	Id ou Arn, Name et Type

Politiques IAM créées par les connecteurs

Cette section décrit les politiques Gestion des identités et des accès AWS (IAM) créées AWS SAM lors de l'utilisation de connecteurs.

`AWS::DynamoDB::Table` sur `AWS::Lambda::Function`

Type de politique

La politique gérée par le client jointe au rôle de `AWS::Lambda::Function`.

Catégories d'accès

`Read`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:DescribeStream",
        "dynamodb:GetRecords",
        "dynamodb:GetShardIterator",
        "dynamodb:ListStreams"
      ],
      "Resource": [
        "%{Source.Arn}/stream/*"
      ]
    }
  ]
}
```

`AWS::Events::Rule` sur `AWS::SNS::Topic`

Type de politique

`AWS::SNS::TopicPolicy` jointe à `AWS::SNS::Topic`.

Catégories d'accès

`Write`

```
{
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "events.amazonaws.com"
      },
      "Resource": "%{Destination.Arn}",
      "Action": "sns:Publish",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "%{Source.Arn}"
        }
      }
    }
  ]
}
```

`AWS::Events::Rule` sur `AWS::Events::EventBus`

Type de politique

La politique gérée par le client jointe au rôle de `AWS::Events::Rule`.

Catégories d'accès

`Write`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:PutEvents"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

`AWS::Events::Rule` sur `AWS::StepFunctions::StateMachine`

Type de politique

La politique gérée par le client jointe au rôle de `AWS::Events::Rule`.

Catégories d'accès

`Write`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:StartExecution"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

`AWS::Events::Rule` sur `AWS::Lambda::Function`

Type de politique

`AWS::Lambda::Permission` jointe à `AWS::Lambda::Function`.

Catégories d'accès

`Write`

```
{
  "Action": "lambda:InvokeFunction",
  "Principal": "events.amazonaws.com",
  "SourceArn": "%{Source.Arn}"
}
```

`AWS::Events::Rule` sur `AWS::SQS::Queue`

Type de politique

`AWS::SQS::QueuePolicy` jointe à `AWS::SQS::Queue`.

Catégories d'accès

`Write`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "events.amazonaws.com"
      },
      "Resource": "%{Destination.Arn}",
      "Action": "sqs:SendMessage",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "%{Source.Arn}"
        }
      }
    }
  ]
}
```

`AWS::Lambda::Function` sur `AWS::Lambda::Function`

Type de politique

La politique gérée par le client jointe au rôle de `AWS::Lambda::Function`.

Catégories d'accès

`Write`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeAsync",
        "lambda:InvokeFunction"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
```

```
}
```

`AWS::Lambda::Function` sur `AWS::S3::Bucket`

Type de politique

La politique gérée par le client jointe au rôle de `AWS::Lambda::Function`.

Catégories d'accès

`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` sur `AWS::DynamoDB::Table`

Type de politique

La politique gérée par le client jointe au rôle de `AWS::Lambda::Function`.

Catégories d'accès

`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` sur `AWS::SQS::Queue`

Type de politique

La politique gérée par le client jointe au rôle de `AWS::Lambda::Function`.

Catégories d'accès

`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` sur `AWS::SNS::Topic`

Type de politique

La politique gérée par le client jointe au rôle de `AWS::Lambda::Function`.

Catégories d'accès

`Write`

```
{
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

`AWS::Lambda::Function` sur `AWS::StepFunctions::StateMachine`

Type de politique

La politique gérée par le client jointe au rôle de `AWS::Lambda::Function`.

Catégories d'accès

`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` sur `AWS::Events::EventBus`

Type de politique

La politique gérée par le client jointe au rôle de `AWS::Lambda::Function`.

Catégories d'accès

`Write`

```
{
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Action": [
        "events:PutEvents"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

`AWS::Lambda::Function` sur `AWS::Location::PlaceIndex`

Type de politique

La politique gérée par le client jointe au rôle de `AWS::Lambda::Function`.

Catégories d'accès

`Read`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "geo:DescribePlaceIndex",
        "geo:GetPlace",
        "geo:SearchPlaceIndexForPosition",
        "geo:SearchPlaceIndexForSuggestions",
        "geo:SearchPlaceIndexForText"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

`AWS::ApiGatewayV2::Api` sur `AWS::Lambda::Function`

Type de politique

AWS::Lambda::Permission jointe à AWS::Lambda::Function.

Catégories d'accès

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 sur AWS::Lambda::Function

Type de politique

AWS::Lambda::Permission jointe à AWS::Lambda::Function.

Catégories d'accès

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 sur AWS::SQS::Queue

Type de politique

AWS::SQS::QueuePolicy jointe à AWS::SQS::Queue.

Catégories d'accès

Write

```
{
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.amazonaws.com"
      },
      "Resource": "%{Destination.Arn}",
      "Action": "sqs:SendMessage",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "%{Source.Arn}"
        }
      }
    }
  ]
}
```

`AWS::SNS::Topic` sur `AWS::Lambda::Function`

Type de politique

`AWS::Lambda::Permission` jointe à `AWS::Lambda::Function`.

Catégories d'accès

`Write`

```
{
  "Action": "lambda:InvokeFunction",
  "Principal": "sns.amazonaws.com",
  "SourceArn": "%{Source.Arn}"
}
```

`AWS::SQS::Queue` sur `AWS::Lambda::Function`

Type de politique

La politique gérée par le client jointe au rôle de `AWS::Lambda::Function`.

Catégories d'accès

`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` sur `AWS::Lambda::Function`

Type de politique

`AWS::Lambda::Permission` jointe à `AWS::Lambda::Function`.

Catégories d'accès

`Write`

```
{
  "Action": "lambda:InvokeFunction",
  "Principal": "s3.amazonaws.com",
```

```
  "SourceArn": "%{Source.Arn}",
  "SourceAccount": "${AWS::AccountId}"
}
```

`AWS::StepFunctions::StateMachine` sur `AWS::Lambda::Function`

Type de politique

La politique gérée par le client jointe au rôle de `AWS::StepFunctions::StateMachine`.

Catégories d'accès

`Write`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeAsync",
        "lambda:InvokeFunction"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

`AWS::StepFunctions::StateMachine` sur `AWS::SNS::Topic`

Type de politique

La politique gérée par le client jointe au rôle de `AWS::StepFunctions::StateMachine`.

Catégories d'accès

`Write`

```
{
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Action": [
        "sns:Publish"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

`AWS::StepFunctions::StateMachine` sur `AWS::SQS::Queue`

Type de politique

La politique gérée par le client jointe au rôle de `AWS::StepFunctions::StateMachine`.

Catégories d'accès

`Write`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sqs:SendMessage"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

`AWS::StepFunctions::StateMachine` sur `AWS::S3::Bucket`

Type de politique

La politique gérée par le client jointe au rôle de `AWS::StepFunctions::StateMachine`.

Catégories d'accès

`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` sur `AWS::DynamoDB::Table`

Type de politique

La politique gérée par le client jointe au rôle de `AWS::StepFunctions::StateMachine`.

Catégories d'accès

`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` sur `AWS::StepFunctions::StateMachine`

Type de politique

La politique gérée par le client jointe au rôle de `AWS::StepFunctions::StateMachine`.

Catégories d'accès

`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` sur `AWS::Events::EventBus`

Type de politique

La politique gérée par le client jointe au rôle de `AWS::StepFunctions::StateMachine`.

Catégories d'accès

`Write`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:PutEvents"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

`AWS::AppSync::DataSource` sur `AWS::DynamoDB::Table`

Type de politique

La politique gérée par le client jointe au rôle de `AWS::AppSync::DataSource`.

Catégories d'accès

`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` sur `AWS::Lambda::Function`

Type de politique

La politique gérée par le client jointe au rôle de `AWS::AppSync::DataSource`.

Catégories d'accès

`Write`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeAsync",
        "lambda:InvokeFunction"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}:*"
      ]
    }
  ]
}
```

`AWS::AppSync::DataSource` sur `AWS::Events::EventBus`

Type de politique

La politique gérée par le client jointe au rôle de `AWS::AppSync::DataSource`.

Catégories d'accès

`Write`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:PutEvents"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

`AWS::AppSync::GraphQLApi` sur `AWS::Lambda::Function`

Type de politique

`AWS::Lambda::Permission` jointe à `AWS::Lambda::Function`.

Catégories d'accès

`Write`

```
{
  "Action": "lambda:InvokeFunction",
  "Principal": "appsync.amazonaws.com",
  "SourceArn": "arn:${AWS::Partition}:appsync:${AWS::Region}:${AWS::AccountId}:apis/
%{Source.ResourceId}"
}
```

Installation de Docker pour une utilisation avec la CLI AWS SAM

Docker est une application qui exécute des conteneurs sur votre machine. AvecDocker, AWS SAM peut fournir un environnement local similaire à AWS Lambda un conteneur pour créer, tester et déboguer vos applications sans serveur.

> **Note**
>
> Docker n'est requis que pour tester vos applications localement et pour créer des packages de déploiement à l'aide de l'option `--use-container`.

Rubriques

- Installation deDocker
- Étapes suivantes

Installation deDocker

Pour installer Docker sur votre système d'exploitation, suivez ces instructions.

Linux

Docker est disponible sur de nombreux systèmes d'exploitation différents, y compris les distributions Linux les plus modernes, telles que CentOS, Debian et Ubuntu. Pour plus d'informations sur

l'installation de Docker sur votre système d'exploitation particulier, veuillez consulter la section Obtenir Docker sur le site web Docker Docs (langue française non garantie).

Pour installer Docker sur Amazon Linux 2 ou Amazon Linux 2023

1. Mettez à jour les packages installés et le cache du package sur votre instance.

```
$ sudo yum update -y
```

2. Installez la version la plus récente du package Docker Community Edition.

   - Pour Amazon Linux 2, exécutez la commande suivante :

```
$ sudo amazon-linux-extras install docker
```

   - Pour Amazon Linux 2023, exécutez la commande suivante :

```
$ sudo yum install -y docker
```

3. Lancez le service Docker.

```
$ sudo service docker start
```

4. Ajoutez `ec2-user` au groupe `docker` afin de pouvoir exécuter les commandes Docker sans utiliser sudo.

```
$ sudo usermod -a -G docker ec2-user
```

5. Déconnectez-vous et reconnectez-vous pour récupérer les nouvelles autorisations de groupe `docker`. Pour cela, fermez votre fenêtre de terminal SSH actuelle et en vous reconnectant à votre instance dans une nouvelle fenêtre. Votre nouvelle session SSH devrait disposer des autorisations de groupe `docker` appropriées.
6. Vérifiez que `ec2-user` puisse exécuter les commandes Docker sans utiliser sudo.

```
$ docker ps
```

   La sortie suivante doit s'afficher. Elle confirme que Docker est installé et en cours d'exécution :

```
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
```

> **Note**
>
> Sous Linux, pour créer et exécuter des fonctions Lambda avec une architecture de jeu d'instructions différente de celle de votre machine hôte, la configuration de Docker nécessite quelques étapes supplémentaires. Par exemple, pour exécuter des fonctions `arm64` sur une machine `x86_64`, vous pouvez exécuter la commande suivante afin de configurer le démon Docker : `docker run --rm --privileged multiarch/qemu-user-static --reset -p yes`.

Si vous rencontrez des problèmes lors de l'installation de Docker, veuillez consulter la section Résolution des erreurs d'installation. Ou, consultez la section Dépannage de Étapes de post-installation pour Linux sur le site web Docker Docs.

macOS

> **Note**
>
> Docker Desktop est officiellement pris en charge, mais à partir de la version 1.47.0 de la CLI AWS SAM, vous pouvez utiliser des alternatives tant qu'elles utilisent le moteur d'exécution Docker.

1. Installer Docker

   La CLI AWS SAM prend en charge Docker s'exécutant sur macOS Sierra 10.12 ou version ultérieure. Pour découvrir comment installer Docker, veuillez consulter la section Installer Docker Desktop pour Mac sur le site web Docker Docs (langue française non garantie).
2. Configurer vos lecteurs partagés

   Le AWS SAMCLI nécessite que le répertoire du projet, ou tout répertoire parent, soit répertorié sur un lecteur partagé. Cela peut nécessiter le partage de fichiers. Pour plus d'informations, consultez la rubrique de résolution des problèmes liés au montage de volumes nécessitant le partage de fichiers dans la section Documents Docker.
3. Vérifier l'installation

   Une fois Docker installé, vérifiez qu'il fonctionne. Confirmez également que vous pouvez exécuter des commandes Docker à partir de la ligne de commande (par exemple, `docker ps`).

Vous n'avez pas besoin d'installer, de récupérer ou de tirer un conteneur, la CLI AWS SAM le fait automatiquement selon les besoins.

Si vous rencontrez des problèmes lors de l'installation de Docker, veuillez consulter la section Dépannage et diagnostic du site web Docker Docs afin d'obtenir davantage de conseils de dépannage (langue française non garantie).

Windows

> **Note**
>
> AWS SAM prend officiellement en charge Docker Desktop. Toutefois, à partir de la version 1.47.0 de la CLI AWS SAM, vous pouvez utiliser des alternatives tant qu'elles utilisent le moteur d'exécution Docker.

1. Installer Docker.

   Docker Desktop prend en charge la version la plus récente du système d'exploitation Windows. Pour les versions héritées de Windows, la boîte à outils Docker est disponible. Choisissez votre version de Windows pour les étapes d'installation correctes de Docker :

   - Pour installer Docker sous Windows 10, veuillez consulter la section Installer Docker Desktop pour Windows sur le site web Docker Docs (langue française non garantie).
   - Pour effectuer l'installation Docker pour les versions antérieures de Windows, voir The Docker Toolbox dans le référentiel Docker Toolbox. GitHub

2. Configurez vos lecteurs partagés.

   Le AWS SAMCLI nécessite que le répertoire du projet, ou tout répertoire parent, soit répertorié sur un lecteur partagé. Dans certains cas, vous devez partager votre disque pour que Docker fonctionne correctement.

3. Vérifiez l’installation.

   Une fois Docker installé, vérifiez qu'il fonctionne. Confirmez également que vous pouvez exécuter des commandes Docker à partir de la ligne de commande (par exemple, docker ps). Vous n'avez pas besoin d'installer, de récupérer ou de tirer un conteneur, la CLI AWS SAM le fait automatiquement selon les besoins.

Si vous rencontrez des problèmes lors de l'installation de Docker, veuillez consulter la section Dépannage et diagnostic du site web Docker Docs afin d'obtenir davantage de conseils de dépannage (langue française non garantie).

Étapes suivantes

Pour savoir comment installer le AWS SAMCLI, voirInstaller la CLI AWS SAM.

Installation Finch à utiliser avec la AWS SAM CLI

Finch AWS est un outil de développement de conteneurs qui fournit une alternative particulièrement avantageuse sur les systèmes macOS et Linux. Docker Pour plus d'informations, voir Finch sur GitHub. AvecFinch, AWS SAM peut fournir un environnement local similaire à AWS Lambda un conteneur pour créer, tester et déboguer vos applications sans serveur.

> **Note**
>
> Vous devez Finch tester vos applications localement, créer des packages de déploiement à l'aide de l'`--use-container`option et créer des fonctions d'image OCI. À partir de la version 1.145.0 de la AWS SAM CLI, la AWS SAM CLI prend Finch en charge l'exécution de conteneurs alternative à. Docker

Rubriques

- Installation deFinch
- Configuration à l'Finchaide de la AWS SAM CLI
- Vérification de l'installation
- Arrêt d'Finch
- Résolution des problèmes
- Étapes suivantes

Installation deFinch

Pour installer Finch sur votre système d'exploitation, suivez ces instructions.

Conditions préalables

> **Note**
>
> Finchl'intégration dans la AWS SAM CLI est disponible sur les systèmes macOS et Linux. Pour les utilisateurs de Windows, pensez à utiliser une machine virtuelle ou un environnement Linux pour les Finch fonctionnalités.

Versions de macOS :

- macOS 15 Séquoia
- macOS 14 Sonoma

> **Note**
>
> Finchprend en charge les deux dernières versions de macOS.

Linux :

- Noyau Linux v4.x+

Installation de Finch

macOS

Pour installer Finch sur MacOS

1. Installation Finch à l'aide de la `brew` CLI. Si vous y êtes invité, entrez votre mot de passe macOS.

   ```
   $ brew install finch
   ```

2. Vérifiez la Finch version installée.

   ```
   $ finch --version
   ```

3. Initialisez la machine Finch virtuelle (première configuration).

```
$ finch vm init
```

4. Démarrez la machine Finch virtuelle.

```
$ finch vm start
```

5. Vérifiez l'état de la machine Finch virtuelle.

```
$ finch vm status
```

6. Vérifiez que cela Finch fonctionne correctement.

```
$ finch --info
$ finch ps
```

Pour plus d'informations, consultez la section Installation de Finch sur macOS.

Linux

Vous pouvez effectuer l'installation Finch sur Amazon Linux, Ubuntu ou en utilisant une méthode d'installation générique. Pour plus d'informations, consultez Installation de Finch sous Linux.

Configuration à l'Finchaide de la AWS SAM CLI

Après l'installationFinch, la AWS SAM CLI détecte et utilise automatiquement Finch comme environnement d'exécution du conteneur lorsqu'elle Docker ne s'exécute pas.

> **Note**
>
> Finchsert de solution de repli à. Docker Si vous avez Finch installé Docker et exécuté les deux, la AWS SAM CLI hiérarchise Docker par défaut.

Configuration des préférences de l'administrateur (macOS)

Vous pouvez configurer la AWS SAM CLI pour utiliser un environnement d'exécution de conteneur spécifique par défaut en créant un fichier de préférences d'administrateur. Cette préférence à l'échelle du système remplace le comportement de secours par défaut.

Options de configuration disponibles

- Définissez Finch comme environnement d'exécution du conteneur préféré :

```
$ sudo /usr/libexec/PlistBuddy -c "Add :DefaultContainerRuntime string finch" /
Library/Preferences/com.amazon.samcli.plist
```

- Définissez Docker comme environnement d'exécution du conteneur préféré :

```
$ sudo /usr/libexec/PlistBuddy -c "Add :DefaultContainerRuntime string docker" /
Library/Preferences/com.amazon.samcli.plist
```

- Supprimer la préférence (retour au comportement par défaut) :

```
$ sudo /usr/libexec/PlistBuddy -c "Remove :DefaultContainerRuntime" /Library/
Preferences/com.amazon.samcli.plist
```

> **Note**
>
> Lorsqu'aucune préférence d'administrateur n'est définie, la AWS SAM CLI l'utilise Docker par défaut lorsqu'elle est disponible, avec Finch comme solution de secours lorsqu'elle n'Dockerest pas en cours d'exécution.

Différences de montage de répertoires (macOS)

Sur macOS, les commandes AWS SAM locales échouent si vous localisez votre projet en dehors de votre répertoire personnel (~) ou`/Volumes`. Cette conception restreint le montage automatique de répertoires pour des raisons de sécurité à votre insu.

Options d'atténuation

- Option 1 : Ouvrez la configuration Finch dans un éditeur de texte `~/.finch/finch.yaml` et ajoutez le chemin de votre projet sous la `additional_directories` section, puis redémarrez la machine Finch virtuelle. Pour plus d'informations, consultez la documentation Finch.

```
$ finch vm stop
$ finch vm start
```

- Option 2 : déplacez votre projet vers votre répertoire personnel.

Vérification de l'installation

Après l'installation et le démarrageFinch, vérifiez qu'il fonctionne avec la AWS SAM CLI.

Vérification de base

Exécutez la commande suivante pour vérifier que la AWS SAM CLI peut utiliser Finch :

```
$ sam build --use-container
```

Vous devriez voir la AWS SAM CLI Finch utilisée pour créer votre application dans un environnement de conteneur.

> **Note**
>
> Finchsert de solution de repli à. Docker Si elle Docker est en cours d'exécution, la AWS SAM CLI donnera la priorité Docker par défaut.

Les tests Finch comme solution de secours

Pour effectuer un test Finch en tant que solution de secours en Docker cours d'exécution, vous devez d'Dockerabord vous arrêter.

Pour tester la fonctionnalité de Finch secours

1. Arrêtez Docker d'utiliser les commandes adaptées à votre système d'exploitation.

   macOS :

   ```
   $ docker stop $(docker ps -q)
   $ pkill -f "Docker Desktop"
   ```

   DockerLa vérification est arrêtée.

   ```
   $ docker ps
   ```

   Linux :

   ```
   $ docker stop $(docker ps -q)
   ```

```
$ sudo systemctl stop docker
```

DockerLa vérification est arrêtée.

```
$ sudo systemctl status docker
```

2. Exécutez la `sam build --use-container` commande pour vérifier que la AWS SAM CLI est automatiquement utilisée Finch comme solution de secours.

Arrêt d'Finch

Si vous devez vous arrêterFinch, utilisez les commandes suivantes :

macOS :

```
$ finch vm stop
$ sudo pkill -f finch
```

FinchLa vérification est arrêtée.

```
$ finch vm status
```

Linux :

```
$ sudo systemctl stop finch
$ sudo pkill -f finch
```

FinchLa vérification est arrêtée.

```
$ sudo systemctl status finch
```

Résolution des problèmes

Si vous rencontrez des problèmes avec Finch :

- Vérifiez que votre système d'exploitation répond aux exigences requises.
- Vérifiez que vous avez activé la virtualisation sur votre système.
- Pour macOS, vérifiez que vous disposez de suffisamment d'espace disque pour la machine virtuelle.

- Pour Linux, vérifiez que vous disposez des autorisations nécessaires pour exécuter `systemctl` des commandes.
- Si vous rencontrez des échecs de CloudFormation déploiement avec des erreurs « type de support non pris en charge » lorsque vous transférez des images multi-architectures vers Amazon Elastic Container Registry (Amazon ECR), créez un artefact d'image unique au lieu d'utiliser des images multi-architectures. Cela se produit car envoie Finch à la fois l'index de l'image et l'image vers Amazon ECR, alors que seule l'Dockerimage est envoyée.
- Si la construction multiplateforme échoue lorsque l'architecture de la fonction cible ne correspond pas à l'architecture de la machine hôte sous Linux, activez l'émulation multiplateforme en exécutant. `sudo finch run --privileged --rm tonistiigi/binfmt:master --install all`

Pour un dépannage supplémentaire, consultez la documentation Finch ou leRésolution des problèmes de la CLI AWS SAM.

Étapes suivantes

Maintenant que vous l'avez Finch installé, vous pouvez l'utiliser avec la AWS SAM CLI pour :

- Créez des applications sans serveur à l'aide de conteneurs avec`sam build --use-container`. Pour de plus amples informations, veuillez consulter the section called “sam build”.
- Testez les fonctions Lambda localement avec. `sam local invoke` Pour de plus amples informations, veuillez consulter the section called “sam local invoke”.
- Démarrez une API Gateway locale avec`sam local start-api`. Pour de plus amples informations, veuillez consulter the section called “sam local start-api”.
- Démarrez un point de terminaison Lambda local avec. `sam local start-lambda` Pour de plus amples informations, veuillez consulter the section called “sam local start-lambda”.

Pour plus d'informations sur l'utilisation de la AWS SAM CLI avec des conteneurs, consultezConstruit par défaut avec AWS SAM.

Référentiels d'images pour AWS SAM

AWS SAM simplifie les tâches d'intégration continue et de livraison continue (CI/CD) pour les applications sans serveur à l'aide de la création d'images de conteneur. Les images AWS SAM

fournies incluent l'interface de ligne de AWS SAM commande (CLI) et les outils de génération pour un certain nombre d'environnements d' AWS Lambda exécution pris en charge. Cela facilite la création et l'empaquetage d'applications sans serveur à l'aide de la CLI AWS SAM. Vous pouvez utiliser ces images avec des CI/CD systèmes pour automatiser la création et le déploiement d' AWS SAM applications. Pour obtenir des exemples, consultez Déployez avec CI/CD des systèmes et des pipelines.

AWS SAM URIs les images du conteneur de construction sont étiquetées avec la version AWS SAMCLI incluse dans cette image. Si vous spécifiez l'URI non balisé, la dernière version sera utilisée. Par exemple, `public.ecr.aws/sam/build-nodejs20.x` utilise la dernière image. Toutefois, `public.ecr.aws/sam/build-nodejs20.x:1.24.1` utilise l'image contenant la CLI AWS SAM version 1.24.1.

À partir de la version 1.33.0 de, les images AWS SAMCLI, les deux `x86_64` et les images de `arm64` conteneur sont disponibles pour les environnements d'exécution pris en charge. Pour en savoir plus amples, consultez la section Quotas Lambda du Guide du développeur AWS Lambda .

> **Note**
>
> Avant la version 1.22.0 de AWS SAMCLI, DockerHub était le référentiel par défaut à partir duquel l'image du conteneur était AWS SAMCLI extraite. À partir de la version 1.22.0, le référentiel par défaut devient Amazon Elastic Container Registry Public (Amazon ECR Public). Pour extraire une image de conteneur d'un référentiel autre que la valeur par défaut actuelle, vous pouvez utiliser la commande sam build avec l'option --build-image. Les exemples présentés à la fin de cette rubrique montrent comment créer des applications à l'aide d'images de DockerHub référentiel.

Référentiel d'images URIs

Le tableau suivant répertorie les images URIs de conteneur de construction Amazon ECR Public que vous pouvez utiliser pour créer et empaqueter des applications sans serveur. AWS SAM

> **Note**
>
> Amazon ECR Public a remplacé DockerHub à partir de la version 1.22.0 de la CLI AWS SAM. Si vous utilisez une version antérieure du AWS SAMCLI, nous vous recommandons de procéder à une mise à niveau.

Environnement d’exécution	Amazon ECR Public
Temps d'exécution personnalisé (AL2023)	public.ecr. aws/sam/build-provid.al 2023
Environnement d'exécution personnalisé (AL2)	public.ecr. aws/sam/build-provid.al2
Environnement d’exécution personnalisé	public.ecr. aws/sam/build-fourni
Java 25	public.ecr. aws/sam/build-java 25
Java 21	public.ecr. aws/sam/build-java 21
Java 17	public.ecr. aws/sam/build-java17
Java 11	public.ecr. aws/sam/build-java 11
Java 8	public.ecr. aws/sam/build-java 8
.NET 9	public.ecr. aws/sam/build-dotnet9
.NET 8	public.ecr. aws/sam/build-dotnet 8
.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-nodejs 18.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

Environnement d'exécution	Amazon ECR Public
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
Ruby 3.4	public.ecr. aws/sam/build- rubis 3,4
Ruby 3.3	public.ecr. aws/sam/build- rubis 3,3
Ruby 3.2	public.ecr. aws/sam/build- rubis 3,2

Exemples

Les deux exemples de commandes suivants créent des applications à l'aide d'images de conteneur issues du référentiel d'images :

Créez une Node.js 24 application à l'aide d'une image de conteneur extraite d'Amazon ECR :

```
$ sam build --use-container --build-image public.ecr.aws/sam/build-nodejs24.x
```

Créez une ressource fonctionnelle à l'aide de l'image du Python 3.14 conteneur extraite d'Amazon ECR :

```
$ sam build --use-container --build-image Function1=public.ecr.aws/sam/build-python3.14
```

Déploiement progressif d'applications sans serveur avec AWS SAM

AWS Serverless Application Model (AWS SAM) est intégré CodeDeploypour permettre des AWS Lambda déploiements progressifs. À l'aide de quelques lignes de configuration, AWS SAM effectue les opérations suivantes pour vous :

- Déploie de nouvelles versions de la fonction Lambda et crée automatiquement des alias qui pointent vers la nouvelle version.

- Déplace progressivement le trafic client vers la nouvelle version jusqu'à ce que vous estimiez qu'elle fonctionne comme prévu. Si une mise à jour ne fonctionne pas correctement, vous pouvez annuler les modifications.
- Définit des fonctions de test pré- et post-trafic afin de vérifier que le code qui vient d'être déployé est correctement configuré et que l'application fonctionne comme prévu.
- Annule automatiquement le déploiement si des CloudWatch alarmes sont déclenchées.

> **Note**
>
> Si vous activez les déploiements progressifs via votre AWS SAM modèle, une CodeDeploy ressource est automatiquement créée pour vous. Vous pouvez consulter la CodeDeploy ressource directement via le AWS Management Console.

Exemple

L'exemple suivant illustre l'utilisation de CodeDeploy pour transférer progressivement les clients vers votre version récemment déployée de la fonction Lambda :

```
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
```

Ces révisions du AWS SAM modèle permettent d'effectuer les opérations suivantes :

- `AutoPublishAlias`: En ajoutant cette propriété et en spécifiant un nom d'alias, AWS SAM :
  - Détecte le déploiement d'un nouveau code, en fonction des modifications apportées à l'URI Amazon S3 de la fonction Lambda.
  - Crée et publie une version mise à jour de cette fonction avec la dernière mise à jour du code.
  - Crée un alias avec un nom que vous spécifiez (sauf s'il existe déjà un alias) en pointant vers la version mise à jour de la fonction Lambda. Pour cela, les appels de fonction doivent utiliser le qualificateur de l'alias. Si vous n'êtes pas familier avec la gestion des versions et les alias des fonctions Lambda, consultez les versions et alias des fonctions AWS Lambda.
- `Deployment Preference Type` : dans l'exemple précédent, 10 % de votre trafic client est immédiatement déplacé vers la nouvelle version. Après 10 minutes, tout le trafic est déplacé vers la nouvelle version. Toutefois, si vos tests avant ou après le trafic échouent, ou si une CloudWatch alarme est déclenchée, annulez CodeDeploy votre déploiement. Vous pouvez spécifier la manière dont le trafic doit être déplacé d'une version à l'autre comme suit :
  - `Canary` : le trafic est déplacé en deux incréments. Vous pouvez choisir parmi les options Canary prédéfinies. Les options spécifient le pourcentage de trafic qui est déplacé vers la version mise à jour de votre fonction Lambda dans le premier incrément, et l'intervalle en minutes avant que le trafic restant soit déplacé dans le second incrément.
  - `Linear` : le trafic est déplacé en incréments égaux avec un nombre égal de minutes entre chaque incrément. Vous pouvez choisir parmi les options linéaires prédéfinies qui définissent le pourcentage de trafic déplacé pour chaque incrément et le nombre de minutes entre chaque incrément.
  - `AllAtOnce` : tout le trafic est déplacé en même temps de la fonction Lambda initiale vers sa version mise à jour.

  Le tableau ci-dessous décrit les autres options de déplacement du trafic disponibles, outre celle qui est utilisée dans l'exemple.

Type de préférence de déploiement
Canary10Percent30Minutes
Canary10Percent5Minutes
Canary10Percent10Minutes

Type de préférence de déploiement
Canary10Percent15Minutes
Linéaire 10 10 minutes PercentEvery
Linéaire 10 1 minute PercentEvery
Linéaire 10 2 minutes PercentEvery
Linéaire 10 3 minutes PercentEvery
AllAtOnce

- `Alarms`: il s'agit d' CloudWatch alarmes déclenchées par toute erreur générée par le déploiement. Elles annulent automatiquement votre déploiement. Par exemple, si le code mis à jour que vous déployez entraîne des erreurs dans l'application. Autre exemple : si l'une des métriques AWS Lambdaou CloudWatch des métriques personnalisées que vous avez spécifiées ont dépassé le seuil d'alarme.
- `Hooks` : fonctions de test pré- et post-trafic qui procèdent à des vérifications avant que ne commence le déplacement du trafic vers la nouvelle version et une fois ce déplacement terminé.
  - `PreTraffic`: Avant le début du transfert de trafic, CodeDeploy invoque la fonction Lambda du pré-trafic Hook. Cette fonction Lambda doit rappeler CodeDeploy et indiquer le succès ou l'échec. Si la fonction échoue, elle est abandonnée et signale un échec à CloudFormation. Si la fonction réussit, CodeDeploy passe au transfert de trafic.
  - `PostTraffic`: Une fois le transfert de trafic terminé, CodeDeploy invoque la fonction Lambda post-traffic hook. Ceci est similaire au hook pré-trafic, où la fonction doit rappeler CodeDeploy à pour signaler un succès ou un échec. Utilisez les crochets post-trafic pour exécuter des tests d'intégration ou d'autres opérations de validation.

  Pour en savoir plus, consultez SAM Reference to Safe Deployments (Référence SAM pour des déploiements en toute sécurité).

Déploiement progressif d'une fonction Lambda pour la première fois

Lors du déploiement progressif d'une fonction Lambda, une version de fonction préalablement déployée est CodeDeploy nécessaire pour transférer le trafic. Par conséquent, votre premier déploiement doit se faire en deux étapes :

- Étape 1 : déployer votre fonction Lambda et créer automatiquement des alias avec `AutoPublishAlias`.
- Étape 2 : réaliser votre déploiement progressif avec `DeploymentPreference`.

Si vous effectuez votre premier déploiement progressif en deux étapes, vous pouvez utiliser CodeDeploy une version précédente de la fonction Lambda à partir de laquelle transférer le trafic.

Étape 1 : déployer votre fonction Lambda

```
Resources:
MyLambdaFunction:
  Type: AWS::Serverless::Function
  Properties:
    Handler: index.handler
    Runtime: nodejs20.x
    CodeUri: s3://bucket/code.zip

    AutoPublishAlias: live
```

Étape 2 : réaliser votre déploiement progressif

```
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
```

En savoir plus

Pour un exemple pratique de configuration d'un déploiement progressif, consultez le Module 5 – déploiements Canary dans L'atelier AWS SAM complet.

Remarques de référence importantes pour AWS SAM

Cette section contient des notes et des annonces importantes pour AWS Serverless Application Model (AWS SAM).

Rubriques

- Notes importantes pour 2023

Notes importantes pour 2023

Octobre 2023

Fin de la prise en charge de Python 3.7 par l'CLIAWS SAM

Date de publication : 20/10/2023

Python 3.7a reçu le end-of-life statut en juin 2023. Le support AWS SAM CLI cessera d'être pris en charge Python 3.7 le 24 octobre 2023. Pour plus d'informations, consultez l'annonce publiée dans le aws-sam-cli GitHubréférentiel.

Cette modification concerne les utilisateurs suivants :

- Si vous utilisez Python 3.7 et installez le AWS SAM CLI through`pip`.
- Si vous utilisez `aws-sam-cli` en tant que bibliothèque et que vous créez votre application avec Python 3.7.

Si vous l'installez et le gérez AWS SAM CLI par une autre méthode, vous n'êtes pas concerné.

Pour les utilisateurs concernés, nous vous recommandons de mettre à niveau votre environnement de développement vers une version Python 3.8 plus récente.

Cette modification n'affecte pas la prise en charge de l'environnement Python 3.7 AWS Lambda d'exécution. Pour en savoir plus, consultez la rubrique politique d'obsolescence de l'exécution du Guide du développeur AWS Lambda .

Prise en charge de Terraform par la CLI AWS SAM

Cette section traite de l'utilisation de l'interface de ligne de AWS Serverless Application Model commande (AWS SAMCLI) avec vos Terraform projets et Terraform le cloud.

Pour soumettre des commentaires et des demandes de fonctionnalités, créez un Ticket GitHub.

Rubriques

- Démarrer avec la prise en charge de Terraform par la CLI AWS SAM
- Utilisation de la CLI AWS SAM avec Terraform pour le débogage et les tests locaux
- Utilisation de la CLI AWS SAM avec Serverless.tf pour le débogage et les tests locaux
- CLI AWS SAM avec référence Terraform
- Qu'est-ce que la prise en charge de Terraform par la CLI AWS SAM ?

Démarrer avec la prise en charge de Terraform par la CLI AWS SAM

Cette rubrique explique comment commencer à utiliser l'interface de ligne de AWS Serverless Application Model commande (AWS SAMCLI) avecTerraform.

Pour soumettre des commentaires et des demandes de fonctionnalités, créez un Ticket GitHub.

Rubriques

- AWS SAMCLITerraformprérequis
- Utilisation des commandes de la CLI AWS SAM avec Terraform
- Configuration pour les projets Terraform
- Configuration pour Terraform Cloud

AWS SAMCLITerraformprérequis

Remplissez toutes les conditions préalables pour commencer à utiliser la CLI AWS SAM avec vos projets Terraform.

1. Installer ou mettre à niveau la CLI AWS SAM

   Pour vérifier si la CLI AWS SAM est installée, procédez comme suit :

   ```
   $ sam --version
   ```

   Si la CLI AWS SAM est déjà installée, la version sera affichée. Pour procéder à une mise à niveau vers la version la plus récente, consultez Mise à niveau de la CLI AWS SAM en cours.

   Pour obtenir des instructions sur l'installation de la CLI AWS SAM ainsi que sur tous ses conditions préalables, consultez Installer la CLI AWS SAM.

2. Installer Terraform

   Pour vérifier si Terraform est installé, procédez comme suit :

   ```
   $ terraform -version
   ```

   Pour l'installer Terraform, veuillez consulter la section Installer Terraform dans le registre Terraform (langue française non garantie).

3. Installer Docker pour les tests locaux

   La CLI AWS SAM nécessite Docker pour les tests locaux. Pour installer Docker, consultez Installation de Docker pour une utilisation avec la CLI AWS SAM.

Utilisation des commandes de la CLI AWS SAM avec Terraform

Lorsque vous exécutez une commande CLI AWS SAM prise en charge, utilisez l'option `--hook-name` et indiquez la valeur `terraform`. Voici un exemple :

```
$ sam local invoke --hook-name terraform
```

Vous pouvez configurer cette option dans votre fichier de configuration de la CLI AWS SAM avec ce qui suit :

```
hook_name = "terraform"
```

Configuration pour les projets Terraform

Suivez les étapes décrites dans cette rubrique pour utiliser la CLI AWS SAM avec des projets Terraform.

Aucune configuration supplémentaire n'est requise si vous créez vos AWS Lambda artefacts en dehors de votre Terraform projet. Utilisation de la CLI AWS SAM avec Terraform pour le débogage et les tests locauxReportez-vous à la section pour commencer à utiliser le AWS SAMCLI.

Si vous créez vos artefacts Lambda dans le cadre de vos projets Terraform, vous devez faire ce qui suit :

1. Installation de la version Python 3.8 ou ultérieure
2. Installez l'outil Make.
3. Définissez la logique de création de vos artefacts Lambda dans votre projet Terraform.
4. Définissez une ressource `sam metadata` pour informer la CLI AWS SAM de votre logique de création.
5. Utilisez la AWS SAMCLI `sam build` commande pour créer vos artefacts Lambda.

Installation de la version Python 3.8 ou ultérieure

PythonLa version 3.8 ou une version plus récente est requise pour être utilisée avec le AWS SAMCLI. Lorsque vous exécutez `sam build`, la CLI AWS SAM génère des fichiers `makefiles` qui contiennent des commandes Python pour créer vos artefacts Lambda.

Pour obtenir des instructions d'installation, veuillez consulter la rubrique Téléchargement de Python (langue française non garantie) dans le Guide du débutant de Python.

Vérifiez que Python 3.8 ou version ultérieure est ajouté au chemin de votre machine en exécutant :

```
$ python --version
```

La sortie doit afficher une version de Python 3.8 ou plus récente.

Installer l'outil Make

GNU Make est un outil qui contrôle la génération d'exécutables et d'autres fichiers non sources pour votre projet. La CLI AWS SAM crée des fichiers `makefiles` qui dépendent de cet outil pour créer vos artefacts Lambda.

Si la Make n'est pas installée sur votre ordinateur local, installez-la avant de continuer.

Pour Windows, vous pouvez l'installer à l'aide de Chocolatey. PPour les instructions, consultez la rubrique Utilisation de Chocolatey dans le guide Comment installer et utiliser « Make » sous Windows

Définir la logique de création des artefacts Lambda

Utilisez le type de ressource Terraform `null_resource` pour définir votre logique de création Lambda. Voici un exemple qui utilise un script de création personnalisé pour créer une fonction Lambda.

```
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"
    }
}
```

Définir une ressource sam metadata

La ressource `sam metadata` est un type de ressource Terraform `null_resource` qui fournit à la CLI AWS SAM les informations dont elle a besoin pour localiser vos artefacts Lambda. Chaque fonction ou couche Lambda de votre projet nécessite une ressource `sam metadata` unique. Pour en savoir plus sur ce type de ressource, veuillez consulter la section null_resource dans le registre Terraform (langue française non garantie).

Pour définir une ressource sam metadata

1. Donnez à votre ressource un nom commençant par `sam_metadata_` pour l'identifier comme étant une ressource sam metadata.
2. Définissez les propriétés de votre artefact Lambda dans le bloc `triggers` de votre ressource.
3. Spécifiez votre `null_resource` qui contient votre logique de création Lambda avec l'argument `depends_on`.

Voici un exemple de modèle :

```
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
  ]
}
```

Voici un exemple de ressource `sam metadata` :

```
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
    ]
}
```

Le contenu de votre ressource `sam metadata` variera en fonction du type de ressource Lambda (fonction ou couche) et du type d'empaquetage (ZIP ou image). Pour plus d'informations et d'exemples, consultez ressource de métadonnées sam.

Lorsque vous configurez une ressource `sam metadata` et utilisez une commande CLI AWS SAM prise en charge, la CLI AWS SAM génère le fichier de métadonnées avant d'exécuter la commande CLI AWS SAM. Une fois que vous avez généré ce fichier, vous pouvez utiliser l'option `--skip-prepare-infra` associée aux futures commandes CLI AWS SAM pour ignorer le processus de génération de métadonnées et gagner du temps. Cette option ne doit être utilisée que si vous n'avez apporté aucune modification à l'infrastructure, telle que la création de nouvelles fonctions Lambda ou de nouveaux points de terminaison d'API.

Utiliser la CLI AWS SAM pour créer vos artefacts Lambda

Utilisez la AWS SAMCLI `sam build` commande pour créer vos artefacts Lambda. Lorsque vous exécutez `sam build`, la CLI AWS SAM effectue les opérations suivantes :

1. Recherche des ressources `sam metadata` dans votre projet Terraform pour connaître et localiser vos ressources Lambda.
2. Lance votre logique de création Lambda pour créer vos artefacts Lambda.
3. Crée un `.aws-sam` répertoire qui organise votre Terraform projet à utiliser avec les AWS SAMCLI `sam local` commandes.

Pour créer à l'aide de la fonction de création SAM

1. À partir du répertoire contenant votre module racine Terraform, procédez comme suit :

   ```
   $ sam build --hook-name terraform
   ```

2. Pour créer une fonction ou une couche Lambda spécifique, exécutez les actions suivantes

   ```
   $ sam build --hook-name terraform lambda-resource-id
   ```

   L'identifiant de la ressource Lambda peut être le nom de la fonction Lambda ou l'adresse complète de la ressource Terraform, par exemple `aws_lambda_function.list_books` ou `module.list_book_function.aws_lambda_function.this[0]`.

Si le code source de votre fonction ou d'autres fichiers de configuration Terraform se trouvent en dehors du répertoire contenant votre module racine Terraform, vous devez spécifier l'emplacement. Utilisez l'option `--terraform-project-root-path` pour spécifier le chemin absolu ou relatif vers le répertoire de premier niveau contenant ces fichiers. Voici un exemple :

```
$ sam build --hook-name terraform --terraform-project-root-path ~/projects/terraform/
demo
```

Créer à l'aide d'un conteneur

Lorsque vous exécutez la AWS SAMCLI `sam build` commande, vous pouvez configurer le AWS SAMCLI pour créer votre application à l'aide d'un Docker conteneur local.

> **Note**
>
> Vous devez avoir installé et configuré Docker. Pour obtenir des instructions, veuillez consulter Installation de Docker pour une utilisation avec la CLI AWS SAM.

Pour créer à l'aide d'un conteneur

1. Créez un `Dockerfile` contenant les outils Terraform, Python et Make. Vous devez également inclure le moteur d'exécution de votre fonction Lambda.

   Voici un exemple de `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. Utilisez docker build pour créer votre image Docker.

   Voici un exemple :

```
$ docker build --tag terraform-build:v1 <path-to-directory-containing-Dockerfile>
```

3. Exécutez la AWS SAMCLI `sam build` commande avec les `--build-image` options `--use-container` et.

   Voici un exemple :

```
$ sam build --use-container --build-image terraform-build:v1
```

Étapes suivantes

Pour commencer à utiliser la CLI AWS SAM avec vos projets Terraform, consultez Utilisation de la CLI AWS SAM avec Terraform pour le débogage et les tests locaux.

Configuration pour Terraform Cloud

Nous vous recommandons d'utiliser une Terraform v1.6.0 version plus récente. Si vous utilisez une ancienne version, vous devez générer un fichier de Terraform plan localement. Le fichier de plan local fournit les AWS SAM CLI informations dont il a besoin pour effectuer des tests et un débogage locaux.

Pour générer un fichier de plan local

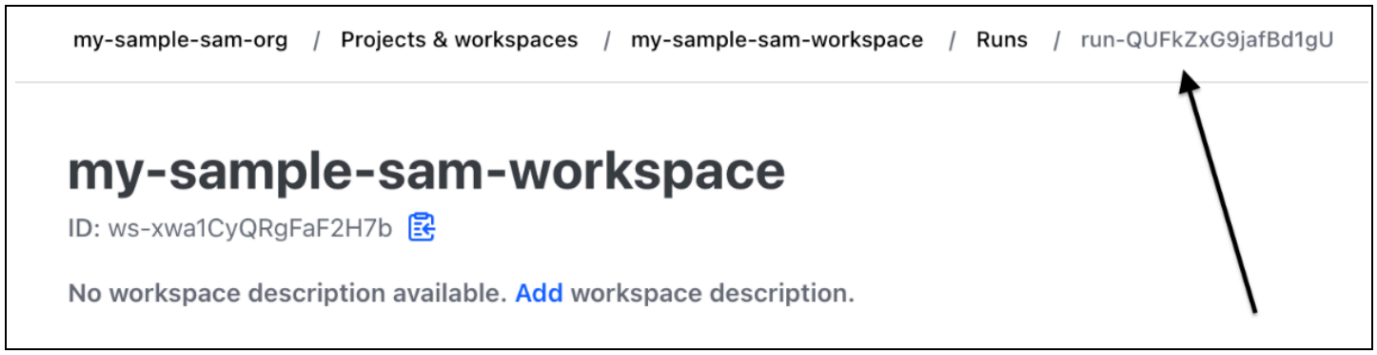
> **Note**
>
> Ces étapes ne sont pas obligatoires pour les versions ultérieures Terraform v1.6.0 ou ultérieures. Pour commencer à utiliser le AWS SAM CLI withTerraform Cloud, voirUtilisation de la CLI AWS SAM avec Terraform.

1. Configurer un jeton d'API : le type de jeton dépend de votre niveau d'accès. Pour en savoir plus, consultez la rubrique Jetons d'API dans la documentation Terraform Cloud.
2. Définisser la variable d'environnement de votre jeton d'API : voici un exemple tiré de la ligne de commande :

```
$ export TOKEN="<api-token-value>"
```

3. Obtenez votre identifiant d'exécution : depuis la Terraform Cloud console, recherchez l'identifiant d'Terraformexécution de l'exécution que vous souhaitez utiliser avec le AWS SAMCLI.

   L'identifiant d'exécution se trouve dans la piste de navigation de votre exécution.



4. Récupérer le fichier de plan : à l'aide de votre jeton d'API, obtenez votre fichier de plan local. Voici un exemple tiré de la ligne de commande :

```
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
```

Vous êtes maintenant prêt à utiliser la CLI AWS SAM avec Terraform Cloud. Lorsque vous utilisez une commande CLI AWS SAM prise en charge, utilisez l'option `--terraform-plan-file` pour spécifier le nom et le chemin de votre fichier de plan local. Voici un exemple :

```
$ sam local invoke --hook-name terraform --terraform-plan-file custom-plan.json
```

Voici un exemple utilisant la commande `sam local start-api` :

```
$ sam local start-api --hook-name terraform --terraform-plan-file custom-plan.json
```

Pour un exemple d'application que vous pouvez utiliser avec ces exemples, consultez [api_gateway_v2_tf_cloud] dans le référentiel GitHub aws-samples.

Étapes suivantes

Pour commencer à utiliser la CLI AWS SAM avec Terraform Cloud, consultez Utilisation de la CLI AWS SAM avec Terraform pour le débogage et les tests locaux.

Utilisation de la CLI AWS SAM avec Terraform pour le débogage et les tests locaux

Cette rubrique explique comment utiliser les commandes d'interface de ligne de AWS Serverless Application Model commande (AWS SAMCLI) prises en charge avec vos Terraform projets etTerraform Cloud.

Pour soumettre des commentaires et des demandes de fonctionnalités, créez un Ticket GitHub.

Rubriques

- Tests locaux avec sam local invoke
- Tests locaux avec sam local start-api
- Tests locaux avec sam local start-lambda
- Restrictions liées à Terraform

Tests locaux avec sam local invoke

> **Note**
>
> Pour utiliser le AWS SAMCLI pour tester localement, Docker doit être installé et configuré. Pour obtenir des instructions, veuillez consulter Installation de Docker pour une utilisation avec la CLI AWS SAM.

Voici un exemple de test local de votre fonction Lambda en transférant un événement :

```
$ sam local invoke --hook-name terraform hello_world_function -e events/event.json -
```

Pour en savoir plus sur l'utilisation de cette commande, veuillez consulter la section Présentation des tests avec sam local invoke.

Tests locaux avec sam local start-api

Pour utiliser `sam local start-api` avec Terraform, procédez comme suit :

```
$ sam local start-api --hook-name terraform
```

Voici un exemple :

```
$ 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)
```

Pour en savoir plus sur la commande, consultez Présentation des tests avec sam local start-api.

Fonctions Lambda qui utilisent des mécanismes d'autorisation Lambda

Pour les fonctions Lambda configurées pour utiliser des mécanismes d'autorisation Lambda, la CLI AWS SAM invoquera automatiquement votre mécanisme d'autorisation Lambda avant d'invoquer le point de terminaison de votre fonction Lambda.

- Pour en savoir plus sur cette fonctionnalité dans le AWS SAMCLI, voirFonctions Lambda qui utilisent des mécanismes d'autorisation Lambda.
- Pour plus d'informations sur l'utilisation des mécanismes d'autorisation Lambda dans Terraform, veuillez consulter la section Resource: aws_api_gateway_authorizer dans le registre Terraform (langue française non garantie).

Tests locaux avec sam local start-lambda

Voici un exemple de test local de votre fonction Lambda avec le AWS Command Line Interface ()AWS CLI :

1. Utilisez la CLI AWS SAM pour créer un environnement de test local :

   ```
   $ sam local start-lambda --hook-name terraform hello_world_function
   ```

2. Utilisez le AWS CLI pour appeler votre fonction localement :

```
$ 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
```

Pour en savoir plus sur la commande, consultez Présentation des tests avec sam local start-lambda.

Restrictions liées à Terraform

Les limites suivantes s'appliquent à l'utilisation de la CLI AWS SAM avec Terraform :

- Fonctions Lambda liées à plusieurs couches.
- Variables locales Terraform définissant les liens entre les ressources.
- Référence à une fonction Lambda qui n'a pas encore été créée. Cela inclut les fonctions définies dans l'attribut body de la ressource de l'API REST.

Pour éviter ces limitations, vous pouvez exécuter `terraform apply` lorsqu'une nouvelle ressource est ajoutée.

Utilisation de la CLI AWS SAM avec Serverless.tf pour le débogage et les tests locaux

L'interface de ligne de AWS Serverless Application Model commande (AWS SAMCLI) peut être utilisée avec les modules Serverless.TF pour le débogage local et le test de vos AWS Lambda fonctions et couches. Les commandes CLI AWS SAM suivantes sont prises en charge :

- `sam build`
- `sam local invoke`
- `sam local start-api`
- `sam local start-lambda`

> **Note**
>
> Les versions 4.6.0 et ultérieures de Serverless.tf prennent en charge l'intégration de la CLI AWS SAM.

Pour commencer à utiliser le AWS SAMCLI avec vos modules ServerLess.TF, passez à la dernière version de ServerLess.TF et du. AWS SAMCLI

À partir de la version 6.0.0 de serverless.tf, vous devez définir le paramètre `create_sam_metadata` sur `true`. Cela génère les ressources de métadonnées requises par la AWS SAMCLI `sam build` commande.

Pour en savoir plus sur Serverless.tf, consultez terraform-aws-lambda-module.

CLI AWS SAM avec référence Terraform

Cette section est la référence pour l'utilisation de l'interface de ligne de AWS Serverless Application Model commande (AWS SAMCLI) Terraform pour le débogage et les tests locaux.

Pour soumettre des commentaires et des demandes de fonctionnalités, créez un Ticket GitHub.

AWS SAM référence des fonctionnalités prises en charge

La documentation de référence sur les fonctionnalités de la CLI AWS SAM prises en charge pour une utilisation avec Terraform est disponible ici :

- sam build
- sam local invoke
- sam local start-api
- sam local start-lambda

Référence spécifique à Terraform

La documentation de référence spécifique à l'utilisation de la CLI AWS SAM avec Terraform est disponible ici :

- ressource de métadonnées sam

ressource de métadonnées sam

Cette page contient des informations de référence sur le type de ressource sam metadata resource utilisé avec les projets Terraform.

- Pour une introduction à l'utilisation de l'interface de ligne de AWS Serverless Application Model commande (AWS SAMCLI) avecTerraform, voirQu'est-ce que la prise en charge de Terraform par la CLI AWS SAM ?.
- Pour utiliser la CLI AWS SAM avec Terraform, consultez Utilisation de la CLI AWS SAM avec Terraform pour le débogage et les tests locaux.

Rubriques

- Arguments
- Exemples

Arguments

Argument	Description
`built_output_path`	Le chemin d'accès aux artefacts créés par votre AWS Lambda fonction.
`docker_build_args`	Chaîne décodée de l'objet JSON des arguments de création Docker. Cet argument est facultatif.
`docker_context`	Chemin d'accès au répertoire contenant le contexte de création de l'image Docker.
`docker_file`	Chemin d'accès au fichier Docker. Ce chemin d'accès est relatif au chemin `docker_context` . Cet argument est facultatif. La valeur par défaut est `Dockerfile` .
`docker_tag`	La valeur de la balise d'image Docker créée. Cette valeur est facultati ve.
`depends_on`	Le chemin d'accès à la ressource de création pour votre fonction ou couche Lambda. Pour en savoir plus, veuillez consulter la section L'argument depends_on dans le registre Terraform (langue française non garantie).

Argument	Description
`original_source_code`	Le chemin d'accès vers l'emplacement de définition de votre fonction Lambda. Cette valeur peut être une chaîne, un tableau de chaînes ou un objet JSON décodé sous forme de chaîne. • Pour les tableaux de chaînes, seule la première valeur est utilisée, car les chemins de code multiples ne sont pas pris en charge. • Pour les objets JSON, la propriété `source_code_property` doit également être définie.
`resource_name`	Nom de la fonction Lambda.
`resource_type`	Format du type de package de votre fonction Lambda. Les valeurs acceptées sont : • `IMAGE_LAMBDA_FUNCTION` • `LAMBDA_LAYER` • `ZIP_LAMBDA_FUNCTION`
`source_code_property`	Le chemin d'accès au code de ressource Lambda dans l'objet JSON. Définissez cette propriété lorsque `original_source_code` est un objet JSON.

Exemples

ressource de métadonnées sam faisant référence à une fonction Lambda utilisant le type de package ZIP

```
# 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
  ]
}
```

ressource de métadonnées sam faisant référence à une fonction Lambda utilisant le type de package image

```
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"
  }
}
```

ressource de métadonnées sam faisant référence à une couche Lambda

```
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]
}
```

Qu'est-ce que la prise en charge de Terraform par la CLI AWS SAM ?

Utilisez l'interface de ligne de AWS Serverless Application Model commande (AWS SAMCLI) avec vos Terraform projets ou Terraform Cloud pour effectuer le débogage et le test locaux de :

- AWS Lambda fonctions et couches.
- Amazon API Gateway APIs.

Pour une introduction à Terraform, consultez Qu'est-ce que c'est Terraform ? sur le HashiCorpsite Web Terraform .

Pour soumettre des commentaires et des demandes de fonctionnalités, créez un Ticket GitHub.

> **Note**
>
> Dans le cadre de l'étape d'analyse AWS SAMCLI de l'intégration, AWS SAMCLI les commandes utilisateur des processus génèrent des fichiers et des données de projet. La sortie de la commande doit rester inchangée, mais dans certains environnements, l'environnement ou le lanceur peuvent injecter des journaux ou des informations supplémentaires dans la sortie.

Rubriques

- Qu'est-ce que c'est AWS SAMCLI ?
- Comment utiliser la CLI AWS SAM avec Terraform ?
- Étapes suivantes

Qu'est-ce que c'est AWS SAMCLI ?

AWS SAMCLIIl s'agit d'un outil de ligne de commande que vous pouvez utiliser avec des AWS SAM modèles et des intégrations tierces prises en chargeTerraform, par exemple pour créer et exécuter vos applications sans serveur. Pour une introduction à la AWS SAMCLI, voirQu'est-ce que c'est AWS SAMCLI ?.

prend AWS SAMCLI en charge les commandes suivantes pour Terraform :

- `sam local invoke`— Lance un appel ponctuel d'une ressource AWS Lambda fonctionnelle localement. Pour en savoir plus sur la commande, consultez Présentation des tests avec sam local invoke.
- `sam local start-api` : exécutez vos ressources Lambda localement et testez-les via un hôte de serveur HTTP local. Ce type de test est utile pour les fonctions Lambda qui sont invoquées par un point de terminaison de passerelle API. Pour en savoir plus sur la commande, consultez Présentation des tests avec sam local start-api.
- `sam local start-lambda`— Démarrez un point de terminaison local pour votre fonction Lambda afin d'invoquer votre fonction localement en utilisant AWS Command Line Interface (AWS CLI) ou. SDKs Pour en savoir plus sur la commande, consultez Présentation des tests avec sam local start-lambda.

Comment utiliser la CLI AWS SAM avec Terraform ?

Le flux de travail Terraform de base comprend trois étapes : écrire, planifier et appliquer. Grâce à la prise en AWS SAMCLI charge deTerraform, vous pouvez tirer parti de l' AWS SAMCLI`sam local`ensemble de commandes tout en continuant à utiliser vos Terraform flux de travail pour gérer vos applications sur AWS. En règle générale, cela signifie ce qui suit :

- Écrire : créez votre infrastructure sous forme de code à l'aide de Terraform.
- Test et débogage : utilisez la CLI AWS SAM pour tester et déboguer localement vos applications.
- Planifier : prévisualisez les modifications avant de les appliquer.
- Appliquer : provisionnez votre infrastructure.

Pour un exemple d'utilisation du AWS SAMCLI withTerraform, voir Better together : AWS SAMCLI and HashiCorp Terraform at the AWS Compute Blog.

Étapes suivantes

Pour remplir toutes les conditions préalables et configurer Terraform, consultez Démarrer avec la prise en charge de Terraform par la CLI AWS SAM.

Publier votre candidature à l'aide du AWS SAMCLI

Pour mettre votre AWS SAM application à la disposition d'autres utilisateurs qui pourront la trouver et la déployer, vous pouvez utiliser le AWS SAMCLI pour la publier sur le AWS Serverless Application Repository. Pour publier votre application à l'aide du AWS SAMCLI, vous devez la définir à l'aide d'un AWS SAM modèle. Vous devez également l'avoir testée localement ou dans le Cloud AWS .

Suivez les instructions de cette rubrique pour créer une nouvelle application, créer une nouvelle version d'une application existante ou mettre à jour les métadonnées d'une application existante. (Ce que vous faites dépend du fait que l'application existe déjà dans le AWS Serverless Application Repository et que les métadonnées de l'application sont en train de changer.) Pour plus d'informations sur les métadonnées des applications, veuillez consulter AWS SAM propriétés de la section de métadonnées du modèle.

Conditions préalables

Avant de publier une application à l' AWS Serverless Application Repository aide du AWS SAMCLI, vous devez disposer des éléments suivants :

- La CLI AWS SAM installée. Pour de plus amples informations, veuillez consulter Installer la CLI AWS SAM. Pour déterminer si la CLI AWS SAM est installée, exécutez la commande suivante :

  ```
  sam --version
  ```

- Un AWS SAM modèle valide.
- Le code de votre application et les dépendances auxquels le AWS SAM modèle fait référence.
- Une version sémantique, uniquement nécessaire pour partager publiquement votre application. Cette valeur peut être aussi simple que 1.0.
- Une URL qui pointe vers le code source de votre application.
- Un fichier `README.md`. Ce fichier doit décrire comment les clients peuvent utiliser votre application et comment la configurer avant de la déployer sur leurs propres comptes AWS .
- Un fichier `LICENSE.txt`, uniquement nécessaire pour partager publiquement votre application.
- Si votre application contient des applications imbriquées, vous devez les avoir déjà publiées dans le AWS Serverless Application Repository.

- Une stratégie de compartiment Amazon Simple Storage Service (Amazon S3) valide qui accorde les autorisations de lecture du service pour les artefacts que vous téléchargez sur Amazon S3 quand vous empaquetez votre application. Pour configurer cette règle, procédez comme suit :

  1. Ouvrez la console Amazon S3 à l'adresse https://console.aws.amazon.com/s3/.
  2. Choisissez le nom du compartiment Amazon S3 que vous avez utilisé pour empaqueter votre application.
  3. Choisissez Permissions (Autorisations).
  4. Dans l'onglet Permissions (Autorisations), sous Bucket Policy (Stratégie de compartiment), choisissez Edit (Modifier).
  5. Dans la page Edit bucket policy (Modifier la stratégie de compartiment), collez l'instruction de stratégie suivante dans l'éditeur de Policy (Stratégie). Dans la déclaration de stratégie, veillez à utiliser le nom de votre compartiment dans l'élément `Resource` et votre ID de compte AWS dans l'élément `Condition`. L'expression contenue dans l'`Condition`élément garantit qu'il AWS Serverless Application Repository est autorisé à accéder uniquement aux applications du AWS compte spécifié. Pour plus d'informations sur les déclarations de stratégie, consultez Référence des éléments de stratégie IAM JSON dans le Guide de l'utilisateur IAM.

     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. Sélectionnez Save Changes (Enregistrer les modifications).

Publication d'une nouvelle application

Étape 1 : ajouter une **`Metadata`** section au AWS SAM modèle

Ajoutez d'abord une `Metadata` section à votre AWS SAM modèle. Fournissez les informations relatives à l'application à publier dans le AWS Serverless Application Repository.

Voici un exemple de section `Metadata` :

```
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
        ...
```

Pour plus d'informations sur la `Metadata` section du AWS SAM modèle, consultezAWS SAM propriétés de la section de métadonnées du modèle.

Étape 2 : empaqueter l'application

Exécutez la commande CLI AWS SAM suivante, qui télécharge les artefacts de l'application sur Amazon S3 et génère un nouveau fichier modèle appelé `packaged.yaml` :

```
sam package --output-template-file packaged.yaml --s3-bucket <your-bucket-name>
```

Vous utilisez le modèle `packaged.yaml` dans l'étape suivante pour publier l'application sur le AWS Serverless Application Repository. Ce fichier est similaire au fichier de modèle original (`template.yaml`) mais présente une différence significative : le `CodeUri`, la `LicenseUrl` et les propriétés `ReadmeUrl` pointent vers le compartiment Amazon S3 et les objets qui contiennent les artefacts respectifs.

L'extrait suivant d'un exemple de fichier de modèle `packaged.yaml` montre la propriété `CodeUri` :

```
MySampleFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: s3://bucketname/fbd77a3647a4f47a352fc0ObjectGUID

...
```

Étape 3 : publier l'application

Pour publier une version privée de votre AWS SAM application sur le AWS Serverless Application Repository, exécutez la AWS SAMCLI commande suivante :

```
sam publish --template packaged.yaml --region us-east-1
```

La sortie de la commande `sam publish` inclut un lien vers votre application sur le AWS Serverless Application Repository. Vous pouvez également accéder directement à la page d'accueil AWS Serverless Application Repository et rechercher votre application.

Étape 4 : partager l'application (facultatif)

Par défaut, votre application est définie sur privée, de sorte qu'elle n'est pas visible par les autres comptes AWS . Pour partager votre application avec d'autres personnes, vous devez soit la rendre publique, soit autoriser une liste spécifique de AWS comptes.

Pour plus d'informations sur le partage de votre application à l'aide de AWS CLI, consultez les exemples de politiques AWS Serverless Application Repository basées sur les ressources dans le guide du AWS Serverless Application Repository développeur. Pour plus d'informations sur le partage de votre application à l'aide du AWS Management Console, consultez Partage d'une application dans le Guide du développeur AWS Serverless Application Repository .

Publication d'une nouvelle version d'une application existante

Après avoir publié une application sur le AWS Serverless Application Repository, vous souhaiterez peut-être en publier une nouvelle version. Par exemple, vous avez peut-être modifié votre code de fonction Lambda ou ajouté un nouveau composant à votre architecture d'application.

Pour mettre à jour une application que vous avez déjà publiée, publiez-la à nouveau en utilisant le même processus que celui décrit précédemment. Dans la section `Metadata` du fichier de modèle AWS SAM , indiquez le même nom d'application que vous avez publié à l'origine, mais incluez une nouvelle valeur `SemanticVersion`.

Par exemple, considérons une application publiée avec le nom `SampleApp` et un `SemanticVersion` de `1.0.0`. Pour mettre à jour cette application , AWS SAM doit avoir le nom de l'application `SampleApp` et un `SemanticVersion` de `1.0.1` (ou autre que `1.0.0`).

Rubriques supplémentaires

- AWS SAM propriétés de la section de métadonnées du modèle

AWS SAM propriétés de la section de métadonnées du modèle

`AWS::ServerlessRepo::Application` est une clé de métadonnées que vous pouvez utiliser pour spécifier les informations d'application que vous souhaitez publier dans le AWS Serverless Application Repository.

> **Note**
>
> CloudFormation les fonctions intrinsèques ne sont pas prises en charge par la clé de `AWS::ServerlessRepo::Application` métadonnées.

Propriétés

Ce tableau fournit des informations sur les propriétés de la `Metadata` section du AWS SAM modèle. Cette section est requise pour publier des applications à l' AWS Serverless Application Repository aide du AWS SAMCLI.

Propriété	Type	Obligatoire	Description
`Name`	String	TRUE	Nom de l'application. Longueur minimale = 1. Longueur maximale = 140. Modèle : `"[a-zA-Z0-9\\-]+";`
`Description`	String	TRUE	Description de l'application. Longueur minimale = 1. Longueur maximale = 256.
`Author`	String	TRUE	Nom de l'auteur qui publie l'application. Longueur minimale = 1. Longueur maximale = 127. Modèle : `"^[a-z0-9](([a-z0-9]|-(?!-))*[a-z0-9])?$";`
`SpdxLicenseId`	String	FAUX	Un identifiant de licence valide. Pour afficher la liste des identifiants de licence valides, veuillez consulter Liste des licences SPDX sur le site web de Data Exchange de packages de progiciels (SPDX).
`LicenseUrl`	Chaîne	FAUX	La référence à un fichier de licence local, ou à un lien Amazon S3 vers un fichier de licence, qui correspond à la valeur spdxLicenseID de votre application. Un fichier AWS SAM modèle qui n'a pas été empaqueté à l'aide de la `sam package` commande peut contenir une référence à un fichier local pour cette propriété. Toutefois, pour qu'une applicati on soit publiée à l'aide de la commande `sam publish`, cette propriété doit être une référence à un compartiment Amazon S3. Taille maximale : 5 Go

Propriété	Type	Obligatoire	Description
			Vous devez fournir une valeur pour cette propriété afin de rendre votre application publique. Notez que vous ne pouvez pas mettre à jour cette propriété après la publication de votre application. Ainsi, pour ajouter une licence à une application, vous devez soit la supprimer d'abord, soit publier une nouvelle application avec un nom différent.
`ReadmeUrl`	String	FAUX	La référence à un fichier readme local ou à un lien Amazon S3 vers le fichier readme qui contient une description plus détaillée de l'application et de son fonctionnement. Un fichier AWS SAM modèle qui n'a pas été empaqueté à l'aide de la `sam package` commande peut contenir une référence à un fichier local pour cette propriété. Toutefois, pour être publié à l'aide de la commande `sam publish`, cette propriété doit être une référence à un compartiment Amazon S3. Taille maximale : 5 Go
`Labels`	String	FAUX	Les étiquettes qui améliorent la découverte d'applications dans les résultats de recherche. Longueur minimale = 1. Longueur maximale = 127. Nombre maximal d'étiquettes : 10. Modèle : `"^[a-zA-Z0-9+\\-_:\\/@]+$";`
`HomePageUrl`	String	FAUX	URL contenant des informations supplémentaires sur l'application, par exemple l'emplacement de votre GitHub référentiel pour l'application.

Propriété	Type	Obligatoire	Description
`SemanticVersion`	String	FAUX	Version sémantique de l'application. Pour obtenir la spécification de gestion sémantique des versions, consultez le site web de Gestion sémantique des versions. Vous devez fournir une valeur pour cette propriété afin de rendre votre application publique.
`SourceCodeUrl`	String	FAUX	Lien vers un référentiel public pour le code source de votre application.

Cas d'utilisation

Cette section répertorie les cas d'utilisation pour la publication d'applications, ainsi que les propriétés de `Metadata` qui sont traitées pour ce cas d'utilisation. Les propriétés qui sont pas répertoriés pour un cas d'utilisation donné sont ignorées.

- Création d'une nouvelle application — Une nouvelle application est créée si aucune application ne porte le AWS Serverless Application Repository même nom pour un compte.
  - `Name`
  - `SpdxLicenseId`
  - `LicenseUrl`
  - `Description`
  - `Author`
  - `ReadmeUrl`
  - `Labels`
  - `HomePageUrl`
  - `SourceCodeUrl`
  - `SemanticVersion`
  - Le contenu du AWS SAM modèle (par exemple, toutes les sources d'événements, les ressources et le code de fonction Lambda)

- Création d'une version d'application — Une version d'application est créée s'il existe déjà une application dans le nom correspondant AWS Serverless Application Repository à un compte et qu'elle SemanticVersion est en train de changer.
  - `Description`
  - `Author`
  - `ReadmeUrl`
  - `Labels`
  - `HomePageUrl`
  - `SourceCodeUrl`
  - `SemanticVersion`
  - Le contenu du AWS SAM modèle (par exemple, toutes les sources d'événements, les ressources et le code de fonction Lambda)

- Mise à jour d'une application — Une application est mise à jour s'il existe déjà une application AWS Serverless Application Repository dont le nom correspond à celui d'un compte et SemanticVersion qu'elle ne change pas.
  - `Description`
  - `Author`
  - `ReadmeUrl`
  - `Labels`
  - `HomePageUrl`

Exemple

Voici un exemple de section `Metadata` :

```
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
```

Historique du document pour AWS SAM

Le tableau suivant décrit les modifications importantes dans chaque édition du Guide du développeur AWS Serverless Application Model . Pour recevoir les notifications des mises à jour de cette documentation, abonnez-vous à un flux RSS.

- Dernière mise à jour de la documentation : 20 juin 2024

Modification	Description	Date
Contenu restructuré et mis à jour dans l'ensemble du guide du développeur	Réorganisation et restructuration du guide pour améliorer la découvrabilité et la convivialité. Titres mis à jour et améliorés. A fourni des détails supplémentaires lors de l'introduction des sujets et des concepts.	20 juin 2024
Ajout AWS SAMCLI du support pour Ruby 3.3	Ruby 3.3 est désormais disponible en tant que référentiel d'exécution et d'images. Voir Référentiels d'images et sam init pour plus de détails.	4 avril 2024
Options de AWS SAMCLI commande ajoutées	De nouvelles options sont disponibles pour la commande sam local start-api :`--ssl-cert-file PATH` ,`--ssl-key-file PATH` . De plus, la nouvelle option de ligne de commande `--add-host LIST` est disponible pour sam local invoke, sam local start-api et sam local start-lambda	20 mars 2024

Ajout AWS SAMCLI du support pour .NET 8	.NET 8 est désormais disponible en tant que référenti el d'exécution et d'images. Les temps d'exécution et les référentiels d'images pour .NET Core 3.1, Node.js 14, Node.js 12, Python 3.7, Ruby 2.7 ne sont plus pris en charge. Voir Référentiels d'images et sam init.	22 février 2024
Ajout du programme d'install ation du package AWS SAMCLI arm64 pour Linux	Pour obtenir des instructions, reportez-vous à la section Installation du AWS SAMCLI.	6 décembre 2023
Ajout de l'option --watch-e xclude pour la commande sam sync AWS SAMCLI	Empêchez les fichiers et les dossiers de lancer une synchronisation. Pour en savoir plus, consultez Spécifier les fichiers et les dossiers qui ne lanceront pas de synchroni sation.	6 décembre 2023
Ajout d'une build-in-source option pour la commande AWS SAMCLI sam sync	Créez votre projet dans votre dossier source pour accélérer le processus de création. Pour en savoir plus, consultez Accélérer les temps de création en créant votre projet dans le dossier source.	6 décembre 2023

Ajout d'une build-in-source option pour la commande AWS SAMCLI sam build	Créez votre projet dans votre dossier source pour accélérer le processus de création. Pour en savoir plus, consultez Accélérer les temps de création en créant votre projet dans le dossier source.	6 décembre 2023
Ajout d'un nouveau support de ressources pour la commande AWS SAMCLI Remote Invoke	Utilisez `sam remote invoke` avec les applications Kinesis Data Streams, les files d'attente Amazon SQS et les machines d'état Step Functions. Pour en savoir plus, consultez Using sam remote test-event.	15 novembre 2023
Ajout d'une nouvelle commande d'événement de test AWS SAMCLI à distance pour les événements de test partageables	Utilisez le AWS SAM CLI pour accéder aux événement s de test partageables et les gérer à partir du registre de EventBridge schémas afin de tester vos fonctions Lambda dans le. AWS Cloud Pour en savoir plus, consultez la rubrique Using sam remote test-event.	3 octobre 2023
La prise en charge de Terraform par la CLI AWS SAM est désormais généralem ent disponible	Pour en savoir plus sur la prise en charge de Terraform par la CLI AWS SAM, consultez Prise en charge de Terraform par la CLI AWS SAM.	5 septembre 2023

AWS SAMCLISupport supplémentaire pour Terraform Cloud	La CLI AWS SAM prend désormais en charge les tests locaux pour Terraform Cloud. Pour en savoir plus, consultez Configuration pour Terraform Cloud.	5 septembre 2023
Ajout YAML du support de format de fichier pour le fichier AWS SAMCLI de configuration	Le format de fichier `[.yaml|.yml]` est AWS SAMCLI désormais pris en charge. Les pages de configuration de la CLI AWS SAM et du fichier de configuration de la CLI AWS SAM ont été mises à jour.	18 juillet 2023
Ajout de AWS SAMCLIsam local start-api la prise en charge des commandes pour Terraform	À quoi sert AWS SAMCLI le support Terraform ? la section a été mise à jour pour inclure AWS SAMCLI `sam local start-api` le support des commandes pourTerraform.	6 juillet 2023
Ajout d'une nouvelle commande d'appel AWS SAMCLI à distance	Pour commencer à utiliser `sam remote invoke`, consultez la section Utilisation de sam remote invoke.	22 juin 2023
Type de ressource AWS AppSyncGraphQL API sans serveur ajouté	Créez une nouvelle `AWS::Serverless::GraphQLApi` section qui décrit comment définir une GraphQL API ressource avec AWS SAM.	22 juin 2023

Ajout AWS SAMCLI du support pour Ruby 3.2	Mettez à jour la page d'initialisation du sam pour inclure une nouvelle image de base et de nouvelles valeurs d'exécution. Mettez à jour la page des référentiels d'images avec l'RubyURI Amazon ECR 3.2.	6 juin 2023
Ajout d'étapes facultatives pour la vérification de l'intégrité du programme d'installation AWS SAMCLI du package	Mettez à jour la page Installation de la CLI AWS SAM pour refléter l'étape optionnelle. Créez la page Vérifier l'intégrité du programme d'installation de la CLI AWS SAMpour documenter les étapes.	31 mai 2023
Ajout de l'option sam sync pour ignorer la synchronisation de l'infrastructure	Personnalisez si un CloudFormation déploiement est requis à `sam sync` chaque exécution. Pour en savoir plus, voir Ignorer le CloudFormation déploiement initial.	23 mars 2023
Ajout du support pour le type de source d'événement DocumentDB	La spécification du AWS SAM modèle prend désormais en charge le type de source d'`DocumentDB` événement pour la `AWS::Serverless::Function` ressource. Pour en savoir plus, consultez DocumentDB.	10 mars 2023

Créer des fonctions Lambda Rust avec Cargo Lambda	Utilisez la CLI AWS SAM pour créer vos fonctions Lambda Rust à l'aide de Cargo Lambda. Pour en savoir plus, consultez Création de fonctions Lambda Rust avec Cargo Lambda.	23 février 2023
Créez des ressources fonctionnelles en dehors de AWS SAM	Ajout de conseils pour ignorer des fonctions lors de l'utilisation de la commande sam build. Pour en savoir plus, consultez la section Création de fonctions en dehors de AWS SAM.	14 février 2023
Nouvelle syntaxe de connecteurs intégrés	Utilisez la nouvelle syntaxe des connecteurs intégrés pour définir vos ressources `AWS::Serverless::Connector` . Pour en savoir plus, consultez la section Gestion des autorisations relatives aux ressources à l'aide de AWS SAM connecteurs.	8 février 2023
Ajout d'une nouvelle commande sam list pour la CLI AWS SAM	Utilisez sam list pour consulter les informations importantes concernant les ressources de l'application sans serveur. Pour en savoir plus, veuillez consulter liste sam.	2 février 2023

Ajout de propriétés de format et de OutExtension construction pour esbuild	Création de fonctions Lambda Node.js avec esbuild prend désormais en charge `Format` et `OutExtension` . Pour en savoir plus, consultez Création de fonctions Lambda Node.js avec esbuild.	2 février 2023
Ajout d'options de gestion du temps d'exécution à la spécification du AWS SAM modèle	Configurez les options de gestion de l'exécution pour vos fonctions Lambda. Pour en savoir plus, veuillez consulter la section `RuntimeManagementConfig` .	24 janvier 2023
Propriété cible ajoutée à EventSource la AWS::Serverless::StateMachine ressource.	Le type de ressource `AWS::Serverless::StateMachine` prend en charge la propriété `Target` pour les sources d'événements `EventBridgeRule` et `Schedule`.	13 janvier 2023
Configurer la mise à l'échelle des observateurs SQS pour les fonctions Lambda	Configurez la mise à l'échelle des observateurs SQS avec la propriété `ScalingConfig` pour `AWS::Serverless::Function` . Pour en savoir plus, veuillez consulter la section `ScalingConfig` .	12 janvier 2023

Validez AWS SAM les applications avec cfn-lint	Vous pouvez utiliser cfn-lint pour valider vos AWS SAM modèles via le. AWS SAMCLI Pour en savoir plus, veuillez consulter la rubrique Valider avec cfn-lint (langue française non garantie).	11 janvier 2023
Surveillez vos applications sans serveur avec CloudWatch Application Insights	Configurez Amazon CloudWatch Application Insights pour surveiller vos AWS SAM applications. Pour en savoir plus, consultez Surveillez vos applications sans serveur avec CloudWatch Application Insights.	19 décembre 2022
Ajout du programme d'installation de packages de la CLI AWS SAM pour macOS	Installez la CLI AWS SAMà l'aide du nouveau programme d'installation de packages macOS. Pour en savoir plus, consultez la section Installation du AWS SAMCLI.	6 décembre 2022
Ajout du support pour Lambda SnapStart	Configurez SnapStart pour que vos fonctions Lambda créent des instantanés, qui sont des états mis en cache de vos fonctions initialisées. Pour en savoir plus, veuillez consulter la section `AWS::Serverless::Function` .	28 novembre 2022

Ajout de la prise en charge de nodejs18.x par la CLI AWS SAM	La CLI AWS SAM prend désormais en charge l'environnement d'exécution nodejs18.x. Pour en savoir plus, veuillez consulter la rubrique démarrage sam.	17 novembre 2022
Ajout de conseils sur la configuration de l'accès et des autorisations	AWS SAM propose deux options qui simplifient la gestion de l'accès et des autorisations pour vos applications sans serveur. Pour en savoir plus, consultez la section Gestion de l'accès aux ressources et des autorisations.	17 novembre 2022
Ajout de la prise en charge de la création de fonctions Lambda .NET 7 avec la compilation anticipée native	Créez et empaquetez vos fonctions Lambda .NET 7 avec AWS SAM, en utilisant la compilation native Ahead-of-Time (AOT) pour améliorer les temps de démarrage à froid de Lambda. Pour en savoir plus, veuillez consulter la rubrique Création de fonctions Lambda .NET 7 avec la compilation anticipée native (langue française non garantie).	15 novembre 2022

Ajout du AWS SAMCLITerraform support pour le débogage et les tests locaux	Utilisez la CLI AWS SAM dans vos projets Terraform pour effectuer un débogage et des tests locaux de vos fonctions et couches Lambda. Pour en savoir plus, veuillez consulter la section Prise en charge de Terraform pour la CLI AWS SAM.	14 novembre 2022
Ajout du AWS SAM support pour EventBridge Scheduler	La spécification du modèle AWS Serverless Application Model (AWS SAM) fournit une syntaxe simple et abrégée que vous pouvez utiliser pour planifier des événements avec EventBridge Scheduler for et. AWS Lambda AWS Step Functions Pour plus d'informations, consultez la section Planification d'événements avec le EventBridge planificateur.	10 novembre 2022
Simplification des instructions d'installation de la CLI AWS SAM	Les conditions préalables et les étapes facultatives de la CLI AWS SAM ont été déplacés sur des pages distinctes. Les étapes d'installation des systèmes d'exploitation pris en charge se trouvent dans la section Installation du AWS SAMCLI.	4 novembre 2022

Ajout d'un correctif pour autoriser les chemins d'accès longs pour les utilisateurs de Windows 10	Le référentiel de modèles d'applications CLI AWS SAM contient de longs chemins d'accès aux fichiers qui peuvent entraîner des erreurs lors de l'exécution de `sam init` en raison des limitations MAX_PATH de Windows 10. Pour plus d'informations, consultez Installation de la CLI AWS SAM	4 novembre 2022
Processus de déploiement progressif mis à jour pour les premiers déploiements	Le déploiement progressif d'une fonction Lambda AWS CodeDeploy nécessite deux étapes. Pour en savoir plus, veuillez consulter Déploiement progressif d'une fonction Lambda pour la première fois.	13 octobre 2022
Prise en charge supplémentaire du filtrage des événements Lambda pour plusieurs types d'événements	Ajout de la propriété `FilterCritera` aux types de sources d'événements `MSK`, `MQ` et `SelfManagedKafka`.	13 octobre 2022
Ajout du support OpenID Connect (OIDC) pour le pipeline AWS SAM	AWS SAM prend en charge l'authentification utilisateur OpenID Connect (OIDC) pour les plateformes Bitbucket, GitHub Actions et d'intégration GitLab continue et de livraison continue (CI/CD). Pour en savoir plus, consultez la section Utilisation de comptes utilisateur OIDC avec un AWS SAM pipeline.	13 octobre 2022

Remarque sur les JwtConfiguration propriétés	Ajout d'une remarque sur la définition des propriétés `issuer` et `audience` sous `JwtConfiguration` pour `OAuth2Authorizer`.	7 octobre 2022
Nouvelles propriétés pour Function et StateMachine EventSource	Ajout des propriétés `Enabled` et `State` à la source d'événement `CloudWatchEvent` pour `AWS::Serverless::Function`. Ajout de la propriété `State` à la source d'événement `Schedule` pour `AWS::Serverless::Function` et `AWS::Serverless::StateMachine`.	6 octobre 2022
AWS SAM connecteurs désormais généralement disponibles	Les connecteurs sont un type de ressource AWS SAM abstrait, identifié comme tel`AWS::Serverless::Connector`, qui fournit une méthode simple et sécurisée d'attribution d'autorisations entre les ressources de vos applications sans serveur. Pour en savoir plus, consultez la section Gestion des autorisations relatives aux ressources à l'aide de AWS Serverless Application Model connecteurs.	6 octobre 2022
Ajout de nouvelles options de synchronisation sam à la CLI AWS SAM	Ajout des options `--dependency-layer` et `--use-container` à sam sync.	20 septembre 2022

Ajout de nouvelles options de déploiement sam à la CLI AWS SAM	Ajout de l'option `--on-failure` à sam deploy.	9 septembre 2022
Prise en charge d'esbuild désormais généralement disponible	Pour créer et empaqueter les fonctions Lambda de Node.js, vous pouvez les utiliser AWS SAMCLI avec le bundler JavaScript esbuild.	1er septembre 2022
Mise à jour de la télémétrie de la CLI AWS SAM	La description des informations sur le système et l'environnement a été mise à jour pour inclure les valeurs de hachage des attributs d'utilisation.	1er septembre 2022
Ajout de la prise en charge des variables d'environnement local à la CLI AWS SAM	Utilisez des variables d'environnement avec la CLI AWS SAM lorsque vous invoquez des fonctions Lambda localement et lorsque vous exécutez la passerelle API localement.	1er septembre 2022

Support des architectures de jeu d'instructions Lambda	Utilisez laCLI AWS SAM pour créer des fonctions Lambda et des couches Lambda pour les architectures de jeux d'instructions `x86_64` ou `arm64`. Pour plus d'informations, consultez la propriété Architectures du type de `AWS::Serverless::Function` ressource et la CompatibleArchitecturespropriété du type de `AWS::Serverless::LayerVersion` ressource.	1er octobre 2021
Génération d'exemples de configurations de pipeline	Utilisez le AWS SAMCLI pour générer des exemples de pipelines pour plusieurs CI/CD systèmes, à l'aide des `sam pipeline init`commandes new `sam pipeline bootstrap` and. Pour plus d'informations, consultez la section Génération d'exemples de CI/CD pipelines.	21 juillet 2021

Intégration de AWS CDK dans la CLI AWS SAM(version préliminaire, phase 2)	Avec la phase 2 de la version préliminaire publique, vous pouvez désormais utiliser le AWS SAMCLI pour empaqueter et déployer AWS CDK des applications. Vous pouvez également télécharger un exemple d' AWS CDK application directement à l'aide du AWS SAMCLI. Pour de plus amples informations, consultez AWS Cloud Development Kit (AWS CDK) (Version préliminaire).	13 juillet 2021
Prise en charge de RabbitMQ en tant que source d'événements pour les fonctions	Ajout de la prise en charge de RabbitMQ en tant que source d'événements pour les fonctions sans serveur. Pour plus d'informations, consultez la propriété `SourceAccessConfigurations` de la source d'événement `MQ` du type de ressource `AWS::Serverless::Function`.	7 juillet 2021

Déploiement d'applications sans serveur à l'aide d'images de conteneur de création Amazon ECR	Utilisez Amazon ECR pour créer des images de conteneur pour déployer des applications sans serveur avec des systèmes CI/CD courants tels que Jenkins AWS CodePipeline, GitLab CI/CD et Actions. GitHub Pour plus d'informations, consultez Déploiement d'applications sans serveur.	24 juin 2021
Déboguer des AWS SAM applications avec des boîtes à outils AWS	AWS Les boîtes à outils prennent désormais en charge le débogage étape par étape avec davantage de combinaisons d'environ nements de développement intégrés (IDEs) et d'environ nements d'exécution. Pour plus d'informations, consultez la section Utilisation des AWS boîtes à outils.	20 mai 2021
Intégration de AWS CDK dans la CLI AWS SAM(version préliminaire)	Vous pouvez désormais utiliser le AWS SAMCLI pour tester et créer des AWS CDK applications localement. Il s'agit d'une version prélimina ire publique. Pour de plus amples informations, consultez AWS Cloud Development Kit (AWS CDK) (Version prélimina ire).	29 avril 2021

Référentiel d'images de conteneur par défaut changé en Amazon ECR Public	Le référentiel d'images de conteneur par défaut est passé DockerHub d'Amazon ECR Public. Pour plus d'informations, consultez Référentiels d'images.	6 avril 2021
Création CLI AWS SAM nocturne	Vous pouvez désormais installer une version préliminaire du AWS SAMCLI, qui est créée tous les soirs. Pour plus d'informations, consultez la section Nightly build du sous-thème du système d'exploitation de votre choix sous Installation du. AWS SAMCLI	25 mars 2021
Prise en charge des variables d'environnement du conteneur de création	Vous pouvez désormais passer des variables d'environnement pour créer des conteneurs. Pour davantage d'informations, consultez les options `--container-env-var` et `--container-env-var-file` dans `sam build`.	4 mars 2021
Nouveau processus d'installation Linux	Vous pouvez désormais installer la CLI AWS SAM en utilisant un programme d'installation Linux natif. Pour de plus amples informations, consultez Installation de la CLI AWS SAM sur Linux.	10 février 2021

Support pour les files d'attente de lettres mortes pour EventBridge	Ajout de la prise en charge des files d'attente en lettres mortes EventBridge et des sources d'`Schedule`événements pour les fonctions sans serveur et les machines d'état. Pour plus d'informations, consultez la propriété des sources d'événements `DeadLetterConfig` `EventBridgeRule` et `Schedule`, pour les types de ressources `AWS::Serverless::Function` et `AWS::Serverless::StateMachine`.	29 janvier 2021
Prise en charge des points de contrôle personnalisés	Ajout de la prise en charge des points de contrôle personnalisés pour les sources d'événements DynamoDB et Kinesis pour les fonctions sans serveur. Pour plus d'informations, consultez la propriété `FunctionResponseTypes` des types de données `Kinesis` et `DynamoDB` du type de ressource `AWS::Serverless::Function`.	29 janvier 2021

Prise en charge des fenêtres à bascule	Ajout de la prise en charge des fenêtres à bascule pour les sources d'événements DynamoDB et Kinesis pour les fonctions sans serveur. Pour plus d'informations, consultez la propriété `TumblingWindowInSeconds` des types de données `Kinesis` et `DynamoDB` du type de ressource `AWS::Serverless::Function`.	17 décembre 2020
Prise en charge des conteneurs warm	Ajout de la prise en charge des conteneurs à chaud lors des tests locaux en utilisant les commandes `sam local start-api` et `sam local start-lambda` de la CLI AWS SAM. Pour plus d'informations, consultez l'option `--warm-containers` pour ces commandes.	16 décembre 2020
Prise en charge des images de conteneur Lambda	Ajout de la prise en charge pour les images de conteneur Lambda. Pour plus d'informations, consultez Création d'applications.	1er décembre 2020

Prise en charge de la signature de code	Ajout de la prise en charge de la signature de code et des déploiements approuvés de code d'application sans serveur. Pour plus d'informations, consultez la section Configuration de la signature de code pour AWS SAM les applications.	23 novembre 2020
Prise en charge des versions parallèles et en cache	Amélioration des performances des constructions d'applications sans serveur en ajoutant deux options à la commande `sam build` : `--parallel` , qui construit des fonctions et des couches en parallèle plutôt que séquentiellement, et `--cached`, qui utilise des artefacts de construction des versions précédentes lorsqu'aucune modification nécessitant une reconstruction n'a été apportée.	10 novembre 2020

Prise en charge d'Amazon MQ et de l'authentification TLS mutuelle	Ajout de la prise en charge d'Amazon MQ en tant que source d'événements pour les fonctions sans serveur. Pour plus d'informations, consultez les types de données `EventSource` et `MQ` du type de ressource `AWS::Serverless::Function`. La prise en charge de l'authentification mutuelle TLS (Transport Layer Security) a également été ajoutée pour API Gateway APIs et HTTP APIs. Pour plus d'informations, consultez le type de données `DomainConfiguration` du type de ressource `AWS::Serverless::Api`, ou le type de données `HttpApiDomainConfiguration` du type de ressource `AWS::Serverless::HttpApi`.	5 novembre 2020
Support pour les autorisateurs Lambda pour HTTP APIs	Ajout de la prise en charge des mécanismes d'autorisation Lambda pour le type de ressource `AWS::Serverless::HttpApi`. Pour plus d'informations, consultez l'exemple de mécanisme d'autorisation Lambda (`AWS::Serverless::HttpApi`).	27 octobre 2020

Prise en charge de plusieurs fichiers et environnements de configuration	Ajout de la prise en charge de fichiers et environnements de configuration multiples pour le stockage des valeurs de paramètres par défaut pour les commandes de la CLI AWS SAM. Pour plus d'informations, consultez Fichier de configuration de la CLI AWS SAM.	24 septembre 2020
Support pour X-Ray with Step Functions et références pour le contrôle de l'accès à APIs	Ajout de la prise en charge de X-Ray en tant que source d'événements pour les machines d'état sans serveur. Pour plus d'informations, consultez la propriété `Tracing` du type de ressource `AWS::Serverless::StateMachine`. La prise en charge des références a également été ajoutée lors du contrôle de l'accès à APIs. Pour plus d'informations, consultez le type de données `ResourcePolicyStatement`.	17 septembre 2020

Prise en charge d'Amazon MSK	Ajout de la prise en charge pour Amazon MSK en tant que source d'événements pour les fonctions sans serveur. Cela permet aux enregistrements d'une rubrique Amazon MSK de déclencher votre fonction Lambda. Pour plus d'informations, consultez les types de données `EventSource` et `MSK` du type de ressource `AWS::Serverless::Function`.	13 août 2020
Prise en charge d'Amazon EFS	Ajout de la prise en charge pour le montage des systèmes de fichiers Amazon EFS sur des répertoires locaux. Cela permet à votre code de fonction Lambda d'accéder aux ressources partagées et de les modifier. Pour plus d'informations, consultez la propriété `FileSystemConfigs` du type de ressource `AWS::Serverless::Function`.	16 juin 2020

Orchestration d'applications sans serveur	Ajout de la prise en charge pour l'orchestration des applications en créant des machines d'état Step Functions avec AWS SAM. Pour plus d'informations, consultez la section Orchestration AWS des ressources avec AWS Step Functions et le type de `AWS::Serverless::StateMachine` ressource.	27 mai 2020
Création d'environnements d'exécution personnalisés	Ajout de la possibilité de construire des exécutions personnalisées. Pour plus d'informations consultez Création d'exécutions personnalisées.	21 mai 2020
Création de couches	Ajout de la possibilité de construire des ressources `LayerVersion` individuelles. Pour plus d'informations, consultez Construction de couches.	19 mai 2020
CloudFormation Ressources générées	Fourni des détails sur les CloudFormation ressources AWS SAM générées et sur la manière de les référencer. Pour plus d'informations, consultez la section CloudFormation Ressources générées.	8 avril 2020

Configuration des AWS informations d'identification	Ajout d'instructions pour configurer les AWS informations d'identification au cas où vous ne les auriez pas déjà configurées pour être utilisées avec d'autres AWS outils, tels que l'un AWS SDKs ou le AWS CLI. Pour plus d'informations, consultez la section Configuration des AWS informations d'identification.	17 janvier 2020
AWS SAM spécifications et AWS SAMCLI mises à jour	J'ai migré la AWS SAM spécification depuis. GitHub Pour plus d'informations, consultez la Spécification AWS SAM. Mise à jour également du le flux de déploiement avec les modifications apportées à la commande `sam deploy`.	25 novembre 2019

Nouvelles options pour contrôler l'accès à API Gateway APIs et aux mises à jour des modèles de politiques	Ajout de nouvelles options pour contrôler l'accès à API Gateway APIs : autorisations IAM, clés d'API et politiques de ressources. Pour plus d'informations, consultez la section Contrôle de l'accès à API Gateway APIs. Deux modèles de politiques ont également été mis à jour : RekognitionFacesPolicy et ElasticsearchHttpPostPolicy. Pour plus d'informations, consultez les modèles de stratégie AWS SAM.	29 août 2019
Mises à jour du guide de démarrage	Mise à jour du chapitre de démarrage avec des instructions d'installation améliorées pour la CLI AWS SAM et le tutoriel Hello World. Pour plus d'informations, consultez la section Mise en route avec AWS SAM.	25 juillet 2019
Contrôle de l'accès à API Gateway APIs	Ajout de la prise en charge du contrôle de l'accès à API Gateway APIs. Pour plus d'informations, consultez la section Contrôle de l'accès à API Gateway APIs.	21 mars 2019

Ajout de `sam publish` à la CLI AWS SAM	La nouvelle commande `sam publish` de l'interface de la CLI AWS SAM simplifie le processus de publication des applications sans serveur dans le AWS Serverless Application Repository. Pour plus d'informations, consultez la section Publication d'applications sans serveur à l'aide du AWS SAMCLI.	21 décembre 2018
Prise en charge des applications imbriquées et des couches	Ajout de la prise en charge des applications imbriquées et des couches. Pour plus d'informations, consultez Utilisation des applications imbriquées et Utilisation des couches.	29 novembre 2018
Ajout de `sam build` à la CLI AWS SAM	La nouvelle commande `sam build` de la CLI AWS SAM simplifie le processus de compilation des applications sans serveur avec des dépendances afin que vous puissiez tester et déployer localement ces applications. Pour plus d'informations, consultez Création d'applications.	19 novembre 2018

Ajout de nouvelles options d'installation pour la CLI AWS SAM	Ajout des options d'installation de Linuxbrew (Linux), MSI (Windows) et Homebrew (macOS) pour le. AWS SAMCLI Pour plus d'informations, consultez la section Installation du AWS SAMCLI.	7 novembre 2018
Nouveau guide	Il s’agit de la première version du Guide du développeur AWS Serverless Application Model .	17 octobre 2018

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.