AWS Serverless Application Repository

Manuel du développeur

AWS Serverless Application Repository: Manuel du développeur

Table of Contents

Présentation d'AWS Serverless Application Repository

La .AWS Serverless Application Repository, permet aux développeurs et aux entreprises de trouver, déployer et publier facilement et rapidement les applications sans serveur dans leAWSCloud. Pour plus d'informations sur les applications sans serveur, consultezInformatique et applications sans serveursur leAWSSite Web.

Vous pouvez publier facilement des applications, les partager publiquement avec la communauté dans son ensemble ou en privé avec votre équipe ou votre entreprise. Pour publier une application sans serveur (ou application), vous pouvez utiliserAWS Management Console, leAWS SAMInterface de ligne de commande (AWS SAMCLI), ouAWSSDK pour télécharger votre code. En même temps que votre code, vous chargez un fichier manifeste simple, aussi appelé modèle AWS Serverless Application Model (AWS SAM). Pour plus d'informations sur AWS SAM, consultez le Manuel du développeur AWS Serverless Application Model.

AWS Serverless Application Repository est complètement intégré à la console AWS Lambda. Cette intégration signifie que les développeurs de tous les niveaux peuvent démarrer dans l'informatique sans serveur sans avoir à apprendre quelque chose de nouveau. Vous pouvez utiliser les mots-clés de la catégorie pour rechercher des applications : par exemple, backends web et mobiles, applications de traitement de données ou chatbots. Vous pouvez aussi rechercher les applications par nom, éditeur ou source d'événement. Pour utiliser une application, vous la choisissez, configurez les champs requis et la déployez en quelques clics.

Dans ce guide, vous allez découvrir les deux façons d'utiliser AWS Serverless Application Repository :

- Publication des applications (p. 7) – Configurez et chargez les applications pour les rendre accessibles à d'autres développeurs, puis publiez de nouvelles versions des applications.
- Déploiement d'applications (p. 23) – Recherchez les applications et affichez les informations à leur sujet, y compris les fichiers du code source et le fichier Lisez-moi. De même, installez, configurez et déployez les applications de votre choix.

Étapes suivantes

- Pour obtenir un didacticiel sur la publication d'un exemple d'application dans leAWS Serverless Application Repository, consultezQuick Start : Publication des applications (p. 2).
- Pour obtenir des instructions sur le déploiement d'applications à partir de laAWS Serverless Application Repository, consultezComment déployerApplications (p. 25).

Quick Start : Publication des applications

Ce guide vous accompagne à travers les étapes de téléchargement, de création, de test et de publication d'un exemple d'application sans serveur dans AWS Serverless Application Repository en utilisant l'interface de ligne de commande AWS SAM. Vous pouvez utiliser cet exemple d'application comme point de départ pour développer et publier votre propre application sans serveur.

Overview

Les étapes suivantes décrivent comment télécharger, créer et publier un exemple d'application sans serveur :

1. Initialiser. Téléchargez un exemple d'application à partir du modèle à l'aide de `sam init`.
2. Testez localement. Testez l'application localement en utilisant `sam local invoke` et/ou `sam local start-api`. Notez qu'avec ces commandes, même si votreLambdaest invoquée localement, elle lit depuis et écrit dansAWSRessources dans leAWSCloud.
3. Package. Lorsque vous êtes satisfait de votre fonction Lambda, regroupez la fonction Lambda, le modèle AWS SAM et toutes les dépendances dans un package de déploiement AWS CloudFormation à l'aide de `sam package`. Dans cette étape, vous allez également inclure des informations sur l'application qui sera téléchargée vers AWS Serverless Application Repository.
4. Publiez. Publiez l'application sur AWS Serverless Application Repository en utilisant `sam publish`. À la fin de cette étape, vous êtes en mesure de visualiser votre application dansAWS Serverless Application Repositoryet déployez-le sur le serveurAWSCloud utilisantAWS Serverless Application Repository.

L'exemple Application Hello World (p. 2) à la section suivante vous guide à travers ces étapes de création et de publication d'une application sans serveur.

Application Hello World

Dans cet exercice, vous téléchargez et testez une application Hello World sans serveur qui représente un back-end d'API simple. Il a un point de terminaison Amazon API Gateway qui prend en charge une opération GET et une fonction Lambda. Lorsqu'une requête GET est envoyée au point de terminaison, API Gateway appelle la fonction Lambda. Ensuite, AWS Lambda exécute la fonction, qui renvoie simplement un message `hello world`.

L'application comporte les composants suivants :

- UnAWS SAMqui définit deuxAWSpour l'application Hello Word : unAPI Gatewayavec une opération GET, et unLambda. Le modèle définit également le mappage entre l'opération API Gateway GET et la fonction Lambda.
- Code d'application écrit en Python.

Avant de commencer

Assurez-vous que vous avez la configuration requise pour cet exercice :

- Vous devez avoir unAWSavec un utilisateur IAM disposant des autorisations d'administrateur. VoirConfiguration d'unAWSCompte.
- Vous devez avoir l'interface de ligne de commande AWS SAM installée. Veuillez consulter Installation de l'interface de ligne de commande AWS SAM.
- Vous devez disposer de la version 1.16.77 ou ultérieure de la version AWS CLI installée. Veuillez consulter Installation de AWS Command Line Interface.

Étape 1 : Initialiser l'application

Dans cette section, vous téléchargez l'exemple d'application, qui se compose d'un modèle AWS SAM et d'un code d'application.

Pour initialiser l'application

1. Exécutez la commande suivante à l'invite de commande de l'interface de ligne de commande AWS SAM.

   ```
   sam init --runtime python3.6
   ```

2. Vérifiez le contenu du répertoire créé par la commande (`sam-app/`) :

   - `template.yaml` –Définit deuxAWSRessource dont l'application Hello World a besoin : unLambdaet unAPI Gatewayqui prend en charge une opération GET. Le modèle définit également le mappage entre les deux ressources.
   - Contenu lié au code de l'application Hello World :
     - Répertoire `hello_world/` – Contient le code de l'application, qui renvoie `hello world` lorsque vous l'exécutez.

     Note

     Pour cet exercice, le code de l'application est écrit en Python et vous spécifiez le moteur d'exécution dans la commande `init`. AWS Lambda prend en charge des langues supplémentaires pour créer du code d'application. Si vous spécifiez un autre moteur d'exécution pris en charge, la commande `init` fournit le code Hello World dans le langage spécifié et un fichier `README.md` que vous pouvez suivre pour ce langage. Pour de plus amples informations sur les runtimes pris en charge, veuillez consulter Environnement d'exécution Lambda et bibliothèques disponibles.

Étape 2 : Tester l'application localement

Maintenant que vous avez l'application AWS SAM sur votre machine locale, suivez les étapes ci-dessous pour la tester localement.

Pour tester l'application localement

1. Démarrez le point de terminaison API Gateway localement. Vous devez exécuter la commande suivante à partir du répertoire qui contient le fichier `template.yaml`.

```
sam-app> sam local start-api --region us-east-1
```

La commande renvoie un point de terminaison API Gateway auquel vous pouvez envoyer des demandes de test local.

2. Testez l'application. Copiez l'URL du point de terminaison API Gateway, collez-la dans le navigateur et choisissez Entrée. Un exemple d'URL de point de terminaison API Gateway est `http://127.0.0.1:3000/hello`.

   API Gateway appelle localement la fonction Lambda à laquelle le point de terminaison est mappé. La fonction Lambda s'exécute dans le conteneur Docker local et retourne `hello world`. API Gateway renvoie une réponse au navigateur qui contient le texte.

Exercice Modifier la chaîne de message

Après avoir testé avec succès l'exemple d'application, vous pouvez expérimenter avec une simple modification : modifier la chaîne de message renvoyée.

1. Modifiez le fichier `/hello_world/app.py` pour remplacer la chaîne de message `'hello world'` par `'Hello World!'`.
2. Rechargez l'URL de test dans votre navigateur et observez la nouvelle chaîne.

Vous remarquerez que votre nouveau code est chargé dynamiquement, sans que vous ayez redémarré le processus `sam local`.

Étape 3 : Création d'un package de l'application

Après avoir testé votre application localement, vous utilisez l'interface de ligne de commande AWS SAM pour créer un package de déploiement et un modèle AWS SAM empaqueté.

> **Note**
>
> Au cours des étapes suivantes, vous créez un fichier .zip pour le contenu du répertoire `hello_world/`, qui contient le code de l'application. Ce fichier .zip est le package de déploiement de votre application sans serveur. Pour de plus amples informations, veuillez consulter Création d'un package de déploiement (Python) dans le AWS Lambda Developer Guide.

Pour créer un package de déploiement Lambda

1. Ajoutez une section `Metadata` à votre fichier de modèle AWS SAM fournissant les informations requises sur l'application. Pour de plus amples informations sur la section `Metadata` des modèles AWS SAM, veuillez consulter Propriétés de la section des métadonnées de modèles AWS SAM dans le Guide du développeur AWS Serverless Application Model.

   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
```

Les propriétés `LicenseUrl` et `ReadmeUrl` peuvent être soit des références à des fichiers locaux (comme dans l'exemple ci-dessus), soit des liens vers des compartiments Amazon S3 qui hébergent déjà ces artefacts.

2. 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.

```
sam-app> aws s3 mb s3://bucketname
```

3. Créez le package de déploiement des fonctions Lambda en exécutant la commande d'interface de ligne de commande `package` AWS SAM suivante.

```
sam-app> sam package \
    --template-file template.yaml \
    --output-template-file packaged.yaml \
    --s3-bucket bucketname
```

La commande exécute les opérations suivantes :

- Zippe le contenu du répertoire `aws-sam/hello_world/` et le télécharge vers Amazon S3.
- Télécharge le package de déploiement, le fichier README et le fichier LICENCE dans le compartiment Amazon S3 spécifié par l'option `--s3-bucket`.
- Affiche un nouveau fichier modèle, appelé `packaged.yaml`, que vous utilisez à l'étape suivante pour publier l'application AWS Serverless Application Repository. Le fichier de modèle `packaged.yaml` est similaire au fichier de modèle d'origine (`template.yaml`), mais a une différence clé — les propriétés `CodeUri`, `LicenseUrl` et`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` :

```
HelloWorldFunction:
    Type: AWS::Serverless::Function # For more information about function
 resources, see https://github.com/awslabs/serverless-application-model/blob/master/
versions/2016-10-31.md#awsserverlessfunction
    Properties:
      CodeUri: s3://bucketname/fbd77a3647a4f47a352fcObjectGUID

...
```

Étape 4 : Publier l'application

Maintenant que vous avez créé le package de déploiement, vous l'utilisez pour publier l'application sur AWS Serverless Application Repository.

Pour publier l'application sans serveur dans le répertoire AWS Serverless Application Repository

- Exécutez la commande suivante pour publier la nouvelle application AWS Serverless Application Repository avec la première version créée en tant que 0.0.1.

```
sam-app> sam publish \
    --template packaged.yaml \
    --region us-east-1
```

Note

L'application sera créée comme privée par défaut. Vous devez partager l'application avant d'autresAWSseront autorisés à afficher et à déployer votre application. Consultez Étapes suivantes ci-dessous pour plus de détails sur le partage de votre application.

Étapes suivantes

Maintenant que vous avez publié votre exemple d'application, voici comme vous pouvez l'utiliser.

- Afficher votre application dans AWS Serverless Application Repository – La sortie de la commande `sam publish` inclura un lien vers le AWS Serverless Application Repository, directement vers la page détaillée de votre application. Vous pouvez également accéder à la page de destination AWS Serverless Application Repository et rechercher votre application.
- Partagez votre application –Parce que votre application est définie sur privée par défaut, elle n'est pas visible par les autresAWSLes comptes. Pour partager votre application avec d'autres utilisateurs, vous devez soit la rendre publique, soit accorder l'autorisation d'accéder à une liste spécifique deAWSLes comptes. Pour de plus amples informations sur le partage de votre application à l'aide de AWS CLI, veuillez consulter Exemples de stratégies basées sur les ressources AWS Serverless Application Repository (p. 46). Pour de plus amples informations sur le partage de votre application à l'aide de la console, veuillez consulter Suppression d'une application (p. 16).

En savoir plus

Pour de plus amples informations sur la section `Metadata` des modèles AWS SAM, et les commandes `sam package` et `sam publish` de l'interface de ligne de commande AWS SAM, veuillez consultez Publication d'applications à l'aide de l'interface de ligne de commande AWS SAM dans le Guide du développeur AWS Serverless Application Model.

Publication des applications

Lorsque vous publiez une application sans serveur sur le AWS Serverless Application Repository, vous la mettez à la disposition d'autres utilisateurs pour qu'ils puissent la rechercher et le déployer.

Vous définissez d'abord votre application avec un modèleAWS Serverless Application Model (AWS SAM). Lorsque vous définissez votre application, vous devez déterminer si ses consommateurs seront tenus de reconnaître les capacités de l'application. Pour de plus amples informations sur l'utilisation de AWS SAM et la reconnaissance des capacités, veuillez consulter Utilisation d'AWS SAM avec l'AWS Serverless Application Repository (p. 7).

Vous pouvez publier des applications sans serveur à l'aide de l'AWS Management Console, leAWS SAMInterface de ligne de commande (AWS SAMCLI), ou unAWSKIT SDK. Pour de plus amples informations sur les procédures de publication d'applications dans le AWS Serverless Application Repository, veuillez consulter HowPublication des applications (p. 12).

Lorsque vous publiez votre application, elle est initialement définie surprivé, ce qui signifie qu'il n'est disponible que pourAWSqui l'a créé. Pour partager votre application avec d'autres utilisateurs, vous devez la définir surPartage privé(partagé uniquement avec un ensemble spécifique deAWScomptes), ouPartage publiquement(partagé avec tout le monde).

Lorsque vous publiez une application dans le AWS Serverless Application Repository et que vous la définissez sur publique, le service rend l'application accessible aux consommateurs de toutes les régions. Lorsqu'un consommateur déploie une application publique dans une région autre que celle dans laquelle l'application a été publiée pour la première fois, AWS Serverless Application Repository copie les artefacts de déploiement de l'application dans un compartiment Amazon S3 dans la région de destination. Il met à jour toutes les ressources du modèle AWS SAM qui utilisent ces artefacts pour référencer les fichiers dans le compartiment Amazon S3 de la région de destination. Les artefacts de déploiement peuvent inclure le code de fonction Lambda, les fichiers de définition d'API, etc.

> Note
>
> PrivéandPartage privéLes applicationsdisponibles uniquement dansAWSRégion dans laquelle ils sont créés. Partage publiquementLes applicationsdisponibles dans toutes lesAWSRégions concernées. Pour de plus amples informations sur le partage d'applications, veuillez consulter Exemples de stratégies basées sur les ressources AWS Serverless Application Repository (p. 46).

Rubriques

Utilisation d'AWS SAM avec l'AWS Serverless Application Repository

La .AWS Serverless Application Model(AWS SAM) est un cadre open source que vous pouvez utiliser pour construireapplications sans serveursurAWS. Pour plus d'informations sur l'utilisation d'AWS SAM pour

construire votre application sans serveur, consultez le Guide du développeur AWS Serverless Application Model.

Lors de la création d'applications qui seront publiées dans leAWS Serverless Application Repository, vous devez considérer l'ensemble deprises en chargeAWSRessources et modèles de stratégie disponibles à l'utilisation. Les sections ci-dessous décrivent ces sujets plus en détail.

prises en chargeAWSRessources dans leAWS Serverless Application Repository

Le AWS Serverless Application Repository prend en charge les applications sans serveur qui sont composées de nombreuses ressources AWS SAM et AWS CloudFormation. Pour afficher la liste complète desAWSqui sont prises en charge parAWS Serverless Application Repositoryvoir,Liste des prises en chargeAWSRessources (p. 8).

Si vous souhaitez demander du support pour uneAWSRessource, contactAWSSupport.

> Important
>
> Si votre modèle d'application contient un ou plusieurs rôles IAM ou stratégies de ressources personnalisés, votre application n'apparaît pas par défaut dans les résultats de la recherche. Les clients doivent également confirmer les rôles IAM ou les stratégies de ressources personnalisés pour pouvoir déployer l'application. Pour plus d'informations, consultez Confirmation des capacités d'une application (p. 24).
> La liste des ressources impactées est la suivante :
>
> - Rôles IAM : AWS::IAM::Group,AWS::IAM::InstanceProfile,AWS::IAM::Policy, etAWS::IAM::Role.
> - Stratégies liées aux ressources : AWS::Lambda::LayerVersionPermission,AWS::Lambda::Permission,AWS::Events::EventBusPolicy,AWS# IAM:Politique,AWS::ApplicationAutoScaling::ScalingPolicy,AWS::S3::BucketPolicy,AWS::SQS::QueuePolicy, etAWS# SNS:TopicPolicy.
>
> Si votre application contient la ressource AWS::Serverless::Application, les clients doivent confirmer que l'application contient une application imbriquée pour pouvoir déployer l'application. Pour plus d'informations sur les applications imbriquées, consultez la rubrique relative aux applications imbriquées dans le Guide du développeur AWS Serverless Application Model. Pour de plus amples informations sur la confirmation des capacités, veuillez consulter Confirmation des capacités d'une application (p. 24).

Modèles de stratégie

AWS SAM vous permet de choisir parmi une liste de modèles de stratégie afin de restreindre les autorisations de vos fonctions Lambda aux ressources utilisées par votre application. L'utilisation de modèles de stratégie ne nécessite pas d'accusés de réception supplémentaires pour rechercher, parcourir ou déployer l'application.

Pour obtenir la liste des modèles de stratégie AWS SAM standard, veuillez consulter Modèles de stratégie AWS SAM dans le Guide du développeur AWS Serverless Application Model.

Liste des prises en chargeAWSRessources

Il s'agit de la liste complète desAWSqui sont prises en charge par leAWS Serverless Application Repository.

- `AWS::AccessAnalyzer::Analyzer`

- AWS::ApiGateway::ApiKey
- AWS::ApiGateway::Authorizer
- AWS::ApiGateway::BasePathMapping
- AWS::ApiGateway::ClientCertificate
- AWS::ApiGateway::Deployment
- AWS::ApiGateway::DocumentationPart
- AWS::ApiGateway::DocumentationVersion
- AWS::ApiGateway::DomainName
- AWS::ApiGateway::GatewayResponse
- AWS::EC2::SecurityGroup
- AWS::EC2::SecurityGroupEgress
- AWS::EC2::SecurityGroupIngress
- AWS::ApiGateway::Method
- AWS::ApiGateway::Model
- AWS::ApiGateway::RequestValidator
- AWS::ApiGateway::Resource
- AWS::ApiGateway::RestApi
- AWS::ApiGateway::Stage
- AWS::ApiGateway::UsagePlan
- AWS::ApiGateway::UsagePlanKey
- AWS::ApiGateway::VpcLink
- AWS::ApiGatewayV2::Api
- AWS::ApiGatewayV2::ApiMapping
- AWS::ApiGatewayV2::Authorizer
- AWS::ApiGatewayV2::DomainName
- AWS::ApiGatewayV2::Deployment
- AWS::ApiGatewayV2::Integration
- AWS::ApiGatewayV2::IntegrationResponse
- AWS::ApiGatewayV2::Model
- AWS::ApiGatewayV2::Route
- AWS::ApiGatewayV2::RouteResponse
- AWS::ApiGatewayV2::Stage
- AWS::AppSync::ApiKey
- AWS::AppSync::DataSource
- AWS::AppSync::GraphQLApi
- AWS::AppSync::GraphQLSchema
- AWS::AppSync::Resolver
- AWS::ApplicationAutoScaling::ScalableTarget
- AWS::ApplicationAutoScaling::ScalingPolicy
- AWS::Athena::NamedQuery
- AWS::CertificateManager::Certificate
- AWS::CloudFormation::CustomResource
- AWS::CloudFormation::Interface

- `AWS::CloudFormation::Macro`
- `AWS::CloudFormation::WaitConditionHandle`
- `AWS::CloudFront::CloudFrontOriginAccessIdentity`
- `AWS::CloudFront::Distribution`
- `AWS::CloudFront::StreamingDistribution`
- `AWS::CloudTrail::Trail`
- `AWS::CloudWatch::Alarm`
- `AWS::CloudWatch::AnomalyDetector`
- `AWS::CloudWatch::Dashboard`
- `AWS::CloudWatch::InsightRule`
- `AWS::CodeBuild::Project`
- `AWS::CodeCommit::Repository`
- `AWS::CodePipeline::CustomActionType`
- `AWS::CodePipeline::Pipeline`
- `AWS::CodePipeline::Webhook`
- `AWS::Cognito::IdentityPool`
- `AWS::Cognito::IdentityPoolRoleAttachment`
- `AWS::Cognito::UserPool`
- `AWS::Cognito::UserPoolClient`
- `AWS::Cognito::UserPoolDomain`
- `AWS::Cognito::UserPoolGroup`
- `AWS::Cognito::UserPoolResourceServer`
- `AWS::Cognito::UserPoolUser`
- `AWS::Cognito::UserPoolUserToGroupAttachment`
- `AWS::Config::AggregationAuthorization`
- `AWS::Config::ConfigRule`
- `AWS::Config::ConfigurationAggregator`
- `AWS::Config::ConfigurationRecorder`
- `AWS::Config::DeliveryChannel`
- `AWS::Config::RemediationConfiguration`
- `AWS::DataPipeline::Pipeline`
- `AWS::DynamoDB::Table`
- `AWS::ECR::Repository`
- `AWS::Elasticsearch::Domain`
- `AWS::Events::EventBusPolicy`
- `AWS::Events::Rule`
- `AWS::EventSchemas::Discoverer`
- `AWS::EventSchemas::Registry`
- `AWS::EventSchemas::Schema`
- `AWS::Glue::Classifier`
- `AWS::Glue::Connection`
- `AWS::Glue::Crawler`

- AWS::Glue::Database
- AWS::Glue::DevEndpoint
- AWS::Glue::Job
- AWS::Glue::Partition
- AWS::Glue::Table
- AWS::Glue::Trigger
- AWS::IAM::Group
- AWS::IAM::InstanceProfile
- AWS::IAM::ManagedPolicy
- AWS::IAM::Policy
- AWS::IAM::Role
- AWS::IoT::Certificate
- AWS::IoT::Policy
- AWS::IoT::PolicyPrincipalAttachment
- AWS::IoT::Thing
- AWS::IoT::ThingPrincipalAttachment
- AWS::IoT::TopicRule
- AWS::KMS::Alias
- AWS::KMS::Key
- AWS::Kinesis::Stream
- AWS::Kinesis::StreamConsumer
- AWS::Kinesis::Streams
- AWS::KinesisAnalytics::Application
- AWS::KinesisAnalytics::ApplicationOutput
- AWS::KinesisFirehose::DeliveryStream
- AWS::Lambda::Alias
- AWS::Lambda::EventInvokeConfig
- AWS::Lambda::EventSourceMapping
- AWS::Lambda::Function
- AWS::Lambda::LayerVersion
- AWS::Lambda::LayerVersionPermission
- AWS::Lambda::Permission
- AWS::Lambda::Version
- AWS::Logs::Destination
- AWS::Logs::LogGroup
- AWS::Logs::LogStream
- AWS::Logs::MetricFilter
- AWS::Logs::SubscriptionFilter
- AWS::Route53::HealthCheck
- AWS::Route53::HostedZone
- AWS::Route53::RecordSet
- AWS::Route53::RecordSetGroup
- AWS::S3::Bucket

- `AWS::S3::BucketPolicy`
- `AWS::SNS::Subscription`
- `AWS::SNS::Topic`
- `AWS::SNS::TopicPolicy`
- `AWS::SQS::Queue`
- `AWS::SQS::QueuePolicy`
- `AWS::SSM::Association`
- `AWS::SSM::Document`
- `AWS::SSM::MaintenanceWindowTask`
- `AWS::SSM::Parameter`
- `AWS::SSM::PatchBaseline`
- `AWS::SSM::ResourceDataSync`
- `AWS::Serverless::Api`
- `AWS::Serverless::Application`
- `AWS::Serverless::Function`
- `AWS::Serverless::HttpApi`
- `AWS::Serverless::LayerVersion`
- `AWS::Serverless::SimpleTable`
- `AWS::Serverless::StateMachine`
- `AWS::StepFunctions::Activity`
- `AWS::StepFunctions::StateMachine`

HowPublication des applications

Cette section fournit des procédures de publication de votre application sans serveur vers AWS Serverless Application Repository à l'aide de l'interface de ligne de commande AWS SAM ou de l'AWS Management Console. Elle vous montre également comment partager votre application pour permettre à d'autres utilisateurs de la déployer et à la supprimer de AWS Serverless Application Repository.

> Important
>
> Les informations que vous entrez lorsque vous publiez une application ne sont pas chiffrées. Ces informations comprennent des données telles que le nom de l'auteur. Si vous disposez d'informations personnelles identifiables que vous ne souhaitez pas stocker ou rendre publiques, nous vous recommandons de ne pas les saisir lors de la publication de votre application.

Publication d'une application (AWS CLI)

La façon la plus simple de publier une application sur AWS Serverless Application Repository consiste à utiliser un ensemble de commandes d'interface de ligne de commande AWS SAM. Pour de plus amples informations, veuillez consulter Publication d'une application à l'aide de l'interface de ligne de commande AWS SAM dans le Guide du développeur AWS Serverless Application Model (AWS SAM).

Publication d'une nouvelle application (console)

Cette section vous montre comment utiliser l'AWS Management Console pour publier une nouvelle application dans AWS Serverless Application Repository. Pour obtenir des instructions sur la publication

d'une nouvelle version d'une application existante, veuillez consulter Publication d'une nouvelle version d'une application existante (p. 19).

Prerequisites

Avant de publier une application sur AWS Serverless Application Repository, vous avez besoin des éléments suivants :

- Un rapport valideAWS.
- Un rapport valideAWS Serverless Application Model(AWS SAM) qui définit l'AWSqui sont utilisées. Pour de plus amples informations sur les modèles AWS SAM, veuillez consulter Concepts de base des modèles AWS SAM.
- Un package pour votre application que vous avez créé à l'aide de la commande AWS CloudFormation `package` pour l'AWS CLI. Cette commande empaquette les artefacts locaux (chemins locaux) auxquels votre modèle AWS SAM fait référence. Pour plus d'informations, consultez package dans la documentation AWS CloudFormation.
- Une URL qui pointe vers le code source de votre application, si vous souhaitez publier votre application publiquement.
- Un fichier readme.txt. Ce fichier doit décrire comment les clients peuvent utiliser votre application et comment la configurer avant de la déployer dans son propreAWS.
- Un fichier license.txt ou un identifiant de licence valide du site web SPDX. Notez qu'une licence n'est requise que si vous souhaitez partager votre application publiquement. Si vous voulez garder votre application privée ou la partager uniquement en privé, vous n'avez pas besoin de spécifier de licence.
- Une stratégie de compartiment Amazon S3 valide qui accorde les autorisations de lecture du service pour artefacts chargés sur Amazon S3 quand vous avez empaqueté votre application. Pour définir cette stratégie, procédez comme suit :

  1. Ouvrez la console Amazon S3 à l'adresse https://console.aws.amazon.com/s3/.
  2. Choisissez le compartiment Amazon S3 que vous avez utilisé pour créer le package de votre application.
  3. Choisissez l'onglet Autorisations.
  4. Choisissez le bouton Stratégie de compartiment.
  5. Collez l'instruction de stratégie suivante dans l'éditeur de stratégie de compartiment. Assurez-vous de remplacer le nom de votre compartiment dans la`Resource`, et votreAWSID de compte dans la`Condition`Élément. L'expression dans la`Condition`Élément assurerAWS Serverless Application Repositorya uniquement l'autorisation d'accéder aux applications à partir de laAWS. Pour plus d'informations sur les instructions de stratégie, consultezIAMRéférence des éléments de stratégie JSONdans leIAM Guide de l'utilisateur.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service":  "serverlessrepo.amazonaws.com"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::bucketname/*",
            "Condition" : {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                }
            }
        }
    ]
```

```
}
```

6. Choisissez le bouton Enregistrer.

Procedure

Créez une application dans AWS Serverless Application Repository à l'aide de la procédure suivante.

Pour créer une application dans AWS Serverless Application Repository

1. Ouvrez la console AWS Serverless Application Repository et choisissez Publish applications (Publier les applications).
2. Sur la page Publier une application entrez les informations d'application suivantes, puis choisissez Publier une application :

Propriété	Obligatoire	Description
Nom de l'application	TRUE	Nom de l'application . Longueur minimale = 1. Longueur maximale = 140. Modèle : [a-zA-Z0-9\-]+
Author (Auteur)	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]) ? $
Page d'accueil	FALSE	URL contenant plus d'informations sur l'application —par exemple, l'emplacement de votre dépôt GitHub pour l'application.
Description	TRUE	Description de l'application. Longueur minimale = 1. Longueur maximale = 256.
Etiquettes	FALSE	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+\\-_:\\/@]+ $
Licence Spdx (liste déroulante)	FALSE	Choisissez un identificateur de licence valide dans la

Propriété	Obligatoire	Description
		liste déroulante qui contient les licences disponibles sur le site web SPDX. Le choix d'un élément dans la liste déroulante remplit la zone de texte Licence située en dessous de celui-ci. Remarque : Le choix d'une licence dans la liste déroulante remplace le contenu de laLicenceet rejette toute modification manuelle que vous avez effectuée.
Licence	FALSE	Téléchargez un fichier de licence .txt ou choisissez une licence dans la liste déroulante de licence Spdx décrite dans la ligne précédente. Le choix d'une licence dans la liste déroulante Licence Spdx remplit automatiquement la zone de texte Licence. Vous pouvez modifier manuellement le contenu de cette zone de texte après avoir téléchargé un fichier de licence ou en avoir choisi un dans la liste déroulante Licence Spdx. Toutefois, si une autre licence Spdx est choisie dans la liste déroulante, toutes les modifications manuelles que vous avez effectuées sont supprimées. Il s'agit d'un champ d'option, mais vous devez fournir une licence afin de partager l'application publiquement.
Readme	FALSE	Téléchargez le contenu du fichier Readme, qui peut être au format texte ou balisage. Ces contenus sont affichés sur la page détaillée de l'application dans le AWS Serverless Application Repository. Vous pouvez modifier manuellement le contenu de cette zone de texte après avoir téléchargé un fichier.

Propriété	Obligatoire	Description
Version sémantique	FALSE	Version sémantique de l'application. Pour de plus amples informations, veuillez consulter le site web Contrôle de version sémantique. Vous devez fournir une valeur pour cette propriété afin de rendre votre application publique.
URL du code source	FALSE	Lien vers un référentiel public pour le code source de votre application.
SAM template (Modèle SAM)	TRUE	UnAWS Serverless Application Model(AWS SAM) qui définit l'AWSqui sont utilisées.

Suppression d'une application

Les applications publiées peuvent disposer d'autorisations définies dans l'une des trois catégories suivantes :

- Privé (par défaut) –Applications qui ont été créées avec le même compte et qui n'ont pas été partagées avec d'autresAWS. Seuls les consommateurs qui partagent votreAWSont l'autorisation de déployer des applications privées.
- Partagé en privé –Applications que l'éditeur a explicitement partagées avec un ensemble spécifique deAWS, ou avecAWSdans unAWSOrganisation. Les consommateurs sont autorisés à déployer des applications qui ont été partagées avec leurAWSouAWSOrganisation. Pour plus d'informations sur AWS Organizations, consultez le Manuel de l'utilisateur AWS Organizations.
- Partagé publiquement – Applications que l'éditeur a partagées avec tout le monde. Tous les consommateurs ont l'autorisation de déployer n'importe quelle application partagée publiquement.

Une fois que vous avez publié une application sur AWS Serverless Application Repository, elle est définie par défaut comme privée. Cette section vous montre comment partager une application en privé avec desAWSou unAWSOrganisation, ou partager publiquement avec tout le monde.

Partage d'une application via la console

Vous avez deux options pour partager votre application avec d'autres : 1) Partagez-le avec desAWSou les comptesAWSau sein de votreAWSou 2) Partagez publiquement avec tout le monde. Pour plus d'informations sur AWS Organizations, consultez le Manuel de l'utilisateur AWS Organizations.

Option 1 : Pour partager votre application avec desAWScompte (s) ou compte (s) au sein de votreAWSorganisation

1. Ouvrez la console AWS Serverless Application Repository .
2. Dans le panneau de navigation, choisissez Published Applications (Applications publiées) pour afficher la liste des applications que vous avez créées.
3. Choisissez l'application que vous souhaitez partager.

4. Cliquez sur l'onglet Sharing (Partager) .
5. Dans la section Application policy statements (Déclarations de stratégie d'application), cliquez sur le bouton Create Statement (Créer une déclaration).
6. Dans la fenêtre Statement Configuration (Configuration de la déclaration), remplissez les champs en fonction de la façon dont vous souhaitez partager votre application.

   Note

   Si vous partager avec une organisation, vous pouvez uniquement spécifier l'organisation que votreAWSest un membre de. Si vous essayez de spécifier unAWSOrganisation dont vous n'êtes pas membre, une erreur se produit.
   Pour partager votre application avec votreAWSOrganisation, vous devez reconnaître que le`UnshareApplication`sera ajoutée à votre déclaration de stratégie, au cas où le partage doit être révoqué à l'avenir.
7. Choisissez le bouton Enregistrer.

Option 2 : Partagé publiquement votre application avec tout le monde

1. Ouvrez la console AWS Serverless Application Repository .
2. Dans le panneau de navigation, choisissez Published Applications (Applications publiées) pour afficher la liste des applications que vous avez créées.
3. Choisissez l'application que vous souhaitez partager.
4. Cliquez sur l'onglet Sharing (Partager) .
5. Dans la section Public Sharing (Partage public) cliquez sur le bouton Edit (Modifier) .
6. Sous Public sharing (Partage public), choisissez le bouton radio Enabled (Activé) .
7. Dans la zone de texte, saisissez le nom de votre application, puis cliquez sur le bouton Save (Enregistrer) .

   Note

   Pour partager publiquement une application, elle doit avoir à la fois définies les propriétés `LicenseUrl` et `SemanticVersion`.

Partage d'une application via l'AWS CLI

Pour partager une application à l'aide de l'outilAWS CLIvous accordez des autorisations à l'aide du`put-application-policy`pour spécifier la commandeAWSCompte (s) que vous souhaitez partager en tant que mandants.

Pour plus d'informations sur le partage de votre application à l'aide de l'AWSCLI, veuillez consulterExemples de stratégies basées sur les ressources AWS Serverless Application Repository (p. 46).

Annuler le partage d'une application

Il existe deux options pour annuler le partage d'une application d'uneAWSOrganisation :

1. L'éditeur de l'application peut supprimer des autorisations à l'aide de la commande `put-application-policy`.
2. Un utilisateur de l'management accountd'unAWSL'organisation peut effectuer unannulation du partage de l'applicationsur toute application partagée avec l'organisation, même si l'application a été publiée par un utilisateur à partir d'un autre compte.

Note

Lorsqu'une application n'est pas partagée à partir d'unAWSOrganisation avec l'opération « annulation du partage de l'application », elle ne peut pas être partagée avecAWSOrganisation à nouveau.

Pour plus d'informations sur AWS Organizations, consultez le Manuel de l'utilisateur AWS Organizations.

Suppression des autorisations par l’éditeur

Suppression des autorisations par l’éditeur via la console

Pour annuler le partage d'une application via l'AWS Management Console, vous supprimez l'instruction de stratégie qui la partage avec d'autresAWS. Pour cela, procédez comme suit :

1. Ouvrez la console AWS Serverless Application Repository .
2. Choisissez Available Applications (Applications disponibles) dans le volet de navigation gauche.
3. Choisissez l'application dont vous souhaitez annuler le partage.
4. Cliquez sur l'onglet Sharing (Partager) .
5. Dans la section Application policy statements (Déclarations de stratégie d'application), sélectionnez la déclaration de stratégie qui partage l'application avec les comptes pour lesquels vous souhaitez annuler le partage.
6. Sélectionnez Delete (Supprimer).
7. Un message de confirmation s'affiche. Choisissez Supprimer à nouveau.

Suppression des autorisations par l’éditeur via l’AWS CLI

Pour annuler le partage d'une application via l'AWS CLI, l'éditeur peut supprimer ou modifier les autorisations à l'aide de l'outil`put-application-policy`pour rendre l'application privée, ou partager avec un autre ensemble deAWS.

Pour plus d'informations sur la modification des autorisations à l'aide de l'AWSCLI, veuillez consulterExemples de stratégies basées sur les ressources AWS Serverless Application Repository (p. 46).

Management accountAnnuler le partage d'une application

Management accountannuler le partage d'une application à partir d'unAWSOrganisation via la console

Pour annuler le partage d'une application d'unAWSOrganisation via l'AWS Management Console, un utilisateur de lamanagement accountLes fonctions des permettent d'effectuer les opérations suivantes :

1. Ouvrez la console AWS Serverless Application Repository .
2. Choisissez Available Applications (Applications disponibles) dans le volet de navigation gauche.
3. Dans la tuile de l'application, choisir Unshare (Annuler le partage).
4. Dans la zone de message d'annulation du partage, confirmez que vous souhaitez annuler le partage de l'application en saisissant l'ID de l'organisation et le nom de l'application, puis en choisissant Save (Enregistrer).

Management accountannuler le partage d'une application à partir d'unAWSOrganisation via l'AWS CLI

Pour annuler le partage d'une application d'unAWSOrganisation, un utilisateur de lamanagement accountpeut exécuter le`aws serverlessrepo unshare-application`.

La commande suivante supprime le partage d'une application d'unAWSOrganisation, où*ID d'application*nom de ressource Amazon (ARN) de l'application et*identifiant de l'organisation*est leAWSID de l'organisation :

```
aws serverlessrepo unshare-application --application-id application-id --organization-
id organization-id
```

Suppression d'une application

Vous pouvez supprimer des applications depuis AWS Serverless Application Repository à l'aide de l'AWS Management Console ou de l'interface de ligne de commande AWS SAM.

Suppression d'une application (console)

Pour supprimer une application publiée via l'AWS Management Console, procédez comme suit.

1. Ouvrez la console AWS Serverless Application Repository .
2. Pour My Applications (Mes applications), choisissez l'application que vous souhaitez supprimer.
3. Sur la page des détails de l'application, choisissez Supprimer l'application.
4. Choisissez Supprimer l'application pour terminer la suppression.

Suppression d'une application (AWS CLI)

Pour supprimer une application publiée à l'aide de l'AWS CLI, exécutez la commande `aws serverlessrepo delete-application`.

La commande suivante supprime une application, où l'*application-id* est le nom de ressource Amazon (ARN) de l'application :

```
aws serverlessrepo delete-application --application-id application-id
```

Publication d'une nouvelle version d'une application existante

Cette section vous montre comment publier une nouvelle version d'une application existante sur AWS Serverless Application Repository en utilisant l'interface de ligne de commande AWS SAM ou AWS Management Console. Pour obtenir des instructions sur la publication d'une nouvelle application, veuillez consulter HowPublication des applications (p. 12).

Publication d'une nouvelle version d'une application existante (AWS CLI)

La façon la plus simple de publier une nouvelle version d'une application existante consiste à utiliser un ensemble de commandes d'interface de ligne de commande AWS SAM. Pour de plus amples informations,

veuillez consulter Publication d'une application à l'aide de l'interface de ligne de commande AWS SAM dans le Guide du développeur AWS Serverless Application Model (AWS SAM).

Publication d'une nouvelle version d'une application existante (console)

Pour publier une nouvelle version d'une application que vous avez déjà publiée, procédez comme suit :

1. Ouvrez la console AWS Serverless Application Repository .
2. Dans le panneau de navigation, choisissez Mes applications pour afficher la liste des applications que vous avez créées.
3. Choisissez l'application pour laquelle vous voulez publier une nouvelle version.
4. Choisissez Publish new version (Publier une nouvelle version).
5. Dans Versions, entrez les informations d'application suivantes :

Propriété	Obligatoire	Description
Version sémantique	TRUE	Version sémantique de l'application. Pour de plus amples informations, veuillez consulter le site web Contrôle de version sémantique. Vous devez fournir une valeur pour cette propriété afin de rendre votre application publique.
URL du code source	FALSE	Lien vers un référentiel public pour le code source de votre application.
SAM template (Modèle SAM)	TRUE	Un rapport valideAWS Serverless Application Model(AWS SAM) qui définit l'AWSqui sont utilisées.

6. Choisissez Publier la version.

Badge Auteur vérifié

Les auteurs vérifiésdans leAWS Serverless Application Repositorysont ceux pour lesquelsAWSa procédé à un examen de bonne foi, en tant que fournisseur de services raisonnable et prudent, des informations fournies par le demandeur et a confirmé que l'identité du demandeur est telle que déclarée.

Les applications des auteurs vérifiés affichent un badge d'auteur vérifié, ainsi qu'un lien vers le profil public de l'auteur. Le badge Auteur vérifié s'affiche à la fois dans les résultats de recherche et sur la page détaillée de l'application.

Demande d'un badge Auteur vérifié

Vous pouvez demander à être approuvé en tant qu'auteur vérifié dans le AWS Serverless Application Repository en envoyant un e-mail à serverlessrepo-verified-author@amazon.com. Vous devez fournir les informations suivantes :

- Nom de l'auteur
- ID de compte AWS
- Lien de profil accessible au public, tel que votre profil GitHub ou LinkedIn

Après avoir soumis une demande de badge d'auteur vérifié, vous devez attendre une réponse deAWSDans quelques jours. Vous serez parfois invité à fournir des informations supplémentaires avant que votre demande soit approuvée.

Une fois votre demande approuvée, le badge d'auteur vérifié est généralement affiché pour vos applications dans un délai d'un jour.

> Note
>
> Le badge d'auteur vérifié s'affiche pour toutes les applications qui correspondent à la fois auAWSnom du compte et de l'auteur. Etant donné queAWSLes badges ne sont pas affichés sur les applications dont le nom d'auteur est différent. Pour que des badges d'auteur apparaissent sur les demandes portant des noms d'auteur différents, vous devez soumettre une autre demande pour cet auteur.

Partage de couches Lambda

Si vous avez implémenté des fonctionnalités dans une couche Lambda, vous pouvez partager cette dernière sans en héberger une instance globale. Le partage de couches de cette manière permet à d'autres personnes de déployer une instance de votre couche sur leur propre compte. Cela empêche les applications clientes de dépendre d'une instance globale de votre couche. AWS Serverless Application Repository vous permet de partager des couches Lambda de cette manière facilement.

Pour de plus amples informations sur les couches Lambda, veuillez consulter Couches AWS Lambda dans le AWS Lambda Developer Guide.

Fonctionnement

Voici les étapes de partage de votre couche à l'aide de la AWS Serverless Application Repository. Cela permet de créer une copie de votre couche dans leAWS.

1. Définissez une application sans serveur avec un modèle AWS SAM qui inclut votre couche en tant que ressource —, c'est-à-dire une ressource `AWS::Serverless::LayerVersion` ou `AWS::Lambda::LayerVersion`.
2. Publiez votre application sur le AWS Serverless Application Repository et partagez-la (publiquement ou en privé).
3. Un client déploie votre application, ce qui crée une copie de votre couche dans son propreAWS. Le client peut désormais référencer le nom de ressource Amazon (ARN) de la couche dans sonAWSDans leur application cliente.

Example

Voici un exemple de modèle AWS SAM pour une application qui contient la couche Lambda que vous souhaitez partager :

```
Resources:
  SharedLayer:
    Type: AWS::Serverless::LayerVersion
    Properties:
```

```
      LayerName: shared-layer
      ContentUri: source/layer-code/
      CompatibleRuntimes:
        - python3.7
Outputs:
  LayerArn:
    Value: !Ref SharedLayer
```

Lorsqu'un client déploie votre application à partir duAWS Serverless Application Repository, une couche est créée dans leurAWS. L'ARN de la couche ressemble à ce qui suit :

```
arn:aws:lambda:us-east-1:012345678901:layer:shared-layer:1
```

Le client peut désormais référencer cet ARN dans sa propre application cliente, comme dans cet exemple :

```
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.handler
      Runtime: python3.7
      CodeUrl: source/app-code/
      Layers:
        - arn:aws:lambda:us-east-1:012345678901:layer:shared-layer:1
```

Déploiement d'applications

Cette section vous apprend à rechercher et à déployer des applications sans serveur publiées sur AWS Serverless Application Repository. Vous pouvez rechercher des applications publiquement disponibles, sans avoir deAWSen accédant ausite public. Vous pouvez aussi rechercher des applications à partir de la console AWS Lambda.

Certaines applications ont un badge Auteur vérifié avec un lien vers le profil de l'auteur. Un auteur est considéré comme unauteur vérifiéWHENAWSa procédé à un examen de bonne foi, en tant que fournisseur de services raisonnable et prudent, des renseignements fournis par le demandeur et a confirmé que l'identité du demandeur est telle que déclarée.

Avant de déployer des applications à partir de AWS Serverless Application Repository, veuillez consulter les rubriques suivantes pour en savoir plus sur les autorisations de déploiement d'applications et les capacités d'application.

Rubriques

Autorisations de déploiement d'applications

Pour déployer une application dans le AWS Serverless Application Repository, vous devez avoir l'autorisation de le faire. Il existe trois catégories d'applications pour lesquelles vous disposez d'autorisations de déploiement :

- Privé – Applications qui ont été créées avec le même compte et qui n'ont pas été partagées avec un autre compte. Vous avez l'autorisation de déployer des applications créées à l'aide deAWS.
- Partagé en privé –Partagé de façon explicite Applications que l'éditeur a explicitement partagées avec un ensemble spécifique deAWS. Vous avez l'autorisation de déployer des applications qui ont été partagées avec votreAWS.
- Partagé publiquement – Applications que l'éditeur a partagées avec tout le monde. Vous avez l'autorisation de déployer n'importe quelle application partagée publiquement.

Vous pouvez uniquement rechercher et rechercher les applications pour lesquelles vous disposez d'autorisations. Il s'agit de applications créées à l'aide deAWS, partagé en privé avec votreAWSet partagé publiquement. Toutes les autres applications ne sont pas affichées pour vous.

> Important
>
> Les applications qui contiennent des applications imbriquées héritent des restrictions de partage des applications imbriquées. Par exemple, supposons qu'une application soit partagée publiquement, mais qu'elle contienne une application imbriquée qui n'est partagée qu'en privé avecAWSqui a créé l'application parent. Dans ce cas, si votreAWSn'est pas autorisé à déployer l'application imbriquée, vous ne pouvez pas déployer l'application parente. Pour plus d'informations sur les applications imbriquées, consultez la rubrique relative aux applications imbriquées dans le Guide du développeur AWS Serverless Application Model.

Fonctionnalités des applications : Rôles IAM, stratégies de ressources et applications imbriquées

Avant de pouvoir déployer une application, leAWS Serverless Application Repositoryvérifie le modèle de l'application pourIAMRôles,AWSet les applications imbriquées que le modèle spécifie qu'il doit créer.IAM, telles qu'unIAMavec un accès complet, peut modifier n'importe quelle ressource dans votreAWS. Par conséquent, nous vous recommandons de passer en revue les autorisations associées à l'application avant de poursuivre. Vous éviterez ainsi de créer par erreur des ressources disposant d'autorisations que vous ne souhaitez pas accorder. Pour ce faire, vous devez confirmer que l'application contient des capacités pour qu'AWS Serverless Application Repository puisse déployer l'application en votre nom.

Les applications peuvent contenir l'une des quatre capacités suivantes : `CAPABILITY_IAM`, `CAPABILITY_NAMED_IAM`, `CAPABILITY_RESOURCE_POLICY` et `CAPABILITY_AUTO_EXPAND`.

Les ressources suivantes requièrent que vous spécifiez`CAPABILITY_IAM`ou`CAPABILITY_NAMED_IAM` : AWS::IAM::Group,AWS::IAM::InstanceProfile,AWS::IAM::Policy, etAWS::IAM::Role. Si l'application contient des ressources IAM dotées de noms personnalisés, vous devez spécifier `CAPABILITY_NAMED_IAM`. Pour obtenir un exemple de spécification des capacités, veuillez consulter Recherche et confirmation des capacités d'une application (AWS CLI) (p. 26).

Les ressources suivantes requièrent que vous spécifiez`CAPABILITY_RESOURCE_POLICY` : AWS::Lambda::LayerVersionPermission,AWS::Lambda::Permission,AWS::Events::EventBusPolicy,AWS# IAM:Politique,AWS::ApplicationAutoScaling::ScalingPolicy,AWS::S3::BucketPolicy,AWS::SQS::QueuePolicy, etAWS::SNS::TopicPolicy.

Les applications contenant une ou plusieurs applications imbriquées exigent de spécifier `CAPABILITY_AUTO_EXPAND`. Pour plus d'informations sur les applications imbriquées, consultez la rubrique relative aux applications imbriquées dans le Guide du développeur AWS Serverless Application Model.

Recherche et confirmation des capacités d'une application (console)

La liste des applications disponibles dans AWS Serverless Application Repository est accessible sur le site Internet d'AWS Serverless Application Repository ou via la console Lambda sur la page Créer une fonction sous l'onglet AWS Serverless Application Repository.

Les applications qui exigent la confirmation des capacités pour créer des rôles IAM ou des stratégies de ressources personnalisés ne s'affichent pas dans les résultats de la recherche par défaut. Pour rechercher les applications contenant ces capacités, vous devez cocher la case Afficher les applications qui créent des rôles IAM ou des stratégies de ressources personnalisés.

Vous pouvez passer en revue les capacités d'une application dans l'onglet Permissions (Autorisations) lorsque vous sélectionnez l'application. Pour déployer l'application, vous devez cocher la case Je confirme que cette application crée des rôles IAM ou des stratégies de ressources personnalisés. Si vous ne confirmez pas ces capacités, le message d'erreur suivant s'affiche : Confirmation obligatoire. Pour procéder au déploiement, cochez la case dans la section Configurer les paramètres de l'application.

Affichage des fonctionnalités des applications (AWS CLI)

Pour afficher les capacités d'une application à l'aide de AWS CLI, vous avez d'abord besoin de l'ARN (Amazon Resource Name) de l'application. Vous pouvez ensuite exécuter la commande suivante :

```
aws serverlessrepo get-application \
--application-id application-arn
```

La propriété de réponse requiredCapabilities contient la liste des capacités d'application que vous devez confirmer pour pouvoir déployer l'application. Notez que si la propriété requiredCapabilities est vide, l'application n'a pas de capacités requises.

Comment déployerApplications

Cette section fournit des procédures de déploiement d'applications sans serveur à partir de la AWS Serverless Application Repository en utilisant le AWS Management Console ou l'AWS CLI.

Déploiement d'une nouvelle application (console)

Cette section vous montre comment déployer une nouvelle application à partir de AWS Serverless Application Repository en utilisant AWS Management Console. Pour obtenir des instructions sur le déploiement d'une nouvelle version d'une application existante, veuillez consulter Mise à jour des applications (p. 27).

Exploration, recherche et déploiement d'applications

Recherchez, configurez et déployez une application dans AWS Serverless Application Repository à l'aide de la procédure suivante.

Pour rechercher et configurer une application dans AWS Serverless Application Repository

1. Ouvrez la page d'accueil publique AWS Serverless Application Repository, ou ouvrez la console AWS Lambda. Choisissez Créer une fonction, puis choisissez Parcourir le référentiel d'applications sans serveur.
2. Recherchez une application.

   Note

   Pour afficher les applications contenant des rôles IAM ou des stratégies de ressources personnalisés, cochez la case Show apps that create custom IAM roles or resource policies (Afficher les applications qui créent des rôles IAM ou des stratégies de ressources personnalisés). Pour plus d'informations sur les rôles IAM et les stratégies de ressources personnalisés, consultez Confirmation des capacités d'une application (p. 24).
3. Choisissez une application pour afficher des informations, comme ses autorisations, ses capacités et le nombre de fois où elle a été déployée parAWSclients.

   Le nombre de déploiements est affiché pour leAWSRégion dans laquelle vous essayez de déployer l'application.
4. Sur la page des détails de l'application, consultez les autorisations et les ressources de l'application en affichant le modèle AWS SAM, la licence et le fichier Lisez-moi. Sur cette page, vous pouvez également rechercher le lien Source code URL (URL du code source) des applications publiquement partagées. Si l'application comprend des applications imbriquées, vous pouvez également afficher les détails de ces applications sur cette page.
5. Configurez l'application dans la section Application settings (Paramètres de l'application). Pour obtenir des instructions sur la configuration d'une application particulière, consultez le fichier Lisez-moi de l'application.

Par exemple, la configuration requise peut inclure la spécification du nom d'une ressource à laquelle vous voulez que l'application accède. Cette ressource peut être une table Amazon DynamoDB, un compartiment Amazon S3 ou une API Amazon API Gateway.

6. Choisissez Deploy (Déployer). Vous accédez ainsi à la page Statut du déploiement.

   Note

   Si l'application dispose de fonctionnalités nécessitant un accusé de réception, vous devez cocher la case J'accuse réception que cette application crée des rôles IAM personnalisés ou des stratégies de ressources avant de déployer l'application. Si vous ne le faites pas, une erreur se produit. Pour plus d'informations sur les rôles IAM et les stratégies de ressources personnalisés, consultez Confirmation des capacités d'une application (p. 24).

7. Sur la page Deployment status (Statut du déploiement), vous pouvez consulter la progression de votre déploiement. En attendant que votre déploiement se termine, vous pouvez rechercher d'autres applications et revenir sur cette page via la console Lambda.

Une fois que votre application a été déployée avec succès, vous pouvez vérifier et gérer les ressources créées à l'aide desAWSOutils.

Déploiement d'une nouvelle application (AWS CLI)

Cette section vous montre comment déployer une nouvelle application à partir de l'AWS Serverless Application Repository en utilisant la AWS CLI. Pour obtenir des instructions sur le déploiement d'une nouvelle version d'une application existante, veuillez consulter Mise à jour des applications (p. 27).

Recherche et confirmation des capacités d'une application (AWS CLI)

Pour confirmer les capacités d'une application à l'aide de l'AWS CLI, procédez comme suit :

1. Examinez les capacités de l'application. Utilisez leAWS CLIpour examiner les capacités d'une application :

   ```
   aws serverlessrepo get-application \
   --application-id application-arn
   ```

   La propriété de réponse requiredCapabilities contient la liste des capacités d'application que vous devez confirmer pour pouvoir déployer l'application. Vous pouvez également utiliser l'API GetApplicationdans leAWSSDK pour obtenir ces données.

2. Créez le jeu de modifications. Vous devez fournir l'ensemble desCapacitéslorsque vous créez leAWS CloudFormationChangeset Par exemple, utilisez la commande de l'AWS CLI suivante pour déployer une application en confirmant ses capacités :

   ```
   aws serverlessrepo create-cloud-formation-change-set \
   --application-id application-arn \
   --stack-name unique-name-for-cloud-formation-stack \
   --capabilities list-of-capabilities
   ```

   L'ID de changeset est renvoyé lorsque cette commande est exécutée avec succès. Vous avez besoin de l'ID de changeset pour l'étape suivante. Vous pouvez également utiliser l'API CreateCloudFormationChangeSetdans leAWSSDK pour créer le jeu de modifications.

   Par exemple, la commande AWS CLI suivante permet de reconnaître une application contenant une ressource AWS::IAM::Role avec un nom personnalisé et une ou plusieurs applications imbriquées :

```
aws serverlessrepo create-cloud-formation-change-set \
--application-id application-arn \
--stack-name unique-name-for-cloud-formation-stack \
--capabilities CAPABILITY_NAMED_IAM CAPABILITY_AUTO_EXPAND
```

3. Exécutez le changeset L'exécution du changeset effectue réellement le déploiement. Indiquez l'ID de changeset qui a été renvoyé lorsque vous avez créé le changeset à l'étape précédente.

   L'exemple de commande AWS CLI suivant exécute le changeset de l'application pour déployer cette dernière :

   ```
   aws cloudformation execute-change-set \
   --change-set-name changeset-id-arn
   ```

   Vous pouvez également utiliser l'API ExecuteChangeSetdans leAWSSDK pour exécuter le changeset

Suppression des piles d'applications

Pour supprimer une application précédemment déployée à l'aide de l'AWS Serverless Application Repository, suivez la même procédure que pour la suppression d'une pile AWS CloudFormation :

- AWS Management Console : pour supprimer une application avec AWS Management Console, consultez Suppression d'une pile sur la console AWS CloudFormation dans le AWS CloudFormation Guide de l'utilisateur.
- AWS CLI : pour supprimer une application avec AWS CLI, veuillez consulter Suppression d'une pile dans le AWS CloudFormation Guide de l'utilisateur.

Mise à jour des applications

Après avoir déployé une application à partir de AWS Serverless Application Repository, vous pouvez la mettre à jour. Par exemple, vous pouvez modifier un paramètre d'application ou mettre à jour l'application vers la dernière version publiée.

Les sections suivantes décrivent comment déployer une nouvelle version d'une application à l'aide de la AWS Management Console ou de l'AWS CLI.

Mise à jour des applications (console)

Pour mettre à jour une application que vous avez précédemment déployée, utilisez la même procédure que pour le déploiement d'une nouvelle application et fournissez le même nom d'application que celui avec lequel vous l'avez initialement déployée. Plus particulièrement, AWS Serverless Application Repository ajoute le préfixe `serverlessrepro-` au nom de votre application. Toutefois, pour déployer une nouvelle version de votre application, vous fournissez le nom d'application d'origine sans `serverlessrepo-` en préfixe.

Par exemple, si vous avez déployé une application portant le nom `MyApplication`, le nom de la pile est `serverlessrepo-MyApplication`. Pour mettre à jour cette application, vous devez fournir le nom `MyApplication` à nouveau—Ne spécifiez pas le nom complet de la pile de `serverlessrepo-MyApplication`.

Pour tous les autres paramètres d'application, vous pouvez conserver les mêmes valeurs que le déploiement précédent ou fournir de nouvelles valeurs.

Mise à jour des applications (AWS CLI)

Pour mettre à jour une application que vous avez précédemment déployée, utilisez la même procédure que le déploiement d'une nouvelle application et fournissez la même `--stack-name` que celle avec laquelle vous l'avez initialement déployée. Plus particulièrement, AWS Serverless Application Repository ajoute le préfixe `serverlessrepo-` à votre nom de pile. Toutefois, pour déployer une nouvelle version de votre application, vous fournissez le nom de la pile d'origine sans ajouter le préfixe `serverlessrepo-`.

Par exemple, si vous avez déployé une application avec le nom de la pile `MyApplication`, le nom de la pile créée est `serverlessrepo-MyApplication`. Pour mettre à jour cette application, vous devez fournir le nom `MyApplication` à nouveau—Ne spécifiez pas le nom complet de la pile de `serverlessrepo-MyApplication`.

Sécurité dans le AWS Serverless Application Repository

Chez AWS, la sécurité dans le cloud est notre priorité numéro 1. En tant que client AWS, vous bénéficiez d'un centre de données et d'une architecture réseau conçus pour répondre aux exigences des organisations les plus pointilleuses en termes de sécurité.

La sécurité est une responsabilité partagée entre AWS et vous-même. Le modèle de responsabilité partagée décrit ceci comme la sécurité du cloud et la sécurité dans le cloud :

- Sécurité du cloud – AWS est responsable de la protection de l'infrastructure qui exécute des services AWS dans le cloud AWS. AWS vous fournit également les services que vous pouvez utiliser en toute sécurité. Des auditeurs tiers testent et vérifient régulièrement l'efficacité de notre sécurité dans le cadre des programmes de conformité AWS. Pour de plus amples informations sur les programmes de conformité qui s'appliquent à AWS Serverless Application Repository, veuillez consulter Services AWS concernés par le programme de conformité.
- Sécurité dans le cloud – Votre responsabilité est déterminée par le service AWS que vous utilisez. Vous êtes également responsable d'autres facteurs, y compris la sensibilité de vos données, les exigences de votre entreprise,et la législation et la réglementation applicables.

Cette documentation vous aide à comprendre comment appliquer le modèle de responsabilité partagée lorsque vous utilisez AWS Serverless Application Repository. Les rubriques suivantes vous montrent comment configurer l'AWS Serverless Application Repository pour qu'elle réponde à vos objectifs de sécurité et de conformité. Vous apprendrez également à utiliser d'autres services AWS pour surveiller et sécuriser vos ressources AWS Serverless Application Repository.

Rubriques

Protection des données dans l'AWS Serverless Application Repository

Le modèle de responsabilité partagée AWS s'applique à la protection des données dans AWS Serverless Application Repository. Comme décrit dans ce modèle, AWS est responsable de la protection de l'infrastructure globale sur laquelle s'exécute l'ensemble du AWS Cloud . La gestion du contrôle de votre contenu hébergé sur cette infrastructure est de votre responsabilité. Ce contenu comprend les tâches de configuration et de gestion de la sécurité des services AWS que vous utilisez. Pour plus d'informations sur la confidentialité des données, veuillez consulter FAQ sur la confidentialité des données. Pour plus d'informations sur la protection des données en Europe, veuillez consulter le billet de blog AWSShared Responsibility Model and GDPR sur la page AWSSecurity Blog.

À des fins de protection des données, nous vous recommandons de protéger les informations d'identification du Compte AWS et de configurer les comptes d'utilisateur individuels avec AWS Identity and Access Management (IAM). Ainsi, chaque utilisateur se voit attribuer uniquement les autorisations nécessaires pour exécuter ses tâches. Nous vous recommandons également de sécuriser vos données comme indiqué ci-dessous :

- Utilisez l'authentification multi-facteurs (MFA) avec chaque compte.
- Utilisez SSL/TLS pour communiquer avec des ressources AWS. Nous recommandons TLS 1.2 ou version ultérieure.
- Configurez l'API et la consignation des activités utilisateur avec AWS CloudTrail.
- Utilisez des solutions de chiffrement AWS, ainsi que tous les contrôles de sécurité par défaut au sein des services AWS.
- Utilisez des services de sécurité gérés comme Amazon Macie, qui contribue à la découverte et à la sécurisation des données personnelles stockées dans Amazon S3.
- Si vous avez besoin de modules cryptographiques validés FIPS 140-2 lorsque vous accédez à AWS via une interface de ligne de commande ou une API, utilisez un point de terminaison FIPS. Pour de plus amples informations sur les points de terminaison FIPS disponibles, consultez Federal Information Processing Standard (FIPS) 140-2.

Nous vous recommandons fortement de ne jamais placer d'informations confidentielles ou sensibles, telles que les adresses e-mail de vos clients, dans des balises ou des champs de forme libre tels qu'un champ Nom. Cela inclut lorsque vous utilisez AWS Serverless Application Repository ou d'autres services AWS à l'aide de la console, de l'API, de l'interface de ligne de commande (AWS CLI) ou des kits SDK AWS. Toutes les données que vous saisissez dans des balises ou des champs de forme libre utilisés pour les noms peuvent être utilisées pour les journaux de facturation ou de diagnostic. Si vous fournissez une URL à un serveur externe, nous vous recommandons fortement de ne pas inclure d'informations d'identification dans l'URL pour valider votre demande auprès de ce serveur.

Chiffrement en transit

AWS Serverless Application Repository Les points de terminaison d'API ne prennent en charge que des connexions sécurisées sur HTTPS. Lorsque vous gérezAWS Serverless Application RepositoryRessources avecAWS Management Console,AWSSDK, ou leAWS Serverless Application RepositoryToutes les communications sont chiffrées avec Transport Layer Security (TLS).

Pour obtenir la liste complète des points de terminaison d'API, consultezAWSRégions et points de terminaisondans leAWS General Reference.

Chiffrement au repos

AWS Serverless Application Repository chiffre toujours les fichiers que vous chargez sur AWS Serverless Application Repository, y compris les packages de déploiement et les archives de couches.

Identity and Access Management (IAM) pour l'AWS Serverless Application Repository

AWS Identity and Access Management (IAM) est un service AWS qui permet à un administrateur de contrôler en toute sécurité l'accès aux ressources AWS. Les administrateurs IAM contrôlent qui peut être authentifié (connecté) et autorisé (disposant des autorisations) à utiliser les ressources AWS Serverless Application Repository. IAM est un service AWS que vous pouvez utiliser sans frais supplémentaires.

Pour obtenir une vue d'ensemble du fonctionnement de IAM, veuillez consulter Présentation du fonctionnement de IAM dans le IAM Guide de l'utilisateur.

Rubriques

Audience

Votre utilisation d'AWS Identity and Access Management (IAM) évolue selon la tâche que vous réalisez dans AWS Serverless Application Repository.

Utilisateur du service – Si vous utilisez le service AWS Serverless Application Repository pour effectuer votre tâche, votre administrateur vous fournit les informations d'identification et les autorisations dont vous avez besoin. Plus vous utiliserez de fonctionnalités AWS Serverless Application Repository pour effectuer votre travail, plus vous pourrez avoir besoin d'autorisations supplémentaires. Comprendre la gestion des accès peut vous aider à demander à votre administrateur les autorisations appropriées. Si vous ne pouvez pas accéder à une fonctionnalité dans AWS Serverless Application Repository, consultez Résolution des problèmes d'identité et d'accès AWS Serverless Application Repository (p. 51).

Administrateur du service –Si vous êtes le responsable des ressources AWS Serverless Application Repository de votre entreprise, vous bénéficiez probablement d'un accès total à AWS Serverless Application Repository. C'est à vous de déterminer les fonctionnalités et les ressources AWS Serverless Application Repository auxquelles vos employés pourront accéder. Vous devez ensuite soumettre les demandes à votre administrateur IAM pour modifier les autorisations des utilisateurs de votre service. Consultez les informations sur cette page pour comprendre les concepts de base d'IAM. Pour en savoir plus sur la façon dont votre entreprise peut utiliser IAM avec AWS Serverless Application Repository, consultez Comment AWS Serverless Application Repository fonctionne avec IAM (p. 35).

Administrateur IAM – Si vous êtes un administrateur IAM, vous souhaiterez peut-être obtenir des détails sur la façon dont vous pouvez écrire des stratégies pour gérer l'accès à AWS Serverless Application Repository. Pour obtenir des exemples de stratégies AWS Serverless Application Repository basées sur l'identité que vous pouvez utiliser dans IAM, consultez Exemples de stratégies AWS Serverless Application Repository basées sur l'identité (p. 40).

Authentification avec des identités

L'authentification correspond au processus par lequel vous vous connectez à AWS via vos informations d'identification. Pour de plus amples informations sur la connexion avec AWS Management Console, veuillez consulter Connexion à AWS Management Console en tant qu'utilisateur IAM ou utilisateur racine dans le IAM Guide de l'utilisateur.

Vous devez être authentifié (connecté à AWS) en tant que Utilisateur racine Compte AWS ou utilisateurIAM, ou en assumant un rôle IAM. Vous pouvez également utiliser l'authentification de connexion unique de votre entreprise ou vous connecter par le biais de Google ou de Facebook. Dans ce cas, votre administrateur aura précédemment configuré une fédération d'identités avec des rôles IAM. Lorsque vous

accédez à AWS avec des informations d'identification d'une autre entreprise, vous assumez indirectement un rôle.

Pour vous connecter directement à AWS Management Console, utilisez votre mot de passe avec votre e-mail utilisateur racine ou votre nom d'utilisateur IAM. Vous pouvez accéder à AWS par programmation avec vos clés d'accès utilisateur utilisateur racine ou IAM. AWS fournit un kit SDK et des outils de ligne de commande pour signer de manière chiffrée votre demande avec vos informations d'identification. Si vous n'utilisez pas les outils AWS, vous devez signer la requête vous-même. Pour ce faire, utilisez Signature Version 4, un protocole permettant d'authentifier les demandes d'API entrantes. Pour en savoir plus sur l'authentification des demandes, consultez Processus de signature Signature Version 4 dans AWS General Reference.

Quelle que soit la méthode d'authentification que vous utilisez, vous devrez peut-être également fournir des informations de sécurité supplémentaires. Par exemple, AWS vous recommande d'utiliser l'authentification multi-facteurs (MFA) pour améliorer la sécurité de votre compte. Pour en savoir plus, consultez Utilisation de Multi-Factor Authentication (MFA) dans AWS dans le IAM Guide de l'utilisateur.

Utilisateur racine Compte AWS

Lorsque vous créez pour la première fois un Compte AWS , vous commencez avec une identité de connexion unique qui bénéficie d'un accès complet à tous les services et ressources AWS du compte. Cette identité est appelée la utilisateur racinedu Compte AWS et elle est accessible après connexion à l'aide de l'adresse e-mail et du mot de passe utilisés pour la création du compte. Il est vivement recommandé de ne pas utiliser l'utilisateur racine pour vos tâches quotidiennes, y compris pour les tâches administratives. Au lieu de cela, respectez la bonne pratique qui consiste à avoir recours à l'utilisateur racine uniquement pour créer le premier utilisateurIAM. Ensuite, mettez en sécurité les informations d'identification de l'utilisateur racine et utilisez-les pour effectuer uniquement certaines tâches de gestion des comptes et des services.

Utilisateurs et groupes IAM

Un utilisateur IAM est une identité dans votre Compte AWS qui dispose d'autorisations spécifiques pour une seule personne ou application. Un utilisateur IAM peut disposer d'informations d'identification à long terme comme un nom d'utilisateur et un mot de passe ou un ensemble de clés d'accès. Pour savoir comment générer des clés d'accès, consultez Gestion des clés d'accès pour les utilisateurs IAM dans le IAM Guide de l'utilisateur. Lorsque vous générez des clés d'accès pour un utilisateur IAM, veillez à afficher et enregistrer la paire de clés de manière sécurisée. Vous ne pourrez plus récupérer la clé d'accès secrète à l'avenir. Au lieu de cela, vous devrez générer une nouvelle paire de clés d'accès.

Un groupe IAM est une identité qui spécifie un ensemble d'utilisateurs IAM. Vous ne pouvez pas vous connecter en tant que groupe. Vous pouvez utiliser les groupes pour spécifier des autorisations pour plusieurs utilisateurs à la fois. Les groupes permettent de gérer plus facilement les autorisations pour de grands ensembles d'utilisateurs. Par exemple, vous pouvez avoir un groupe nommé Admins IAM et accorder à ce groupe les autorisations leur permettant d'administrer des ressources IAM.

Les utilisateurs sont différents des rôles. Un utilisateur est associé de manière unique à une personne ou une application, alors qu'un rôle est conçu pour être endossé par tout utilisateur qui en a besoin. Les utilisateurs disposent d'informations d'identification permanentes, mais les rôles fournissent des informations d'identification temporaires. Pour en savoir plus, consultez Quand créer un utilisateur IAM (au lieu d'un rôle) dans le IAM Guide de l'utilisateur.

Rôles IAM

Un rôle IAM est une entité au sein de votre Compte AWS qui dispose d'autorisations spécifiques. Le concept ressemble à celui d'utilisateur IAM, mais un rôle n'est pas associé à une personne en particulier. Vous pouvez temporairement endosser un rôle IAM dans l'AWS Management Console grâce au changement de rôle. Vous pouvez obtenir un rôle en appelant une opération d'API AWS CLI ou AWS à

l'aide d'une URL personnalisée. Pour plus d'informations sur les méthodes d'utilisation des rôles, consultez Utilisation de rôles IAM dans le IAM Guide de l'utilisateur.

Les rôles IAM avec des informations d'identification temporaires sont utiles dans les cas suivants :

- Autorisations utilisateur IAM temporaires – Un utilisateur IAM peut endosser un rôle IAM pour accepter différentes autorisations temporaires concernant une tâche spécifique.
- Accès d'utilisateurs fédérés – Au lieu de créer un utilisateur IAM, vous pouvez utiliser des identités d'utilisateur préexistantes provenant d'AWS Directory Service, de l'annuaire d'utilisateurs de votre entreprise ou d'un fournisseur d'identité web. On parle alors d'utilisateurs fédérés. AWS attribue un rôle à un utilisateur fédéré lorsque l'accès est demandé via un fournisseur d'identité. Pour de plus amples informations sur les utilisateurs fédérés, veuillez consulter Utilisateurs et rôles fédérés dans le IAM Guide de l'utilisateur.
- Accès entre comptes – Vous pouvez utiliser un rôle IAM pour permettre à un utilisateur (mandataire de confiance) d'un compte différent d'accéder aux ressources de votre compte. Les rôles constituent le principal moyen d'accorder l'accès entre plusieurs comptes. Toutefois, certains services AWS vous permettent d'attacher une stratégie directement à une ressource (au lieu d'utiliser un rôle en tant que proxy). Pour en savoir plus sur la différence entre les rôles et les stratégies basées sur les ressources pour l'accès entre comptes, consultez Différence entre les rôles IAM et les stratégies basées sur les ressources dans le IAM Guide de l'utilisateur.
- Accès inter-service – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
  - Autorisations principales – When you use an IAM user or role to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions. To see whether an action requires additional dependent actions in a policy, see Actions, Resources, and Condition Keys for AWS Serverless Application Repository in the Service Authorization Reference.
  - Rôle de service –Un rôle de service est un rôle IAM qu'un service assume pour exécuter des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer un rôle de service à partir d'IAM. Pour de plus amples informations, veuillez consulter Création d'un rôle pour la délégation d'autorisations à un service AWS dans le IAM Guide de l'utilisateur.
  - Rôle lié à un service – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your IAM account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.
- Applications qui s'exécutent sur Amazon EC2 –Vous pouvez utiliser un rôle IAM pour gérer des informations d'identification temporaires pour les applications qui s'exécutent sur une instance EC2 et effectuent des demandes d'API AWS CLI ou AWS. Cette solution est préférable au stockage des clés d'accès au sein de l'instance EC2. Pour attribuer un rôle AWS à une instance EC2 et le rendre disponible à toutes les applications associées, vous pouvez créer un profil d'instance attaché à l'instance. Un profil d'instance contient le rôle et permet aux programmes qui s'exécutent sur l'instance EC2 d'obtenir des informations d'identification temporaires. Pour de plus amples informations, veuillez consulter Utilisation d'un rôle IAM pour accorder des autorisations à des applications s'exécutant sur des instances Amazon EC2 dans le IAM Guide de l'utilisateur.

Pour savoir si vous devez utiliser les rôles IAM ou les utilisateurs IAM, veuillez consulter Quand créer un rôle IAM (au lieu d'un utilisateur) dans le IAM Guide de l'utilisateur.

Gestion de l'accès à l'aide de stratégies

Vous contrôler les accès dans AWS en créant des stratégies et en les attachant à des identités IAM ou à des ressources AWS. Une stratégie est un objet dans AWS qui, lorsqu'il est associé à une identité ou

à une ressource, définit ses autorisations. Vous pouvez vous connecter en tant que utilisateur racine, en tant qu'utilisateur IAM ou vous pouvez assumer un rôle IAM. Lorsque vous effectuez ensuite une demande, AWS évalue les stratégies relatives basées sur l'identité ou les ressources. Les autorisations dans les stratégies déterminent si la demande est autorisée ou refusée. La plupart des stratégies sont stockées dans AWS en tant que documents JSON. Pour plus d'informations sur la structure et le contenu du document de stratégie JSON, consultez Présentation des stratégies JSON dans le IAM Guide de l'utilisateur.

Administrators can use AWS JSON policies to specify who has access to what. That is, which principal can perform actions on what resources, and under what conditions.

Chaque entité IAM (utilisateur ou rôle) démarre sans autorisation. En d'autres termes, par défaut, les utilisateurs ne peuvent rien faire, pas même changer leurs propres mots de passe. Pour autoriser un utilisateur à effectuer une opération, un administrateur doit associer une stratégie d'autorisations à ce dernier. Il peut également ajouter l'utilisateur à un groupe disposant des autorisations prévues. Lorsqu'un administrateur accorde des autorisations à un groupe, tous les utilisateurs de ce groupe se voient octroyer ces autorisations.

Les stratégies IAM définissent les autorisations d'une action quelle que soit la méthode que vous utilisez pour exécuter l'opération. Par exemple, supposons que vous disposiez d'une stratégie qui autorise l'action `iam:GetRole`. Un utilisateur avec cette stratégie peut obtenir des informations utilisateur à partir de l'AWS Management Console, de l'AWS CLI ou de l'API AWS.

Stratégies basées sur l'identité

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the IAM Guide de l'utilisateur.

Les stratégies basées sur l'identité peuvent être classées comme étant des stratégies en ligne ou des stratégies gérées. Les stratégies en ligne sont intégrées directement à un utilisateur, groupe ou rôle. Les stratégies gérées sont des stratégies autonomes que vous pouvez lier à plusieurs utilisateurs, groupes et rôles de votre Compte AWS . Les stratégies gérées incluent les stratégies gérées par AWS et les stratégies gérées par le client. Pour découvrir comment choisir entre une politique gérée ou une politique en ligne, consultez Choix entre les stratégies gérées et les stratégies en ligne dans le IAM Guide de l'utilisateur.

Stratégies basées sur une ressource

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM role trust policies and Amazon S3 bucket policies. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

Les stratégies basées sur les ressources sont des stratégies en ligne situées dans ce service. Vous ne pouvez pas utiliser les stratégies gérées AWS depuis IAM dans une stratégie basée sur les ressources.

Listes de contrôle d'accès (ACL)

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF et Amazon VPC sont des exemples de services qui prennent en charge les listes de contrôle d'accès (ACL). Pour en savoir plus sur les listes de contrôle d'accès, veuillez consulter

Présentation des listes de contrôle d'accès (ACL) dans le Guide du développeur Amazon Simple Storage Service.

Autres types de stratégie

AWS prend en charge d'autres types de stratégies moins courantes. Ces types de stratégies peuvent définir le nombre maximal d'autorisations qui vous sont accordées par des types de stratégies plus courants.

- Limite d'autorisations – Une limite d'autorisations est une fonctionnalité avancée dans laquelle vous définissez les autorisations maximales qu'une stratégie basée sur l'identité peut accorder à une entité IAM (utilisateur ou rôle IAM). Vous pouvez définir une limite d'autorisations pour une entité. Les autorisations obtenues représentent la combinaison des stratégies basées sur l'identité de l'entité et de ses limites d'autorisations. Les stratégies basées sur les ressources qui spécifient l'utilisateur ou le rôle dans le champ `Principal` ne sont pas limitées par les limites d'autorisations. Un refus explicite dans l'une de ces stratégies remplace l'autorisation. Pour plus d'informations sur les limites d'autorisations, consultez Limites d'autorisations pour des entités IAM dans le IAM Guide de l'utilisateur.
- Politiques de contrôle des services (SCP) – Les SCP sont des stratégies JSON qui spécifient le nombre maximal d'autorisations pour une organisation ou une unité d'organisation (OU) dans AWS Organizations. AWS Organizations est un service qui vous permet de regrouper et de gérer de façon centralisée plusieurs Comptes AWS détenus par votre entreprise. Si vous activez toutes les fonctions d'une organisation, vous pouvez appliquer les stratégies de contrôle de service (SCP) à l'un ou à l'ensemble de vos comptes. La politique de contrôle des services limite les autorisations pour les entités dans les comptes membres, y compris dans chaque Utilisateur racine Compte AWS . Pour plus d'informations sur les Organizations et les SCP, consultez Fonctionnement des stratégies de contrôle de service dans le Manuel de l'utilisateur AWS Organizations.
- Stratégies de session – Les stratégies de session sont des stratégies avancées que vous transmettez en tant que paramètre lorsque vous créez par programmation une session temporaire pour un rôle ou un utilisateur fédéré. Les autorisations de la session obtenue sont une combinaison des stratégies basées sur l'identité de l'utilisateur ou du rôle et des stratégies de session. Les autorisations peuvent également provenir d'une stratégie basée sur les ressources. Un refus explicite dans l'une de ces stratégies remplace l'autorisation. Pour plus d'informations, consultez Stratégies de session dans le IAM Guide de l'utilisateur.

Plusieurs types de stratégie

Lorsque plusieurs types de stratégies s'appliquent à la requête, les autorisations obtenues sont plus compliquées à comprendre. Pour découvrir la façon dont AWS détermine s'il convient d'autoriser une requête en présence de plusieurs types de stratégies, consultez Logique d'évaluation de stratégies dans le IAM Guide de l'utilisateur.

Comment AWS Serverless Application Repository fonctionne avec IAM

Avant d'utiliser IAM pour gérer l'accès à la gestion de configuration d'AWS Serverless Application Repository, vous devez connaître les fonctions IAM pouvant être utilisées avec la gestion de configuration d'AWS Serverless Application Repository.

Pour obtenir une vue d'ensemble du fonctionnement de IAM, veuillez consulter Présentation du fonctionnement de IAM dans le IAM Guide de l'utilisateur. Pour obtenir une vue d'ensemble de la façon dont AWS Serverless Application Repository et d'autres services AWS fonctionnent avec IAM, veuillez consulter Services AWS qui fonctionnent avec IAM dans le IAM Guide de l'utilisateur.

Rubriques

Stratégies AWS Serverless Application Repository basées sur l'identité

Avec les stratégies IAM basées sur l'identité, vous pouvez spécifier des actions et ressources autorisées ou refusées, ainsi que les conditions dans lesquelles les actions sont autorisées ou refusées. La gestion de configuration d'AWS Serverless Application Repository prend en charge des actions, ressources et clés de condition spécifiques. Pour en savoir plus sur tous les éléments que vous utilisez dans une stratégie JSON, consultez Références des éléments de stratégie JSON IAM dans le IAM Guide de l'utilisateur.

Un exemple de stratégie d'autorisation est exposé ci-dessous.

```
{

    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CreateApplication",
            "Effect": "Allow",
            "Action": [
                "serverlessrepo:CreateApplication"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CreateApplicationVersion",
            "Effect": "Allow",
            "Action": [
                "serverlessrepo:CreateApplicationVersion"
            ],
            "Resource": "arn:partition:serverlessrepo:region:account-
id:applications/application-name"
        }
    ]
}
```

La stratégie possède deux énoncés:

- La première instruction accorde les autorisations pour l'action AWS Serverless Application Repository `serverlessrepo:CreateApplication` sur toutes les ressources AWS Serverless Application Repository, tel que spécifié par le caractère générique (*) comme valeur `Resource`.
- La deuxième instruction accorde l'autorisation pour l'instruction d'AWS Serverless Application RepositoryAction`serverlessrepo:CreateApplicationVersion`sur unAWSà l'aide de l'Amazon Resource Name (ARN) d'uneAWS Serverless Application Repositoryapplication. L'application est spécifiée par la valeur `Resource`.

La stratégie ne spécifie pas l'élément `Principal` car, dans une stratégie basée sur une identité, vous ne spécifiez pas le mandataire qui obtient l'autorisation. Quand vous attachez une stratégie à un utilisateur, l'utilisateur est le mandataire implicite. Lorsque vous attachez une stratégie d'autorisation à un rôle IAM, le mandataire identifié dans la stratégie d'approbation de ce rôle obtient les autorisations.

Pour obtenir un tableau montrant tous les éléments deAWS Serverless Application RepositoryOpérations d'API et l'APIAWSauxquelles elles s'appliquent, consultezAWS Serverless Application RepositoryAutorisations d'API : Référence des actions et ressources (p. 50).

Actions

Administrators can use AWS JSON policies to specify who has access to what. That is, which principal can perform actions on what resources, and under what conditions.

L'élément `Action` d'une stratégie JSON décrit les actions que vous pouvez utiliser pour autoriser ou refuser l'accès à une stratégie. Les actions de stratégie possèdent généralement le même nom que l'opération d'API AWS associée. Il existe quelques exceptions, telles que les actions avec autorisations uniquement qui n'ont pas d'opération API correspondante. Certaines opérations nécessitent également plusieurs actions dans une stratégie. Ces actions supplémentaires sont nommées actions dépendantes.

Intégration d'actions dans une stratégie afin d'accorder l'autorisation d'exécuter les opérations associées.

Actions de stratégie dans laAWS Serverless Application RepositoryUtilisez le préfixe suivant avant l'action : `serverlessrepo:`. Par exemple, pour accorder à une personne l'autorisation d'exécuter une opération d'APIAWS Serverless Application Repositoryavec l'instanceAWS Serverless Application Repository `SearchApplications`, vous incluez l'opération`serverlessrepo:SearchApplications`action dans leur politique. Les déclarations de stratégie doivent inclure un élément `Action` ou `NotAction`. AWS Serverless Application Repository définit son propre ensemble d'actions qui décrivent les tâches que vous pouvez effectuer avec ce service.

Pour spécifier plusieurs actions dans une seule déclaration, séparez-les par des virgules comme suit :

```
"Action": [
      "serverlessrepo:action1",
      "serverlessrepo:action2"
]
```

Vous pouvez aussi spécifier plusieurs actions à l'aide de caractères génériques (*). Par exemple, pour spécifier toutes les actions qui commencent par le mot `List`, incluez l'action suivante :

```
"Action": "serverlessrepo:List*"
```

Pour afficher la liste des actions AWS Serverless Application Repository, consultez Actions Defined by AWS Serverless Application Repository dans le IAM Guide de l'utilisateur.

Resources

Administrators can use AWS JSON policies to specify who has access to what. That is, which principal can perform actions on what resources, and under what conditions.

L'élément de stratégie JSON `Resource` indique le ou les objets pour lesquels l'action s'applique. Les instructions doivent inclure un élément `Resource` ou `NotResource`. Au titre de bonne pratique, il est recommandé de définir une ressource à l'aide de son Amazon Resource Name (ARN). Vous pouvez le faire pour les actions qui sont compatibles avec un type de ressource spécifique, connu sous la dénomination autorisations de niveau ressource.

Pour les actions qui ne sont pas compatibles avec les autorisations de niveau ressource, telles que les opérations de liste, utilisez un caractère générique (*) afin d'indiquer que l'instruction s'applique à toutes les ressources.

```
"Resource": "*"
```

DansAWS Serverless Application Repository, le principalAWSest unAWS Serverless Application Repository Application.AWS Serverless Application RepositoryLes applications disposent d'ARN (Amazon Resource Name) uniques qui leur sont associés, comme illustré dans le tableau suivant.

Type de ressource AWS	Format ARN (Amazon Resource Name)
Application	arn:*partition*:serverlessrepo:*region*:*account-id*:applications/*application-name*

Pour de plus amples informations sur le format des ARN, veuillez consulter Noms ARN (Amazon Resource Name) et espaces de noms du service AWS.

Voici un exemple de stratégie qui accorde des autorisations requises pour l'opération d'API`serverlessrepo:ListApplications`action sur tousAWSAWS. Dans l'implémentation actuelle, le paramètreAWS Serverless Application Repositoryne prend pas en charge l'identification spécifiqueAWSà l'aide de laAWSARN de ressource (également appelés autorisations au niveau des ressources) pour certaines actions d'API. Dans ce cas, vous devez spécifier un caractère générique (*).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListExistingApplications",
            "Effect": "Allow",
            "Action": [
                "serverlessrepo:ListApplications"
            ],
            "Resource": "*"
        }
    ]
}
```

Pour obtenir un tableau montrant tous les éléments deAWS Serverless Application RepositoryLes actions d'API et l'APIAWSauxquelles elles s'appliquent, consultezAWS Serverless Application RepositoryAutorisations d'API : Référence des actions et ressources (p. 50).

Clés de condition

AWS Serverless Application Repository ne fournit pas de clés de condition spécifiques au service, mais prend en charge l'utilisation de certaines clés de condition globales. Pour voir toutes les clés de condition globales AWS, consultez Clés de contexte de condition globales AWS dans le IAM Guide de l'utilisateur.

Examples

Pour voir des exemples de stratégies AWS Serverless Application Repository basées sur l'identité, consultez Exemples de stratégies AWS Serverless Application Repository basées sur l'identité (p. 40).

Stratégies AWS Serverless Application Repository basées sur les ressources

Les stratégies basées sur les ressources déterminent les actions et les conditions requises qu'un mandataire spécifié peut effectuer sur une ressource AWS Serverless Application Repository.

UnAWS Serverless Application Repository Applicationest le principalAWSressource dans laAWS Serverless Application Repository. Vous pouvez ajouter des autorisations à la stratégie associée à

une application AWS Serverless Application Repository. Les stratégies d'autorisations attachées aux applications AWS Serverless Application Repository sont appelées stratégies basées sur une ressource (ou stratégies d'application). Vous pouvez utiliser les stratégies d'application AWS Serverless Application Repository pour gérer les autorisations de déploiement d'applications.

Les stratégies d'applications AWS Serverless Application Repository sont principalement utilisées par les éditeurs pour accorder aux consommateurs l'autorisation de déployer leurs applications, et les opérations connexes, telles que la recherche et l'affichage des détails de ces applications. Les éditeurs peuvent définir les autorisations d'application selon les trois catégories suivantes :

- Privé – Applications qui ont été créées avec le même compte et qui n'ont pas été partagées avec un autre compte. Vous avez l'autorisation de déployer des applications créées à l'aide de votreAWS.
- Partagé en privé –Les applications que l'éditeur a explicitement partagées avec un ensemble spécifique deAWSLes comptes ouAWSOrganizations. Vous avez l'autorisation de déployer des applications qui ont été partagées avec votreAWSCompte ouAWSOrganisation.
- Partagé publiquement – Applications que l'éditeur a partagées avec tout le monde. Vous avez l'autorisation de déployer n'importe quelle application partagée publiquement.

Vous pouvez accorder des autorisations à l'aide de l'outil d'AWS CLI, leAWSou les kits SDKAWS Management Console.

Examples

Pour obtenir des exemples de gestion des stratégies AWS Serverless Application Repository basées sur les ressources, veuillez consulter Exemples de stratégies basées sur les ressources AWS Serverless Application Repository (p. 46).

Autorisation basée sur les balises AWS Serverless Application Repository

AWS Serverless Application Repository ne prend pas en charge le contrôle d'accès aux ressources ou aux actions basées sur des balises.

Rôles IAM d'AWS Serverless Application Repository

Un rôle IAM est une entité au sein de votre compteAWS qui dispose d'autorisations spécifiques.

Utilisation d'informations d'identification temporaires avec la gestion de configuration d'AWS Serverless Application Repository

Vous pouvez utiliser des informations d'identification temporaires pour vous connecter avec la fédération, endosser un rôle IAM, ou encore pour endosser un rôle entre comptes. Vous obtenez des informations d'identification de sécurité temporaires en appelant des opérations d'API AWS STS comme AssumeRole ou GetFederationToken.

AWS Serverless Application Repository prend en charge l'utilisation des informations d'identification temporaires.

Rôles liés à un service

AWS Serverless Application Repository ne prend pas en charge les rôles liés à un service.

Rôles de service

AWS Serverless Application Repository ne prend pas en charge les rôles de service.

Exemples de stratégies AWS Serverless Application Repository basées sur l'identité

Par défaut, les utilisateurs et les rôles IAM ne sont pas autorisés à créer ou modifier les ressources AWS Serverless Application Repository. Ils ne peuvent pas non plus exécuter des tâches à l'aide de AWS Management Console, AWS CLI ou de l'API AWS. Un administrateur IAM doit créer des stratégies IAM autorisant les utilisateurs et les rôles à exécuter des opérations d'API spécifiques sur les ressources spécifiées dont ils ont besoin. Il doit ensuite attacher ces stratégies aux utilisateurs ou aux groupes IAM ayant besoin de ces autorisations.

Pour apprendre à créer une stratégie basée sur l'identité IAM à l'aide de ces exemples de document de stratégie JSON, consultez Création de stratégies dans l'onglet JSON dans le IAM Guide de l'utilisateur.

Rubriques

Bonnes pratiques en matière de stratégies

Les stratégies basées sur l'identité sont très puissantes. Elles déterminent si une personne peut créer, consulter ou supprimer des ressources AWS Serverless Application Repository dans votre compte. Ces actions peuvent entraîner des frais pour votre compte AWS. Lorsque vous créez ou modifiez des stratégies basées sur l'identité, suivez ces instructions et recommandations :

- Accorder l'accès avec le moindre privilège – Lorsque vous créez des stratégies personnalisées, accordez uniquement les autorisations requises pour exécuter une seule tâche. Commencez avec un minimum d'autorisations et accordez-en d'autres si nécessaire. Cette méthode est plus sûre que de commencer avec des autorisations trop permissives et d'essayer de les restreindre plus tard. Pour plus d'informations, consultez Accorder le privilège le plus faible dans le IAM Guide de l'utilisateur.
- Activer MFA pour les opérations sensibles – Pour plus de sécurité, obligez les utilisateurs IAM à utiliser l'authentification multi-facteurs (MFA) pour accéder à des ressources ou à des opérations d'API sensibles. Pour plus d'informations, consultez Utilisation de Multi-Factor Authentication (MFA) dans AWS dans le IAM Guide de l'utilisateur.
- Utiliser des conditions de stratégie pour une plus grande sécurité – Tant que cela reste pratique pour vous, définissez les conditions dans lesquelles vos stratégies basées sur l'identité autorisent l'accès à une ressource. Par exemple, vous pouvez rédiger les conditions pour spécifier une plage d'adresses IP autorisées d'où peut provenir une demande. Vous pouvez également écrire des conditions pour autoriser les requêtes uniquement à une date ou dans une plage de temps spécifiée, ou pour imposer l'utilisation de SSL ou de MFA. Pour de plus amples informations, veuillez consulterIAMÉléments de stratégie JSON : Conditiondans leIAM Guide de l'utilisateur.

Utilisation de la console AWS Serverless Application Repository

La console AWS Serverless Application Repository fournit un environnement intégré vous permettant de découvrir et de gérer les applications AWS Serverless Application Repository. La console offre de nombreuses fonctions et flux de travail qui exigent souvent des autorisations pour gérer une application AWS Serverless Application Repository en plus des autorisations propres à l'API documentées dans AWS Serverless Application RepositoryAutorisations d'API : Référence des actions et ressources (p. 50).

Pour de plus amples informations sur les autorisations qui sont nécessaires pour utiliser la console AWS Serverless Application Repository, veuillez consulter Exemples de stratégies gérées par le client (p. 41).

Autoriser les utilisateurs à afficher leurs propres autorisations

Cet exemple montre comment créer une stratégie qui permet aux utilisateurs IAM d'afficher les stratégies en ligne et gérées attachées à leur identité d'utilisateur. Cette stratégie inclut les autorisations nécessaires pour réaliser cette action sur la console ou par programmation à l'aide de l'AWS CLI ou de l'API AWS.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Exemples de stratégies gérées par le client

Cette section fournit un ensemble d'exemples de stratégies que vous pouvez associer à un utilisateur. Si vous créez des stratégies pour la première fois, nous vous recommandons de commencer par créer un utilisateur IAM dans votre compte et de lui attacher les stratégies dans l'ordre. Vous pouvez également utiliser ces exemples pour créer une stratégie personnalisée unique qui inclut des autorisations pour effectuer plusieurs actions, puis l'attacher à l'utilisateur.

Pour de plus amples informations sur la façon d'attacher des stratégies aux utilisateurs, veuillez consulter Ajout d'autorisations à un utilisateur dans le IAM Guide de l'utilisateur.

Exemples

- Exemple 1 de l'éditeur : Autoriser un éditeur à répertorier les applications (p. 42)
- Exemple 2 de l'éditeur : Autoriser un éditeur à afficher les détails d'une application ou d'une version de l'application (p. 42)
- Exemple 3 de l'éditeur : Autoriser un éditeur à créer une application ou une version de l'application (p. 43)
- Exemple 4 de l'éditeur : Autoriser un éditeur à créer une stratégie d'application pour partager des applications avec d'autres (p. 43)

Exemple 1 de l'éditeur : Autoriser un éditeur à répertorier les applications

Un utilisateur IAM de votre compte doit disposer des autorisations pour l'opération `serverlessrepo:ListApplications` avant d'avoir accès au contenu de la console. Lorsque vous accordez ces autorisations, la console peut afficher la liste desAWS Serverless Application Repositoryapplications dans leAWScréé dans le compteAWSRégion à laquelle l'utilisateur appartient.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListExistingApplications",
            "Effect": "Allow",
            "Action": [
                "serverlessrepo:ListApplications"
            ],
            "Resource": "*"
        }
    ]
}
```

Exemple 2 de l'éditeur : Autoriser un éditeur à afficher les détails d'une application ou d'une version de l'application

Un utilisateur peut sélectionner une application AWS Serverless Application Repository et en afficher les détails. Ces détails incluent l'auteur, la description, les versions et les autres informations de configuration. Pour ce faire, l'utilisateur a besoin d'autorisations pour les opérations d'API `serverlessrepo:GetApplication` et `serverlessrepo:ListApplicationVersions` pour AWS Serverless Application Repository.

Dans l'exemple suivant, ces autorisations sont accordées pour l'application spécifique dont l'ARN (Amazon Resource Name) est spécifié comme la valeur `Resource`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewApplication",
            "Effect": "Allow",
            "Action": [
                "serverlessrepo:GetApplication",
                "serverlessrepo:ListApplicationVersions"
            ],
            "Resource": "arn:aws:serverlessrepo:region:account-id:applications/application-
name"
        }
    ]
```

```
}
```

Exemple 3 de l'éditeur : Autoriser un éditeur à créer une application ou une version de l'application

Si vous souhaitez autoriser un utilisateur à avoir des autorisations pour créer des applications AWS Serverless Application Repository, vous devez accorder des autorisations aux opérations `serverlessrepo:CreateApplication` et `serverlessrepo:CreateApplicationVersions`, comme illustré dans la stratégie suivante.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CreateApplication",
            "Effect": "Allow",
            "Action": [
                "serverlessrepo:CreateApplication",
                "serverlessrepo:CreateApplicationVersion",
            ],
            "Resource": "*"
        }
    ]
}
```

Exemple 4 de l'éditeur : Autoriser un éditeur à créer une stratégie d'application pour partager des applications avec d'autres

Pour que les utilisateurs partagent des applications avec d'autres, vous devez leur accorder des autorisations pour créer des stratégies d'application, comme illustré dans la stratégie suivante.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ShareApplication",
            "Effect": "Allow",
            "Action": [
                "serverlessrepo:PutApplicationPolicy",
                "serverlessrepo:GetApplicationPolicy",
            ],
            "Resource": "*"
        }
    ]
}
```

Exemple 1 du consommateur : Autoriser un consommateur à rechercher des applications

Pour que les consommateurs recherchent des applications, vous devez leur accorder les autorisations suivantes.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SearchApplications",
            "Effect": "Allow",
            "Action": [
                "serverlessrepo:SearchApplications"
            ],
            "Resource": "*"
        }
    ]
}
```

Exemple 2 du consommateur : Autoriser un consommateur à afficher les détails d'une application

Un utilisateur peut sélectionner une application AWS Serverless Application Repository et en afficher les détails, tels que l'auteur, la description, les versions et les autres informations de configuration. Pour ce faire, l'utilisateur doit disposer des autorisations pour les opérations AWS Serverless Application Repository suivantes.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewApplication",
            "Effect": "Allow",
            "Action": [
                "serverlessrepo:GetApplication",
                "serverlessrepo:ListApplicationVersions"
            ],
            "Resource": "*"
        }
    ]
}
```

Exemple 3 du consommateur : Autoriser un consommateur à déployer une application

Pour que les clients déploient des applications, vous devez leur accorder des autorisations pour exécuter un certain nombre d'opérations. La stratégie suivante accorde aux clients les autorisations requises.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DeployApplication",
            "Effect": "Allow",
            "Action": [
                "serverlessrepo:CreateCloudFormationChangeSet",
                "cloudformation:CreateChangeSet",
                "cloudformation:ExecuteChangeSet",
                "cloudformation:DescribeStacks"
```

```
            ],
            "Resource": "*"
        }
    ]
}
```

Note

Il est possible que le déploiement d'une application nécessite des autorisations pour utiliser desAWSAWS. Parce que leAWS Serverless Application Repositoryutilise le même mécanisme de déploiement sous-jacent queAWS CloudFormation, voir,Contrôle de l'accès avecAWSIdentity and Access Managementpour en savoir plus. Pour obtenir de l'aide sur les problèmes de déploiement liés aux autorisations, consultezRésolution de problèmes Autorisations IAM insuffisantes.

Exemple 4 du consommateur : Refuser l'accès aux ressources de déploiement

Lorsqu'une application est partagée en privé avec unAWS, par défaut, tous les utilisateurs de ce compte peuvent accéder aux ressources de déploiement de tous les autres utilisateurs du même compte. La stratégie suivante empêche les utilisateurs d'un compte d'accéder aux ressources de déploiement, qui sont stockées dans le compartiment Amazon S3 pour le AWS Serverless Application Repository.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyDeploymentAssetAccess",
            "Effect": "Deny",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::awsserverlessrepo-changesets*/*"
            ]
        }
    ]
}
```

Exemple 5 du consommateur : Empêcher un consommateur de rechercher et de déployer des applications publiques

Vous pouvez empêcher des utilisateurs d'effectuer certaines actions sur les applications.

La stratégie suivante s'applique aux demandes publiques en spécifiant que `serverlessrepo:applicationType` est `public`. Elle empêche les utilisateurs d'effectuer un certain nombre d'actions en spécifiant qu' `Effect` est `Deny`. Pour plus d'informations sur les clés de condition disponibles pour AWS Serverless Application Repository, consultez Actions, Ressources et Clés de condition pour AWS Serverless Application Repository.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Condition": {
                "StringEquals": {
                    "serverlessrepo:applicationType": "public"
                }
            },
            "Action": [
                "serverlessrepo:SearchApplications",
```

```
                "serverlessrepo:GetApplication",
                "serverlessrepo:CreateCloudFormationTemplate",
                "serverlessrepo:CreateCloudFormationChangeSet",
                "serverlessrepo:ListApplicationVersions",
                "serverlessrepo:ListApplicationDependencies"
            ],
            "Resource": "*",
            "Effect": "Deny"
        }
    ]
}
```

Note

Cette déclaration de stratégie peut également être utilisée en tant que stratégie de contrôle de service et appliquée à unAWSorganisation. Pour plus d'informations sur les stratégies de contrôle de service, consultez Stratégies de contrôle de service dans le Manuel de l'utilisateur AWS Organizations.

Exemples de stratégies basées sur les ressources AWS Serverless Application Repository

Les stratégies d'autorisations attachées aux applications AWS Serverless Application Repository sont appelées stratégies basées sur une ressource (ou stratégies d'application). Les stratégies basées sur les ressources déterminent les actions et les conditions requises qu'un mandataire spécifié peut effectuer sur une ressource AWS Serverless Application Repository.

UnAWS Serverless Application Repository Applicationest leAWSRessource dans leAWS Serverless Application Repository.AWS Serverless Application RepositoryLes stratégies d'applications sont principalement utilisées par les éditeurs pour accorder aux consommateurs l'autorisation de déployer leurs applications, et les opérations connexes, telles que la recherche et l'affichage des détails de ces applications.

Les éditeurs peuvent définir les autorisations d'application selon les trois catégories suivantes :

- Privé – Applications qui ont été créées avec le même compte et qui n'ont pas été partagées avec un autre compte. Seuls les consommateurs qui partagent votreAWSont l'autorisation de déployer des applications privées.
- Partagé en privé –Applications que l'éditeur a explicitement partagées avec un ensemble spécifique deAWS, ou avecAWSdans unAWSL'organisation. Les consommateurs sont autorisés à déployer des applications qui ont été partagées avec leurAWS.AWSL'organisation. Pour plus d'informations surAWS, consultez leManuel de l'utilisateur AWS Organizations.
- Partagé publiquement – Applications que l'éditeur a partagées avec tout le monde. Tous les consommateurs ont l'autorisation de déployer n'importe quelle application partagée publiquement.

Note

PourPartagé en privé, leAWS Serverless Application Repositoryne prend en chargeAWSComptesEn tant que mandataires. Les éditeurs peuvent accorder ou refuser à tous les utilisateurs dans unAWSen tant que groupe unique à un compteAWS Serverless Application Repositoryapplication. Les éditeurs ne peuvent pas accorder ou refuser des utilisateurs individuels dans unAWSà unAWS Serverless Application Repositoryapplication.

Pour de plus amples informations sur la définition des autorisations d'application à l'aide de la AWS Management Console, veuillez consulterSuppression d'une application (p. 16).

Pour de plus amples informations sur la définition des autorisations d'application à l'aide de AWS CLI et des exemples, veuillez consulter les sections suivantes.

Autorisations d'application (AWS CLIandAWSKits SDK)

Lorsque vous utilisez leAWS CLIou leAWSKits SDK pour définir les autorisations pour unAWS Serverless Application Repository, vous pouvez spécifier les actions suivantes :

Action	Description
GetApplication	Accorde une autorisation pour afficher des informations sur l'application.
CreateCloudFormationChangeSet	Accorde une autorisation pour déployer l'application. Remarque : Cette action nepasaccorder toute autre autorisation que celle permettant de déployer.
CreateCloudFormationTemplate	Accorde l'autorisation de créer un modèle AWS CloudFormation pour l'application.
ListApplicationVersions	Accorde l'autorisation de répertorier les versions de l'application.
ListApplicationDependencies	Accorde l'autorisation de répertorier les applications imbriquées dans l'application qu'elles contiennent.
SearchApplications	Accorde une autorisation pour rechercher l'application.
Déploiement	Cette action active toutes les actions répertoriées précédemment dans le tableau. Autrement dit, elle accorde l'autorisation d'afficher l'application, de la déployer, de lister les versions et de la rechercher.

Exemples de stratégies basées sur les ressources

Les exemples suivants montrent comment accorder des autorisations à l'aide de la AWS CLI. Pour de plus amples informations sur la façon d'accorder des autorisations à l'aide de l'AWS Management Console, veuillez consulter Suppression d'une application (p. 16).

Tous les exemples de cette section utilisent les commandes AWS CLI suivantes pour gérer les stratégies d'autorisations associées aux applications AWS Serverless Application Repository :

- put-application-policy
- get-application-policy

Rubriques

Exemple 1 : Partager une application avec un autre compte

Pour partager une application avec un autre compte spécifique, tout en l'empêchant d'être partagée avec d'autres, vous spécifiez leAWSL'ID de compte que vous souhaitez partager en tant que mandataire. Ceci correspond à la définition de l'application sur partagé en privé. Pour ce faire, exécutez la commande AWS CLI suivante.

```
aws serverlessrepo put-application-policy \
--region region \
--application-id application-arn \
--statements Principals=account-id,Actions=Deploy
```

> **Note**
>
> Partagé en privéLes applicationspriées ne peuvent être utilisées que dans le mêmeAWSRégion dans laquelle l'application est créée.

Exemple 2 : Partager publiquement une application

Pour rendre une application publique, vous la partagez avec tous en spécifiant « * » comme mandataire, comme illustré dans l'exemple suivant. Les applications qui sont partagées publiquement sont disponibles dans toutes les régions.

```
aws serverlessrepo put-application-policy \
--region region \
--application-id application-arn \
--statements Principals=*,Actions=Deploy
```

> **Note**
>
> Pour partager publiquement une application, elle doit avoir à la fois définies les propriétés `LicenseUrl` et `SemanticVersion`.

Exemple 3 : Rendre une application privée

Vous pouvez rendre une application privée, afin qu'elle ne soit partagée avec personne et puisse uniquement déployée par leAWSqui le possède. Pour ce faire, vous effacez les mandataires et les actions de la stratégie, ce qui supprime également les autorisations accordées à d'autres comptes dans votreAWSpour le déploiement de votre application.

```
aws serverlessrepo put-application-policy \
--region region \
--application-id application-arn \
--statements '[]'
```

> **Note**
>
> PrivéLes applicationspriées ne peuvent être utilisées que dans le mêmeAWSRégion dans laquelle l'application est créée.

Exemple 4 : Spécification de plusieurs comptes et autorisations

Vous pouvez accorder plusieurs autorisations, à plusieursAWS. Pour ce faire, vous spécifiez des listes en tant que mandataire et actions, comme indiqué dans l'exemple suivant.

```
aws serverlessrepo put-application-policy \
--region region \
--application-id application-arn \
--statements Principals=account-id-1,account-
id-2,Actions=GetApplication,CreateCloudFormationChangeSet
```

Exemple 5 : Partager une application avec tous les comptes dans unAWSOrganisation

Des autorisations peuvent être accordées à tous les utilisateurs dans unAWSL'organisation. Pour ce faire, spécifiez l'ID de votre organisation, comme dans l'exemple suivant.

```
aws serverlessrepo put-application-policy \
--region region \
--application-id application-arn \
--statements Principals=*,PrincipalOrgIDs=org-id,Actions=Deploy,UnshareApplication
```

Pour plus d'informations surAWS, consultez leManuel de l'utilisateur AWS Organizations.

> Note
>
> Vous ne pouvez spécifier que leAWSL'organisation que votreAWSest un membre de. Si vous essayez de spécifier unAWSL'organisation dont vous n'êtes pas membre, une erreur se produit. Pour partager votre application avec votreAWS, vous devez inclure l'autorisation pour le`UnshareApplication`action, au cas où le partage doit être révoqué à l'avenir.

Exemple 6 : Partage d'une application avec certains comptes dans unAWSOrganisation

Des autorisations peuvent être accordées à des comptes spécifiques dans unAWSL'organisation. Pour ce faire, spécifiez une liste deAWSLes comptes en tant que mandataires et l'ID de votre organisation, comme dans l'exemple suivant.

```
aws serverlessrepo put-application-policy \
--region region \
--application-id application-arn \
--statements Principals=account-id-1,account-id-2,PrincipalOrgIDs=org-
id,Actions=Deploy,UnshareApplication
```

> Note
>
> Vous ne pouvez spécifier que leAWSL'organisation que votreAWSest un membre de. Si vous essayez de spécifier unAWSL'organisation dont vous n'êtes pas membre, une erreur se produit. Pour partager votre application avec votreAWS, vous devez inclure l'autorisation pour le`UnshareApplication`action, au cas où le partage doit être révoqué à l'avenir.

Exemple 7 : Récupérer une stratégie d'application

Pour afficher la stratégie en cours d'une application, par exemple pour voir si elle est actuellement partagée, vous utilisez la commande `get-application-policy`, tel qu'illustré dans l'exemple suivant.

```
aws serverlessrepo get-application-policy \
--region region \
--application-id application-arn
```

Exemple 8 : Autoriser l'imbrication de l'application par certains comptes

Tout le monde peut imbriquer des applications publiques. Pour autoriser l'imbrication de votre application par certains comptes uniquement, vous devez définir les autorisations minimales suivantes, comme illustré à l'exemple suivant.

```
aws serverlessrepo put-application-policy \
--region region \
--application-id application-arn \
--statements Principals=account-id-1,account-
id-2,Actions=GetApplication,CreateCloudFormationTemplate
```

AWS Serverless Application RepositoryAutorisations d'API : Référence des actions et ressources

Lorsque vous configurez un contrôle d'accès (p. 33) et écrivez des stratégies d'autorisations que vous pouvez attacher à une identité IAM (stratégies basées sur une identité), vous pouvez utiliser la table ci-dessous comme référence. La .EACHAWS Serverless Application RepositoryOpération d'API, les actions correspondantes pour lesquelles vous pouvez accorder des autorisations d'exécution etAWSRessource pour laquelle vous pouvez accorder les autorisations. Vous spécifiez les actions dans le champ `Action` de la stratégie ainsi que la valeur des ressources dans le champ `Resource` de la stratégie.

Pour spécifier une action, utilisez le préfixe `serverlessrepo:` suivi du nom de l'opération d'API (par exemple, `serverlessrepo:ListApplications`).

Opération	URI	Méthode	AWSRessources (ARN)
Opération : ListApplications Autorisations requises : serverlessrepo:ListApplications	/applications	GET	*
Opération : CreateApplication Autorisations requises : serverlessrepo:CreateApplication	/applications	POST	*
Opération : GetApplication Autorisations requises : serverlessrepo:GetApplication	/applications/*id-application*	GET	arn:aws:serverlessrepo:*région*:*id-compte*:applications/*nom-application*
Opération : DeleteApplication Autorisations requises : serverlessrepo:DeleteApplication	/applications/*id-application*	DELETE	arn:aws:serverlessrepo:*région*:*id-compte*:applications/*nom-application*
Opération : UpdateApplication Autorisations requises : serverlessrepo:UpdateApplication	/applications/*id-application*	CORRECTIF	arn:aws:serverlessrepo:*région*:*id-compte*:applications/*nom-application*

Opération	URI	Méthode	AWSRessources (ARN)
Opération : CreateCloudFormationChangeSet Autorisations requises : serverlessrepo:CreateCloudFormationChangeSet	/applications/*id-application*/changesets	POST	arn:aws:serverlessrepo:*région*:*id-compte*:applications/*nom-application*
Opération : getApplicationPolicy Autorisations requises : serverlessrepo:GetApplicationPolicy	/applications/*id-application*/policy	GET	arn:aws:serverlessrepo:*région*:*id-compte*:applications/*nom-application*
Opération : PutApplicationPolicy Autorisations requises : serverlessrepo:PutApplicationPolicy	/applications/*id-application*/policy	PUT	arn:aws:serverlessrepo:*région*:*id-compte*:applications/*nom-application*
Opération : ListApplicationVersions Autorisations requises : serverlessrepo:ListApplicationVersions	/applications/*id-application*/versions	GET	arn:aws:serverlessrepo:*région*:*id-compte*:applications/*nom-application*
Opération : CreateApplicationVersion Autorisations requises : serverlessrepo:CreateApplicationVersion	/applications/*id-application*/versions/*version-sémantique*	PUT	arn:aws:serverlessrepo:*région*:*id-compte*:applications/*nom-application*
Opération : ListApplicationDependencies Autorisations requises : serverlessrepo:ListApplicationDependencies	/applications/*id-application*/dependencies	GET	arn:aws:serverlessrepo:*région*:*id-compte*:applications/*nom-application*
Opération : SearchApplications Autorisations requises : serverlessrepo:SearchApplications	Non applicable	s/o	*

Résolution des problèmes d'identité et d'accès AWS Serverless Application Repository

Utilisez les informations suivantes pour identifier et résoudre les problèmes courants que vous pouvez rencontrer lorsque vous travaillez avec AWS Serverless Application Repository et IAM.

Rubriques

- Je veux permettre à des personnes extérieures à mon compte AWS d'accéder à mes ressources AWS Serverless Application Repository (p. 53)

Je ne suis pas autorisé à effectuer une action dans AWS Serverless Application Repository

Si AWS Management Console indique que vous n'êtes pas autorisé à exécuter une action, vous devez contacter votre administrateur pour obtenir de l'aide. Votre administrateur est la personne qui vous a fourni votre nom d'utilisateur et votre mot de passe.

L'exemple d'erreur suivant se produit lorsque l'utilisateur IAM `mateojackson` tente d'utiliser la console pour afficher des informations détaillées concernant une application mais ne dispose pas des autorisations `serverlessrepo:GetApplication`.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 serverlessrepo:GetApplication on resource: my-example-application
```

Dans ce cas, Mateo demande à son administrateur de mettre à jour ses stratégies pour l'autoriser à accéder à la ressource *my-example-application* en utilisant l'opération `serverlessrepo:GetApplication`.

Je ne suis pas autorisé à exécuter iam:PassRole

Si vous recevez un message d'erreur selon lequel vous n'êtes pas autorisé à exécuter l'action `iam:PassRole`, vous devez contacter votre administrateur pour obtenir de l'aide. Votre administrateur est la personne qui vous a fourni votre nom d'utilisateur et votre mot de passe. Demandez à cette personne de mettre à jour vos stratégies pour vous permettre de transmettre un rôle à AWS Serverless Application Repository.

Certains services AWS vous permettent de transmettre un rôle existant à ce service, au lieu de créer un nouveau rôle de service ou rôle lié à un service. Pour ce faire, un utilisateur doit disposer des autorisations nécessaires pour transmettre le rôle au service.

L'exemple d'erreur suivant se produit lorsqu'un utilisateur IAM nommé `marymajor` essaie d'utiliser la console pour exécuter une action dans AWS Serverless Application Repository. Toutefois, l'action nécessite que le service ait des autorisations accordées par un rôle de service. Mary ne dispose pas des autorisations nécessaires pour transférer le rôle au service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform: iam:PassRole
```

Dans ce cas, Mary demande à son administrateur de mettre à jour ses stratégies pour lui permettre d'exécuter l'action `iam:PassRole`.

Je veux afficher mes clés d'accès

Une fois que vous avez créé vos clés d'accès IAM, vous pouvez afficher votre ID de clé d'accès à tout moment. Toutefois, vous ne pouvez pas afficher à nouveau votre clé d'accès secrète. Si vous perdez votre clé d'accès secrète, vous devez créer une nouvelle paire de clés.

Les clés d'accès se composent de deux parties : un ID de clé d'accès (par exemple, `AKIAIOSFODNN7EXAMPLE`) et une clé d'accès secrète (par exemple, `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`). À l'instar d'un nom d'utilisateur et un mot de passe, vous devez utiliser à la fois l'ID de clé d'accès et la clé d'accès secrète pour authentifier vos demandes. Gérez vos clés d'accès de manière aussi sécurisée que votre nom d'utilisateur et votre mot de passe.

Important

Ne communiquez pas vos clés d'accès à un tiers, même pour qu'il vous aide à trouver votre ID utilisateur canonique. En effet, vous lui accorderiez ainsi un accès permanent à votre compte.

Lorsque vous créez une paire de clé d'accès, enregistrez l'ID de clé d'accès et la clé d'accès secrète dans un emplacement sécurisé. La clé d'accès secrète est accessible uniquement au moment de sa création. Si vous perdez votre clé d'accès secrète, vous devez ajouter de nouvelles clés d'accès pour votre utilisateur IAM. Vous pouvez avoir un maximum de deux clés d'accès. Si vous en avez déjà deux, vous devez supprimer une paire de clés avant d'en créer une nouvelle. Pour obtenir plus d'informations, consultez Gestion des clés d'accès dans le IAM Guide de l'utilisateur.

Je suis administrateur et je veux autoriser d'autres utilisateurs à accéder à la gestion de configuration d'AWS Serverless Application Repository

Pour permettre à d'autres utilisateurs d'accéder à AWS Serverless Application Repository, vous devez créer une entité IAM (utilisateur ou rôle) pour la personne ou l'application qui a besoin de l'accès. Ils utiliseront les informations d'identification de cette entité pour accéder à AWS. Vous devez ensuite associer une stratégie à l'entité qui leur accorde les autorisations appropriées dans AWS Serverless Application Repository.

Pour démarrer immédiatement, consultez Création de votre premier groupe et utilisateur délégué IAM dans le IAM Guide de l'utilisateur.

Je veux permettre à des personnes extérieures à mon compte AWS d'accéder à mes ressources AWS Serverless Application Repository

Vous pouvez créer un rôle que les utilisateurs provenant d'autres comptes ou les personnes extérieures à votre organisation peuvent utiliser pour accéder à vos ressources. Vous pouvez spécifier qui est approuvé pour assumer le rôle. Pour les services qui prennent en charge les stratégies basées sur les ressources ou les listes de contrôle d'accès (ACL), vous pouvez utiliser ces stratégies pour accorder aux personnes l'accès à vos ressources.

Pour en savoir plus, consultez les éléments suivants :

- Pour savoir si AWS Serverless Application Repository prend en charge ces fonctionnalités, consultez Comment AWS Serverless Application Repository fonctionne avec IAM (p. 35).
- Pour savoir comment fournir un accès à vos ressources sur les Comptes AWS que vous détenez, consultez Octroi à un utilisateur IAM de l'autorisation d'accès à un autre Compte AWS vous appartenant dans le IAM Guide de l'utilisateur.
- Pour savoir comment fournir l'accès à vos ressources à des Comptes AWS tiers, consultez Octroi d'un accès à des Comptes AWS appartenant à des tiers dans le IAM Guide de l'utilisateur.
- Pour savoir comment fournir un accès par le biais de la fédération d'identité, consultez Octroi d'accès à des utilisateurs authentifiés en externe (fédération d'identité) dans le IAM Guide de l'utilisateur.
- Pour en savoir plus sur la différence entre l'utilisation des rôles et des stratégies basées sur les ressources pour l'accès entre comptes, consultez Différence entre les rôles IAM et les stratégies basées sur les ressources dans le IAM Guide de l'utilisateur.

Journalisation et surveillance dans AWS Serverless Application Repository

La surveillance est essentielle pour assurer la fiabilité, la disponibilité et les performances de vos solutions AWS. Vous devez recueillir les données de surveillance de toutes les parties de votre solution AWS de manière à pouvoir déboguer plus facilement une éventuelle défaillance à plusieurs points. AWS fournit plusieurs outils pour surveiller vos ressources AWS Serverless Application Repository et répondre aux incidents potentiels, comme les suivants :

Journaux AWS CloudTrail

AWS Serverless Application Repository est intégré à AWS CloudTrail, une service qui fournit un enregistrement des actions réalisées par un utilisateur, un rôle ou un service AWS dans AWS Serverless Application Repository. CloudTrail capture tous les appels d'API pour AWS Serverless Application Repository en tant qu'événements.

Rubriques

Journalisation des appels d'API AWS Serverless Application Repository avec AWS CloudTrail

AWS Serverless Application Repository est intégré à AWS CloudTrail, un service qui fournit un enregistrement des actions réalisées par un utilisateur, un rôle ou un service AWS dans AWS Serverless Application Repository. CloudTrail capture tous les appels d'API pour AWS Serverless Application Repository en tant qu'événements. Les appels capturés incluent des appels de la console AWS Serverless Application Repository et les appels de code vers les opérations d'API AWS Serverless Application Repository.

Si vous créez un journal de suivi, vous pouvez activer la diffusion en continu des événements CloudTrail sur un compartiment Amazon S3, y compris les événements pour AWS Serverless Application Repository. Si vous ne configurez pas de journal de suivi, vous pouvez toujours afficher les événements les plus récents dans la console CloudTrail dans Event history (Historique des événements).

En utilisant les informations collectées par CloudTrail, vous pouvez déterminer quelle demande a été faite à AWS Serverless Application Repository. Vous pouvez aussi déterminer l'adresse IP à partir de laquelle la demande a été faite, qui a effectué la demande, quand elle a eu lieu et autres informations supplémentaires.

Pour en savoir plus sur CloudTrail, consultez AWS CloudTrail User Guide.

Informations AWS Serverless Application Repository dans CloudTrail

CloudTrail est activé sur votre compte AWS lorsque vous créez le compte. Lorsqu'une activité a lieu dans AWS Serverless Application Repository, cette activité est enregistrée dans un événement CloudTrail avec d'autres événements de service AWS dans Historique des événements. Vous pouvez afficher, rechercher et télécharger les événements récents dans votre compte AWS. Pour plus d'informations, consultez Affichage des événements avec l'historique des événements CloudTrail .

Pour un enregistrement continu des événements dans votre compte AWS, y compris les événements pour AWS Serverless Application Repository, créez un journal de suivi. Une journal de suivi permet à

CloudTrail de livrer les fichiers journaux dans un compartiment Amazon S3. Par défaut, lorsque vous créez un journal de suivi dans la console, il s'applique à toutes les régions AWS. Le journal de suivi consigne les événements de toutes les régions AWS dans la partition AWS et livre les fichiers journaux dans le compartiment Amazon S3 de votre choix. En outre, vous pouvez configurer d'autres services AWS pour analyser plus en profondeur les données d'événement collectées dans les journaux CloudTrail et agir sur celles-ci. Pour plus d'informations, consultez les ressources suivantes :

- Présentation de la création d'un journal de suivi
- CloudTrail Intégrations et services pris en charge par
- Configuration des Notifications de Amazon SNS pour CloudTrail
- Réception de fichiers journaux CloudTrail de plusieurs régions et Réception de fichiers journaux CloudTrail de plusieurs comptes

Toutes les actions AWS Serverless Application Repository sont consignées par CloudTrail et documentées sur la page Ressources AWS Serverless Application Repository. À titre d'exemple, les appels vers les opérations `CreateApplication`, `UpdateApplications` et `ListApplications` et génèrent des entrées dans les fichiers journaux CloudTrail.

Chaque événement ou entrée du journal contient des informations sur la personne qui a généré la demande. Les informations relatives à l'identité permettent de déterminer les éléments suivants :

- Si la demande a été effectuée avec des informations d'identification utilisateur racine ou AWS Identity and Access Management (IAM).
- Si la demande a été effectuée avec des informations d'identification de sécurité temporaires pour un rôle ou un utilisateur fédéré.
- Si la requête a été effectuée par un autre service AWS.

Pour plus d'informations, consultez la section Élément userIdentity CloudTrail.

Présentation des entrées des fichiers journaux AWS Serverless Application Repository

Un journal de suivi est une configuration qui active la livraison d'événements en tant que fichiers journaux à un compartiment Amazon S3 que vous spécifiez. Les fichiers journaux CloudTrail contiennent une ou plusieurs entrées de journal. Un événement représente une demande individuelle à partir d'une source quelconque et comprend des informations sur l'action demandée, sur tous les paramètres, les paramètres de la demande, etc. Les fichiers journaux CloudTrail ne sont pas des séries ordonnées retraçant les appels d'API publics. Ils ne suivent aucun ordre précis.

L'exemple suivant montre une entrée de journal CloudTrail qui illustre l'action `CreateApplication`.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "Root",
        "principalId": "999999999999",
        "arn": "arn:aws:iam::999999999999:root",
        "accountId": "999999999999",
        "accessKeyId": "ASIAUVPLBDH76HEXAMPLE",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2018-07-30T16:40:42Z"
            }
        },
        "invokedBy": "signin.amazonaws.com"
    },
```

```
    "eventTime": "2018-07-30T17:37:37Z",
    "eventSource": "serverlessrepo.amazonaws.com",
    "eventName": "CreateApplication",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "72.21.217.161",
    "userAgent": "signin.amazonaws.com",
    "requestParameters": {
        "licenseBody": "<content of license>",
        "sourceCodeUrl": "<sample url>",
        "spdxLicenseId": "<sample license id>",
        "readmeBody": "<content of readme>",
        "author": "<author name>",
        "templateBody": "<content of SAM template>",
        "name": "<application name>",
        "semanticVersion": "<version>",
        "description": "<content of description>",
        "homePageUrl": "<sample url>",
        "labels": [
            "<label1>",
            "<label2>"
        ]
    },
    "responseElements": {
        "licenseUrl": "<url to access content of license>",
        "readmeUrl": "<url to access content of readme>",
        "spdxLicenseId": "<sample license id>",
        "creationTime": "2018-07-30T17:37:37.045Z",
        "author": "<author name>",
        "name": "<application name>",
        "description": "<content of description>",
        "applicationId": "arn:aws:serverlessrepo:us-
east-1:999999999999:applications/<application name>",
        "homePageUrl": "<sample url>",
        "version": {
            "applicationId": "arn:aws:serverlessrepo:us-
east-1:999999999999:applications/<application name>",
            "semanticVersion": "<version>",
            "sourceCodeUrl": "<sample url>",
            "templateUrl": "<url to access content of SAM template>",
            "creationTime": "2018-07-30T17:37:37.027Z",
            "parameterDefinitions": [
                {
                    "name": "<parameter name>",
                    "description": "<parameter description>",
                    "type": "<parameter type>"
                }
            ]
        },
        "labels": [
            "<label1>",
            "<label2>"
        ]
    },
    "requestID": "3f50d899-941f-11e8-ab18-01063f863be5",
    "eventID": "a66a6490-d388-4a4f-8c7b-9d6ec61ab262",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "recipientAccountId": "999999999999"
}
```

Validation de la conformité pour l'AWS Serverless Application Repository

Les auditeurs tiers évaluent la sécurité et la conformité AWS Serverless Application Repository dans le cadre de plusieurs programmes de conformité AWS. Il s'agit notamment des certifications SOC, PCI, FedRAMP, et autres.

Pour obtenir la liste desAWSqui entrent dans le champ d'application de certains programmes de conformité, consultezAWSServices concernés par le programme de conformité. Pour obtenir des informations générales, consultez Programmes de conformité AWS.

Vous pouvez télécharger les rapports d'audit tiers avec AWS Artifact. Pour de plus amples informations, veuillez consulterTéléchargement des rapports dansAWSArtefact.

Votre responsabilité en matière de conformité lorsque vous utilisez AWS Serverless Application Repository est déterminée par la sensibilité de vos données, les objectifs de conformité de votre entreprise, ainsi que la législation et la réglementation en vigueur. AWS fournit les ressources suivantes pour faciliter le respect de la conformité :

- Guides de démarrage rapide de la sécurité et de la conformité – Ces guides de déploiement proposent des considérations architecturales et indiquent les étapes à suivre pour déployer des environnements de référence centrés sur la sécurité et la conformité sur AWS.
- Ressources de conformité AWS – Cet ensemble de manuels et de guides peut s'appliquer à votre secteur et à votre emplacement.
- AWS Config – Ce service AWS permet d'évaluer comment les configurations de vos ressources se conforment aux pratiques internes, aux normes et aux directives industrielles.
- AWS Security Hub – Ce service AWS fournit une vue complète de votre état de sécurité au sein AWS qui vous permet de vérifier votre conformité aux normes du secteur et aux bonnes pratiques de sécurité.

Résilience dans la gestion de configuration d'AWS Serverless Application Repository

L'infrastructure mondiale dAWS repose sur des régions et des zones de disponibilité AWS. Les régions AWS fournissent plusieurs zones de disponibilité physiquement séparées et isolées, reliées par un réseau à latence faible, à débit élevé et à forte redondance. Avec les zones de disponibilité, vous pouvez concevoir et exploiter des applications et des bases de données qui basculent automatiquement d'une zone de disponibilité à l'autre sans interruption. Les zones de disponibilité sont plus hautement disponibles, tolérantes aux pannes et évolutives que les infrastructures traditionnelles à un ou plusieurs centres de données.

Pour plus d'informations sur les régions et les zones de disponibilité AWS, consultez Infrastructure mondiale AWS.

Sécurité de l'infrastructure dans la gestion de configuration d'AWS Serverless Application Repository

En tant que service géré, AWS Serverless Application Repository est protégé par les procédures de sécurité du réseau mondial AWS qui sont décrites dans le livre blanc Amazon Web Services : Présentation des procédures de sécurité.

Vous utilisez les appels d'API publiés dans AWS pour accéder à AWS Serverless Application Repository via le réseau. Les clients doivent prendre en charge le protocole TLS (Transport Layer Security) 1.0 ou version ultérieure. Nous recommandons TLS 1.2 ou version ultérieure. Les clients doivent également prendre en charge les suites de chiffrement PFS (Perfect Forward Secrecy) comme Ephemeral Diffie-Hellman (DHE) ou Elliptic Curve Ephemeral Diffie-Hellman (ECDHE) La plupart des systèmes modernes telles que Java 7 et versions ultérieures prennent en charge ces modes.

En outre, les demandes doivent être signées à l'aide d'un ID de clé d'accès et d'une clé d'accès secrète associée à un mandataire IAM. Vous pouvez également utiliser AWS Security Token Service (AWS STS) pour générer des informations d'identification de sécurité temporaires et signer les demandes.

Quotas AWS Serverless Application Repository

La .AWS Serverless Application RepositoryLe quota pour le nombre d'applications publiques qu'unAWScompte peut avoir dans chaqueAWSRégion . Ce quota s'applique par région et peut être augmenté. Pour demander une augmentation, utilisez la console Support Center.

Ressource	Quota par défaut
Applications publiques (parAWSPar compteAWSRegion)	100

Les quotas suivants s'appliquent au stockage disponible pour les packages de code et les stratégies d'application. Vous ne pouvez pas modifier ces quotas.

Ressource	Quota
FreeAmazon S3Stockage pour les packages de code (parAWSPar compteAWSRegion)	5 Go
Longueur de la stratégie d'application	6 144 caractères

Dépannage de AWS Serverless Application Repository

Quand vous utilisez AWS Serverless Application Repository, vous pouvez rencontrer des problèmes lorsque vous créez, mettez à jour ou supprimez vos applications. Utilisez cette section pour vous aider à résoudre les problèmes courants que vous pourriez rencontrer. Vous pouvez également rechercher des réponses et publier des questions sur les forums AWS Serverless Application Repository.

> Note
>
> Les applications dans AWS Serverless Application Repository sont déployées avec AWS CloudFormation. Pour plus d'informations sur le dépannage des problèmes liés à AWS CloudFormation, consultez le Guide de dépannage AWS CloudFormation.

Rubriques

Impossible de rendre une application publique

Si vous ne pouvez pas rendre votre application publique, il se peut qu'il manque à votre application un fichier de licence approuvé par l'OSI (Open Source Initiative).

Pour rendre votre application publique, vous avez besoin d'un fichier de licence approuvé par l'OSI, ainsi que d'une version publiée avec succès de l'application avec l'URL du code source de la version. Une fois que l'application a été créée, vous ne pouvez pas mettre à jour la licence d'une application.

Si vous ne pouvez pas rendre votre application publique parce qu'il manque à votre application un fichier de licence, supprimez l'application et créez-en une nouvelle avec le même nom. Assurez-vous de lui fournir une ou plusieurs licences open source approuvées par l'organisation OSI (Open Source Initiative).

Un quota a été dépassé

Si vous recevez un message d'erreur indiquant qu'un quota a été dépassé, vérifiez si vous avez atteint un quota de ressources. Pour les quotas AWS Serverless Application Repository,veuillez consulter Quotas AWS Serverless Application Repository (p. 59).

Un fichier Lisez-moi (Readme) mis à jour ne s'affiche pas immédiatement

Quand vous rendez votre application publique, le contenu de votre application peut prendre jusqu'à 24 heures pour être mis à jour. If you experience delays longer than 24 hours, try contacting AWS Support for help. Pour plus d'informations, consultez les rubriques suivantes.

Vous ne pouvez pas déployer une application en raison d'autorisations IAM insuffisantes

Pour déployer une application AWS Serverless Application Repository, vous avez besoin des autorisations pour les ressources AWS Serverless Application Repository et les piles AWS CloudFormation. Il se peut également que vous ayez besoin d'une autorisation pour utiliser les services sous-jacents décrits dans l'application. Par exemple, si vous créez un compartiment Amazon S3 ou une table Amazon DynamoDB, vous avez besoin d'autorisations pour Amazon S3 ou DynamoDB.

Si vous rencontrez ce type de problème, examinez votre stratégie AWS Identity and Access Management (IAM) et vérifiez que vous disposez des autorisations nécessaires. For more information, see Controlling Access with AWS Identity and Access Management.

Vous ne pouvez pas déployer la même application deux fois

Le nom d'application que vous avez fourni est utilisé comme nom de la pile AWS CloudFormation. Si vous rencontrez des problèmes lors du déploiement d'une application, assurez-vous que vous n'avez pas de pile AWS CloudFormation, existante portant le même nom. Si tel est le cas, fournissez un autre nom d'application ou supprimez la pile existante pour déployer l'application avec le même nom.

Pourquoi mon application n'est-elle pas disponible publiquement

Par défaut, les applications sont privées. Pour rendre votre application publique, suivez la procédure décrite ici.

Contacter l'assistance

Dans certains cas, vous risquez de ne pas trouver de solutions de dépannage dans cette section ou via les forums AWS Serverless Application Repository. Si vous avez souscrit un planAWSPremium Support, vous pouvez créer une demande d'assistance technique à l'adresseAWSSupport.

Avant de contacterAWSSupport, veillez à obtenir l'ARN (Amazon Resource Name) de l'application pour laquelle vous avez des questions. L'ARN de l'application est disponible dans la console AWS Serverless Application Repository.

Operations

The AWS Serverless Application Repository REST API includes the following operations.

- CreateApplication (p. 64)

  Creates an application, optionally including an AWS SAM file to create the first application version in the same call.
- CreateApplicationVersion (p. 107)

  Creates an application version.
- CreateCloudFormationChangeSet (p. 91)

  Creates an AWS CloudFormation change set for the given application.
- DeleteApplication (p. 79)

  Deletes the specified application.
- GetApplication (p. 78)

  Gets the specified application.
- GetApplicationPolicy (p. 96)

  Gets the policy for the specified application.
- ListApplications (p. 63)

  Lists applications owned by the requester.
- ListApplicationVersions (p. 102)

  Lists versions for the specified application.
- PutApplicationPolicy (p. 97)

  Puts the policy for the specified application.
- UpdateApplication (p. 80)

  Updates the specified application.

Resources

The AWS Serverless Application Repository REST API includes the following resources.

Rubriques

Applications

URI

`/applications`

HTTP methods

GET

Operation ID: `ListApplications`

Lists applications owned by the requester.

Query parameters

Name	Type	Required	Description
`maxItems`	String	False	The total number of items to return.
`nextToken`	String	False	A token to specify where to start paginating.

Responses

Status code	Response model	Description
`200`	`ApplicationPage (p. 65)`	Success
`400`	`BadRequestException (p. 66)`	One of the parameters in the request is invalid.
`403`	`ForbiddenException (p. 66)`	The client is not authenticated.

Status code	Response model	Description
404	NotFoundException (p. 66)	The resource (for example, an access policy statement) specified in the request doesn't exist.
500	InternalServerErrorException (p. 66)	The AWS Serverless Application Repository service encountered an internal error.

POST

Operation ID: `CreateApplication`

Creates an application, optionally including an AWS SAM file to create the first application version in the same call.

Responses

Status code	Response model	Description
201	Application (p. 65)	Success
400	BadRequestException (p. 66)	One of the parameters in the request is invalid.
403	ForbiddenException (p. 66)	The client is not authenticated.
409	ConflictException (p. 66)	The resource already exists.
429	TooManyRequestsException (p. 66)	The client is sending more than the allowed number of requests per unit of time.
500	InternalServerErrorException (p. 66)	The AWS Serverless Application Repository service encountered an internal error.

Schemas

Request bodies

POST schema

```
{
  "name": "string",
  "description": "string",
  "author": "string",
  "spdxLicenseId": "string",
  "licenseBody": "string",
  "licenseUrl": "string",
  "readmeBody": "string",
  "readmeUrl": "string",
  "labels": [
    "string"
  ],
```

```
  "homePageUrl": "string",
  "semanticVersion": "string",
  "templateBody": "string",
  "templateUrl": "string",
  "sourceCodeUrl": "string"
}
```

Response bodies

ApplicationPage schema

```
{
  "applications": [
    {
      "applicationId": "string",
      "name": "string",
      "description": "string",
      "author": "string",
      "spdxLicenseId": "string",
      "labels": [
        "string"
      ],
      "creationTime": "string",
      "homePageUrl": "string"
    }
  ],
  "nextToken": "string"
}
```

Application schema

```
{
  "applicationId": "string",
  "name": "string",
  "description": "string",
  "author": "string",
  "spdxLicenseId": "string",
  "licenseUrl": "string",
  "readmeUrl": "string",
  "labels": [
    "string"
  ],
  "creationTime": "string",
  "homePageUrl": "string",
  "version": {
    "applicationId": "string",
    "semanticVersion": "string",
    "sourceCodeUrl": "string",
    "templateUrl": "string",
    "creationTime": "string",
    "parameterDefinitions": [
      {
        "name": "string",
        "defaultValue": "string",
        "description": "string",
        "type": "string",
        "noEcho": boolean,
        "allowedPattern": "string",
        "constraintDescription": "string",
        "minValue": integer,
        "maxValue": integer,
        "minLength": integer,
```

```
        "maxLength": integer,
        "allowedValues": [
          "string"
        ],
        "referencedByResources": [
          "string"
        ]
      }
    ]
  }
}
```

BadRequestException schema

```
{
  "message": "string",
  "errorCode": "string"
}
```

ForbiddenException schema

```
{
  "message": "string",
  "errorCode": "string"
}
```

NotFoundException schema

```
{
  "message": "string",
  "errorCode": "string"
}
```

ConflictException schema

```
{
  "message": "string",
  "errorCode": "string"
}
```

TooManyRequestsException schema

```
{
  "message": "string",
  "errorCode": "string"
}
```

InternalServerErrorException schema

```
{
  "message": "string",
  "errorCode": "string"
}
```

Properties

Application

Details about the application.

applicationId

The application Amazon Resource Name (ARN).

Type: string
Required: True

name

The name of the application.

Minimum length=1. Maximum length=140

Pattern: "[a-zA-Z0-9\\-]+";

Type: string
Required: True

description

The description of the application.

Minimum length=1. Maximum length=256

Type: string
Required: True

author

The name of the author publishing the app.

Minimum length=1. Maximum length=127.

Pattern "^[a-z0-9](([a-z0-9]|-(?!-))*[a-z0-9])?$";

Type: string
Required: True

spdxLicenseId

A valid identifier from https://spdx.org/licenses/.

Type: string
Required: False

licenseUrl

A link to a license file of the app that matches the spdxLicenseID value of your application.

Maximum size 5 MB

Type: string
Required: False

readmeUrl

A link to the readme file in Markdown language that contains a more detailed description of the application and how it works.

Maximum size 5 MB

Type: string
Required: False

labels

Labels to improve discovery of apps in search results.

Minimum length=1. Maximum length=127. Maximum number of labels: 10

Pattern: "^[a-zA-Z0-9+\\-_:\\/@]+$";

Type: Array of type string
Required: False

creationTime

The date and time this resource was created.

Type: string
Required: False

homePageUrl

A URL with more information about the application, for example the location of your GitHub repository for the application.

Type: string
Required: False

version

Version information about the application.

Type: Version (p. 76)
Required: False

ApplicationPage

A list of application details.

applications

An array of application summaries.

Type: Array of type ApplicationSummary (p. 69)
Required: True

nextToken

The token to request the next page of results.

Type: string
Required: False

ApplicationSummary

Summary of details about the application.

applicationId

The application Amazon Resource Name (ARN).

Type: string
Required: True

name

The name of the application.

Minimum length=1. Maximum length=140

Pattern: "[a-zA-Z0-9\\-]+";

Type: string
Required: True

description

The description of the application.

Minimum length=1. Maximum length=256

Type: string
Required: True

author

The name of the author publishing the app.

Minimum length=1. Maximum length=127.

Pattern "^[a-z0-9](([a-z0-9]|-(?!-))*[a-z0-9])?$";

Type: string
Required: True

spdxLicenseId

A valid identifier from https://spdx.org/licenses/.

Type: string
Required: False

labels

Labels to improve discovery of apps in search results.

Minimum length=1. Maximum length=127. Maximum number of labels: 10

Pattern: "^[a-zA-Z0-9+\\-_:\\/@]+$";

Type: Array of type string
Required: False

creationTime

The date and time this resource was created.

Type: string
Required: False

homePageUrl

A URL with more information about the application, for example the location of your GitHub repository for the application.

Type: string
Required: False

BadRequestException

One of the parameters in the request is invalid.

message

One of the parameters in the request is invalid.

Type: string
Required: False

errorCode

400

Type: string
Required: False

ConflictException

The resource already exists.

message

The resource already exists.

Type: string
Required: False

errorCode

409

Type: string
Required: False

CreateApplicationInput

Create an application request.

name

The name of the application that you want to publish.

Minimum length=1. Maximum length=140

Pattern: "[a-zA-Z0-9\\-]+";

Type: string
Required: True

description

The description of the application.

Minimum length=1. Maximum length=256

Type: string
Required: True

author

The name of the author publishing the app.

Minimum length=1. Maximum length=127.

Pattern "^[a-z0-9](([a-z0-9]|-(?!-))*[a-z0-9])?$";

Type: string
Required: True

spdxLicenseId

A valid identifier from https://spdx.org/licenses/.

Type: string
Required: False

licenseBody

A raw text file that contains the license of the app that matches the spdxLicenseID value of your application.

Maximum size 5 MB

Type: string
Required: False

licenseUrl

A link to a license file of the app that matches the spdxLicenseID value of your application.

Maximum size 5 MB

Type: string
Required: False

readmeBody

A text readme file in Markdown language that contains a more detailed description of the application and how it works.

Maximum size 5 MB

Type: string
Required: False

readmeUrl

A link to the readme file in Markdown language that contains a more detailed description of the application and how it works.

Maximum size 5 MB

Type: string
Required: False

labels

Labels to improve discovery of apps in search results.

Minimum length=1. Maximum length=127. Maximum number of labels: 10

Pattern: "^[a-zA-Z0-9+\\-_:\\/@]+$";

Type: Array of type string
Required: False

homePageUrl

A URL with more information about the application, for example the location of your GitHub repository for the application.

Type: string
Required: False

semanticVersion

The semantic version of the application:

https://semver.org/

Type: string
Required: False

templateBody

The raw packaged AWS SAM template of your application.

Type: string
Required: False

templateUrl

A link to the packaged AWS SAM template of your application.

Type: string
Required: False

sourceCodeUrl

A link to a public repository for the source code of your application.

Type: string
Required: False

ForbiddenException

The client is not authenticated.

message

The client is not authenticated.

Type: string
Required: False

errorCode

403

Type: string
Required: False

InternalServerErrorException

The AWS Serverless Application Repository service encountered an internal error.

message

The AWS Serverless Application Repository service encountered an internal error.

Type: string
Required: False

errorCode

500

Type: string
Required: False

NotFoundException

The resource (for example, an access policy statement) specified in the request doesn't exist.

message

The resource (for example, an access policy statement) specified in the request doesn't exist.

Type: string
Required: False

errorCode

404

Type: string
Required: False

ParameterDefinition

Parameters supported by the application.

name

The name of the parameter.

Type: string
Required: True

defaultValue

A value of the appropriate type for the template to use if no value is specified when a stack is created. If you define constraints for the parameter, you must specify a value that adheres to those constraints.

Type: string
Required: False

description

A string of up to 4,000 characters that describes the parameter.

Type: string
Required: False

type

The type of the parameter.

Valid values: `String | Number | List<Number> | CommaDelimitedList`

`String`: A literal string.

For example, users can specify `"MyUserName"`.

`Number`: An integer or float. AWS CloudFormation validates the parameter value as a number. However, when you use the parameter elsewhere in your template (for example, by using the `Ref` intrinsic function), the parameter value becomes a string.

For example, users might specify `"8888"`.

`List<Number>`: An array of integers or floats that are separated by commas. AWS CloudFormation validates the parameter value as numbers. However, when you use the parameter elsewhere in your template (for example, by using the `Ref` intrinsic function), the parameter value becomes a list of strings.

For example, users might specify "80,20", and then `Ref` results in `["80","20"]`.

`CommaDelimitedList`: An array of literal strings that are separated by commas. The total number of strings should be one more than the total number of commas. Also, each member string is space-trimmed.

For example, users might specify "test,dev,prod", and then `Ref` results in `["test","dev","prod"]`.

Type: string
Required: False

noEcho

Whether to mask the parameter value whenever anyone makes a call that describes the stack. If you set the value to true, the parameter value is masked with asterisks (*****).

Type: boolean
Required: False

allowedPattern

A regular expression that represents the patterns to allow for `String` types.

Type: string
Required: False

constraintDescription

A string that explains a constraint when the constraint is violated. For example, without a constraint description, a parameter that has an allowed pattern of `[A-Za-z0-9]+` displays the following error message when the user specifies an invalid value:

```
Malformed input-Parameter MyParameter must match pattern [A-Za-z0-9]+
```

By adding a constraint description, such as "must contain only uppercase and lowercase letters and numbers," you can display the following customized error message:

```
Malformed input-Parameter MyParameter must contain only uppercase and lowercase
letters and numbers.
```

Type: string
Required: False

minValue

A numeric value that determines the smallest numeric value that you want to allow for `Number` types.

Type: integer
Required: False

maxValue

A numeric value that determines the largest numeric value that you want to allow for `Number` types.

Type: integer
Required: False

minLength

An integer value that determines the smallest number of characters that you want to allow for `String` types.

Type: integer
Required: False

maxLength

An integer value that determines the largest number of characters that you want to allow for `String` types.

Type: integer
Required: False

allowedValues

An array containing the list of values allowed for the parameter.

Type: Array of type string
Required: False

referencedByResources

A list of AWS SAM resources that use this parameter.

Type: Array of type string
Required: True

TooManyRequestsException

The client is sending more than the allowed number of requests per unit of time.

message

The client is sending more than the allowed number of requests per unit of time.

Type: string
Required: False

errorCode

429

Type: string
Required: False

Version

Application version details.

applicationId

The application Amazon Resource Name (ARN).

Type: string
Required: True

semanticVersion

The semantic version of the application:

https://semver.org/

Type: string
Required: True

sourceCodeUrl

A link to a public repository for the source code of your application.

Type: string
Required: False

templateUrl

A link to the packaged AWS SAM template of your application.

Type: string
Required: True

creationTime

The date and time this resource was created.

Type: string
Required: True

parameterDefinitions

An array of parameter types supported by the application.

Type: Array of type ParameterDefinition (p. 74)
Required: True

See also

For more information about using this API in one of the language-specific AWS SDKs and references, see the following:

ListApplications

- AWS Command Line Interface
- AWS SDK for .NET

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CreateApplication

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

Applications applicationId

URI

`/applications/applicationId`

HTTP methods

GET

Operation ID: `GetApplication`

Gets the specified application.

Path parameters

Name	Type	Required	Description
applicationId	String	True	The ID of the application to get.

Query parameters

Name	Type	Required	Description
semanticVersion	String	False	The semantic version of the application to get.

Responses

Status code	Response model	Description
200	Application (p. 81)	Success
400	BadRequestException (p. 81)	One of the parameters in the request is invalid.
403	ForbiddenException (p. 82)	The client is not authenticated.
404	NotFoundException (p. 82)	The resource (for example, an access policy statement) specified in the request doesn't exist.
429	TooManyRequestsException (p. 83)	The client is sending more than the allowed number of requests per unit of time.
500	InternalServerErrorException (p. 82)	The AWS Serverless Application Repository service encountered an internal error.

DELETE

Operation ID: `DeleteApplication`

Deletes the specified application.

Path parameters

Name	Type	Required	Description
applicationId	String	True	The ID of the application to get.

Responses

Status code	Response model	Description
204	None	Success
400	BadRequestException (p. 81)	One of the parameters in the request is invalid.
403	ForbiddenException (p. 82)	The client is not authenticated.
404	NotFoundException (p. 82)	The resource (for example, an access policy statement) specified in the request doesn't exist.
409	ConflictException (p. 82)	The resource already exists.
429	TooManyRequestsException (p. 83)	The client is sending more than the allowed number of requests per unit of time.

Status code	Response model	Description
500	InternalServerErrorException (p. 83)	The AWS Serverless Application Repository service encountered an internal error.

PATCH

Operation ID: `UpdateApplication`

Updates the specified application.

Path parameters

Name	Type	Required	Description
applicationId	String	True	The ID of the application to get.

Responses

Status code	Response model	Description
200	Application (p. 81)	Success
400	BadRequestException (p. 81)	One of the parameters in the request is invalid.
403	ForbiddenException (p. 82)	The client is not authenticated.
404	NotFoundException (p. 82)	The resource (for example, an access policy statement) specified in the request doesn't exist.
409	ConflictException (p. 82)	The resource already exists.
429	TooManyRequestsException (p. 83)	The client is sending more than the allowed number of requests per unit of time.
500	InternalServerErrorException (p. 83)	The AWS Serverless Application Repository service encountered an internal error.

Schemas

Request bodies

PATCH schema

```
{
  "description": "string",
  "author": "string",
  "readmeBody": "string",
```

```
  "readmeUrl": "string",
  "labels": [
    "string"
  ],
  "homePageUrl": "string"
}
```

Response bodies

Application schema

```
{
  "applicationId": "string",
  "name": "string",
  "description": "string",
  "author": "string",
  "spdxLicenseId": "string",
  "licenseUrl": "string",
  "readmeUrl": "string",
  "labels": [
    "string"
  ],
  "creationTime": "string",
  "homePageUrl": "string",
  "version": {
    "applicationId": "string",
    "semanticVersion": "string",
    "sourceCodeUrl": "string",
    "templateUrl": "string",
    "creationTime": "string",
    "parameterDefinitions": [
      {
        "name": "string",
        "defaultValue": "string",
        "description": "string",
        "type": "string",
        "noEcho": boolean,
        "allowedPattern": "string",
        "constraintDescription": "string",
        "minValue": integer,
        "maxValue": integer,
        "minLength": integer,
        "maxLength": integer,
        "allowedValues": [
          "string"
        ],
        "referencedByResources": [
          "string"
        ]
      }
    ]
  }
}
```

BadRequestException schema

```
{
  "message": "string",
  "errorCode": "string"
}
```

ForbiddenException schema

```
{
  "message": "string",
  "errorCode": "string"
}
```

NotFoundException schema

```
{
  "message": "string",
  "errorCode": "string"
}
```

ConflictException schema

```
{
  "message": "string",
  "errorCode": "string"
}
```

TooManyRequestsException schema

```
{
  "message": "string",
  "errorCode": "string"
}
```

InternalServerErrorException schema

```
{
  "message": "string",
  "errorCode": "string"
}
```

Properties

Application

Details about the application.

applicationId

The application Amazon Resource Name (ARN).

Type: string
Required: True

name

The name of the application.

Minimum length=1. Maximum length=140

Pattern: "[a-zA-Z0-9\\-]+";

Type: string
Required: True

description

The description of the application.

Minimum length=1. Maximum length=256

Type: string
Required: True

author

The name of the author publishing the app.

Minimum length=1. Maximum length=127.

Pattern "^[a-z0-9](([a-z0-9]|-(?!-))*[a-z0-9])?$";

Type: string
Required: True

spdxLicenseId

A valid identifier from https://spdx.org/licenses/.

Type: string
Required: False

licenseUrl

A link to a license file of the app that matches the spdxLicenseID value of your application.

Maximum size 5 MB

Type: string
Required: False

readmeUrl

A link to the readme file in Markdown language that contains a more detailed description of the application and how it works.

Maximum size 5 MB

Type: string
Required: False

labels

Labels to improve discovery of apps in search results.

Minimum length=1. Maximum length=127. Maximum number of labels: 10

Pattern: "^[a-zA-Z0-9+\\-_:\\/@]+$";

Type: Array of type string
Required: False

creationTime

The date and time this resource was created.

Type: string
Required: False

homePageUrl

A URL with more information about the application, for example the location of your GitHub repository for the application.

Type: string
Required: False

version

Version information about the application.

Type: Version (p. 89)
Required: False

BadRequestException

One of the parameters in the request is invalid.

message

One of the parameters in the request is invalid.

Type: string
Required: False

errorCode

400

Type: string
Required: False

ConflictException

The resource already exists.

message

The resource already exists.

Type: string
Required: False

errorCode

409

Type: string
Required: False

ForbiddenException

The client is not authenticated.

message

The client is not authenticated.

Type: string
Required: False

errorCode

403

Type: string
Required: False

InternalServerErrorException

The AWS Serverless Application Repository service encountered an internal error.

message

The AWS Serverless Application Repository service encountered an internal error.

Type: string
Required: False

errorCode

500

Type: string
Required: False

NotFoundException

The resource (for example, an access policy statement) specified in the request doesn't exist.

message

The resource (for example, an access policy statement) specified in the request doesn't exist.

Type: string
Required: False

errorCode

404

Type: string
Required: False

ParameterDefinition

Parameters supported by the application.

name

The name of the parameter.

Type: string
Required: True

defaultValue

A value of the appropriate type for the template to use if no value is specified when a stack is created. If you define constraints for the parameter, you must specify a value that adheres to those constraints.

Type: string
Required: False

description

A string of up to 4,000 characters that describes the parameter.

Type: string
Required: False

type

The type of the parameter.

Valid values: `String | Number | List<Number> | CommaDelimitedList`

`String`: A literal string.

For example, users can specify `"MyUserName"`.

`Number`: An integer or float. AWS CloudFormation validates the parameter value as a number. However, when you use the parameter elsewhere in your template (for example, by using the `Ref` intrinsic function), the parameter value becomes a string.

For example, users might specify `"8888"`.

`List<Number>`: An array of integers or floats that are separated by commas. AWS CloudFormation validates the parameter value as numbers. However, when you use the parameter elsewhere in your template (for example, by using the `Ref` intrinsic function), the parameter value becomes a list of strings.

For example, users might specify "80,20", and then `Ref` results in `["80","20"]`.

`CommaDelimitedList`: An array of literal strings that are separated by commas. The total number of strings should be one more than the total number of commas. Also, each member string is space-trimmed.

For example, users might specify "test,dev,prod", and then `Ref` results in `["test","dev","prod"]`.

Type: string
Required: False

noEcho

Whether to mask the parameter value whenever anyone makes a call that describes the stack. If you set the value to true, the parameter value is masked with asterisks (*****).

Type: boolean
Required: False

allowedPattern

A regular expression that represents the patterns to allow for `String` types.

Type: string
Required: False

constraintDescription

A string that explains a constraint when the constraint is violated. For example, without a constraint description, a parameter that has an allowed pattern of `[A-Za-z0-9]+` displays the following error message when the user specifies an invalid value:

```
Malformed input-Parameter MyParameter must match pattern [A-Za-z0-9]+
```

By adding a constraint description, such as "must contain only uppercase and lowercase letters and numbers," you can display the following customized error message:

```
Malformed input-Parameter MyParameter must contain only uppercase and lowercase
letters and numbers.
```

Type: string
Required: False

minValue

A numeric value that determines the smallest numeric value that you want to allow for `Number` types.

Type: integer
Required: False

maxValue

A numeric value that determines the largest numeric value that you want to allow for `Number` types.

Type: integer
Required: False

minLength

An integer value that determines the smallest number of characters that you want to allow for `String` types.

Type: integer
Required: False

maxLength

An integer value that determines the largest number of characters that you want to allow for `String` types.

Type: integer
Required: False

allowedValues

An array containing the list of values allowed for the parameter.

Type: Array of type string
Required: False

referencedByResources

A list of AWS SAM resources that use this parameter.

Type: Array of type string
Required: True

TooManyRequestsException

The client is sending more than the allowed number of requests per unit of time.

message

The client is sending more than the allowed number of requests per unit of time.

Type: string
Required: False

errorCode

429

Type: string
Required: False

UpdateApplicationInput

Update the application request.

description

The description of the application.

Minimum length=1. Maximum length=256

Type: string
Required: False

author

The name of the author publishing the app.

Minimum length=1. Maximum length=127.

Pattern "^[a-z0-9](([a-z0-9]|-(?!-))*[a-z0-9])?$";

Type: string
Required: False

readmeBody

A text readme file in Markdown language that contains a more detailed description of the application and how it works.

Maximum size 5 MB

Type: string
Required: False

readmeUrl

A link to the readme file in Markdown language that contains a more detailed description of the application and how it works.

Maximum size 5 MB

Type: string
Required: False

labels

Labels to improve discovery of apps in search results.

Minimum length=1. Maximum length=127. Maximum number of labels: 10

Pattern: "^[a-zA-Z0-9+\\-_:\\/@]+$";

Type: Array of type string
Required: False

homePageUrl

A URL with more information about the application, for example the location of your GitHub repository for the application.

Type: string
Required: False

Version

Application version details.

applicationId

The application Amazon Resource Name (ARN).

Type: string
Required: True

semanticVersion

The semantic version of the application:

https://semver.org/

Type: string
Required: True

sourceCodeUrl

A link to a public repository for the source code of your application.

Type: string
Required: False

templateUrl

A link to the packaged AWS SAM template of your application.

Type: string
Required: True

creationTime

The date and time this resource was created.

Type: string
Required: True

parameterDefinitions

An array of parameter types supported by the application.

Type: Array of type ParameterDefinition (p. 86)
Required: True

See also

For more information about using this API in one of the language-specific AWS SDKs and references, see the following:

GetApplication

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript

- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DeleteApplication

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

UpdateApplication

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

Applications applicationId Changesets

URI

`/applications/applicationId/changesets`

HTTP methods

POST

Operation ID: `CreateCloudFormationChangeSet`

Creates an AWS CloudFormation change set for the given application.

Path parameters

Name	Type	Required	Description
applicationId	String	True	The ID of the application to get.

Responses

Status code	Response model	Description
201	ChangeSetDetails (p. 92)	Success
400	BadRequestException (p. 92)	One of the parameters in the request is invalid.
403	ForbiddenException (p. 93)	The client is not authenticated.
429	TooManyRequestsException (p. 93)	The client is sending more than the allowed number of requests per unit of time.
500	InternalServerErrorException (p. 93)	The AWS Serverless Application Repository service encountered an internal error.

Schemas

Request bodies

POST schema

```
{
  "stackName": "string",
  "semanticVersion": "string",
  "parameterOverrides": [
    {
      "name": "string",
      "value": "string"
    }
  ]
}
```

Response bodies

ChangeSetDetails schema

```
{
  "applicationId": "string",
  "semanticVersion": "string",
  "changeSetId": "string",
  "stackId": "string"
}
```

BadRequestException schema

```
{
  "message": "string",
  "errorCode": "string"
}
```

ForbiddenException schema

```
{
  "message": "string",
  "errorCode": "string"
}
```

TooManyRequestsException schema

```
{
  "message": "string",
  "errorCode": "string"
}
```

InternalServerErrorException schema

```
{
  "message": "string",
  "errorCode": "string"
}
```

Properties

BadRequestException

One of the parameters in the request is invalid.

message

One of the parameters in the request is invalid.

Type: string
Required: False

errorCode

400

Type: string
Required: False

ChangeSetDetails

Details of the change set.

applicationId

The application Amazon Resource Name (ARN).

Type: string
Required: True

semanticVersion

The semantic version of the application:

https://semver.org/

Type: string
Required: True

changeSetId

The Amazon Resource Name (ARN) of the change set.

Length constraints: Minimum length of 1.

Pattern: ARN:[-a-zA-Z0-9:/]*

Type: string
Required: True

stackId

The unique ID of the stack.

Type: string
Required: True

CreateCloudFormationChangeSetInput

Create an application change set request.

stackName

The name or the unique ID of the stack for which you are creating a change set. AWS CloudFormation generates the change set by comparing this stack's information with the information that you submit, such as a modified template or different parameter input values.

Constraints: Minimum length of 1.

Pattern: ([a-zA-Z][-a-zA-Z0-9]*)|(arn:\b(aws|aws-us-gov|aws-cn)\b:[-a-zA-Z0-9:/._+]*)

Type: string
Required: True

semanticVersion

The semantic version of the application:

https://semver.org/

Type: string
Required: False

parameterOverrides

A list of parameter values for the parameters of the application.

Type: Array of type ParameterValue (p. 95)
Required: False

ForbiddenException

The client is not authenticated.

message

The client is not authenticated.

Type: string
Required: False

errorCode

403

Type: string
Required: False

InternalServerErrorException

The AWS Serverless Application Repository service encountered an internal error.

message

The AWS Serverless Application Repository service encountered an internal error.

Type: string
Required: False

errorCode

500

Type: string
Required: False

ParameterValue

Parameter value of the application.

name

The key associated with the parameter. If you don't specify a key and value for a particular parameter, AWS CloudFormation uses the default value that is specified in your template.

Type: string
Required: True

value

The input value associated with the parameter.

Type: string
Required: True

TooManyRequestsException

The client is sending more than the allowed number of requests per unit of time.

message

The client is sending more than the allowed number of requests per unit of time.

Type: string
Required: False

errorCode

429

Type: string
Required: False

See also

For more information about using this API in one of the language-specific AWS SDKs and references, see the following:

CreateCloudFormationChangeSet

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

Applications applicationId Policy

URI

`/applications/applicationId/policy`

HTTP methods

GET

Operation ID: `GetApplicationPolicy`

Gets the policy for the specified application.

Path parameters

Name	Type	Required	Description
applicationId	String	True	The ID of the application to get.

Responses

Status code	Response model	Description
200	ApplicationPolicy (p. 98)	Success
400	BadRequestException (p. 98)	One of the parameters in the request is invalid.
403	ForbiddenException (p. 99)	The client is not authenticated.
404	NotFoundException (p. 99)	The resource (for example, an access policy statement) specified in the request doesn't exist.
429	TooManyRequestsException (p. 99)	The client is sending more than the allowed number of requests per unit of time.
500	InternalServerErrorException (p. 99)	The AWS Serverless Application Repository service encountered an internal error.

PUT

Operation ID: `PutApplicationPolicy`

Puts the policy for the specified application.

Path parameters

Name	Type	Required	Description
applicationId	String	True	The ID of the application to get.

Responses

Status code	Response model	Description
200	ApplicationPolicy (p. 98)	Success
400	BadRequestException (p. 98)	One of the parameters in the request is invalid.
403	ForbiddenException (p. 99)	The client is not authenticated.

Status code	Response model	Description
404	NotFoundException (p. 99)	The resource (for example, an access policy statement) specified in the request doesn't exist.
429	TooManyRequestsException (p. 99)	The client is sending more than the allowed number of requests per unit of time.
500	InternalServerErrorException (p. 99)	The AWS Serverless Application Repository service encountered an internal error.

Schemas

Request bodies

PUT schema

```
{
  "statements": [
    {
      "statementId": "string",
      "principals": [
        "string"
      ],
      "actions": [
        "string"
      ]
    }
  ]
}
```

Response bodies

ApplicationPolicy schema

```
{
  "statements": [
    {
      "statementId": "string",
      "principals": [
        "string"
      ],
      "actions": [
        "string"
      ]
    }
  ]
}
```

BadRequestException schema

```
{
```

```
  "message": "string",
  "errorCode": "string"
}
```

ForbiddenException schema

```
{
  "message": "string",
  "errorCode": "string"
}
```

NotFoundException schema

```
{
  "message": "string",
  "errorCode": "string"
}
```

TooManyRequestsException schema

```
{
  "message": "string",
  "errorCode": "string"
}
```

InternalServerErrorException schema

```
{
  "message": "string",
  "errorCode": "string"
}
```

Properties

ApplicationPolicy

Policy statements applied to the application.

statements

An array of policy statements applied to the application.

Type: Array of type ApplicationPolicyStatement (p. 99)
Required: True

ApplicationPolicyStatement

Policy statement applied to the application.

statementId

A unique ID for the statement.

Type: string
Required: False

principals

An AWS account ID, or * to make the application public.

Type: Array of type string
Required: True

actions

A list of supported actions:

`GetApplication`

`CreateCloudFormationChangeSet`

`ListApplicationVersions`

`SearchApplications`

`Deploy` (Note: This action enables all other actions preceding.)

Type: Array of type string
Required: True

BadRequestException

One of the parameters in the request is invalid.

message

One of the parameters in the request is invalid.

Type: string
Required: False

errorCode

400

Type: string
Required: False

ForbiddenException

The client is not authenticated.

message

The client is not authenticated.

Type: string
Required: False

errorCode

403

Type: string
Required: False

InternalServerErrorException

The AWS Serverless Application Repository service encountered an internal error.

message

The AWS Serverless Application Repository service encountered an internal error.

Type: string
Required: False

errorCode

500

Type: string
Required: False

NotFoundException

The resource (for example, an access policy statement) specified in the request doesn't exist.

message

The resource (for example, an access policy statement) specified in the request doesn't exist.

Type: string
Required: False

errorCode

404

Type: string
Required: False

TooManyRequestsException

The client is sending more than the allowed number of requests per unit of time.

message

The client is sending more than the allowed number of requests per unit of time.

Type: string
Required: False

errorCode

429

Type: string
Required: False

See also

For more information about using this API in one of the language-specific AWS SDKs and references, see the following:

GetApplicationPolicy

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

PutApplicationPolicy

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

Applications applicationId Versions

URI

`/applications/applicationId/versions`

HTTP methods

GET

Operation ID: `ListApplicationVersions`

Lists versions for the specified application.

Path parameters

Name	Type	Required	Description
applicationId	String	True	The ID of the application to get.

Query parameters

Name	Type	Required	Description
maxItems	String	False	The total number of items to return.
nextToken	String	False	A token to specify where to start paginating.

Responses

Status code	Response model	Description
200	ApplicationVersionPage (p. 102)	Success
400	BadRequestException (p. 104)	One of the parameters in the request is invalid.
403	ForbiddenException (p. 104)	The client is not authenticated.
404	NotFoundException (p. 104)	The resource (for example, an access policy statement) specified in the request doesn't exist.
429	TooManyRequestsException (p. 104)	The client is sending more than the allowed number of requests per unit of time.
500	InternalServerErrorException (p. 104)	The AWS Serverless Application Repository service encountered an internal error.

Schemas

Response bodies

ApplicationVersionPage schema

```
{
  "versions": [
    {
      "applicationId": "string",
      "semanticVersion": "string",
      "sourceCodeUrl": "string",
      "creationTime": "string"
```

```
    }
  ],
  "nextToken": "string"
}
```

BadRequestException schema

```
{
  "message": "string",
  "errorCode": "string"
}
```

ForbiddenException schema

```
{
  "message": "string",
  "errorCode": "string"
}
```

NotFoundException schema

```
{
  "message": "string",
  "errorCode": "string"
}
```

TooManyRequestsException schema

```
{
  "message": "string",
  "errorCode": "string"
}
```

InternalServerErrorException schema

```
{
  "message": "string",
  "errorCode": "string"
}
```

Properties

ApplicationVersionPage

A list of version summaries for the application.

versions

An array of version summaries for the application.

Type: Array of type VersionSummary (p. 106)
Required: True

nextToken

The token to request the next page of results.

Type: string
Required: False

BadRequestException

One of the parameters in the request is invalid.

message

One of the parameters in the request is invalid.

Type: string
Required: False

errorCode

400

Type: string
Required: False

ForbiddenException

The client is not authenticated.

message

The client is not authenticated.

Type: string
Required: False

errorCode

403

Type: string
Required: False

InternalServerErrorException

The AWS Serverless Application Repository service encountered an internal error.

message

The AWS Serverless Application Repository service encountered an internal error.

Type: string
Required: False

errorCode

500

Type: string
Required: False

NotFoundException

The resource (for example, an access policy statement) specified in the request doesn't exist.

message

The resource (for example, an access policy statement) specified in the request doesn't exist.

Type: string
Required: False

errorCode

404

Type: string
Required: False

TooManyRequestsException

The client is sending more than the allowed number of requests per unit of time.

message

The client is sending more than the allowed number of requests per unit of time.

Type: string
Required: False

errorCode

429

Type: string
Required: False

VersionSummary

An application version summary.

applicationId

The application Amazon Resource Name (ARN).

Type: string
Required: True

semanticVersion

The semantic version of the application:

https://semver.org/

Type: string
Required: True

sourceCodeUrl

A link to a public repository for the source code of your application.

Type: string
Required: False

creationTime

The date and time this resource was created.

Type: string
Required: True

See also

For more information about using this API in one of the language-specific AWS SDKs and references, see the following:

ListApplicationVersions

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

Applications applicationId Versions semanticVersion

URI

`/applications/applicationId/versions/semanticVersion`

HTTP methods

PUT

Operation ID: `CreateApplicationVersion`

Creates an application version.

Path parameters

Name	Type	Required	Description
applicationId	String	True	The ID of the application to get.
semanticVersion	String	True	The semantic version of the new version.

Responses

Status code	Response model	Description
201	Version (p. 108)	Success
400	BadRequestException (p. 109)	One of the parameters in the request is invalid.
403	ForbiddenException (p. 109)	The client is not authenticated.
409	ConflictException (p. 109)	The resource already exists.
429	TooManyRequestsException (p. 110)	The client is sending more than the allowed number of requests per unit of time.
500	InternalServerErrorException (p. 110)	The AWS Serverless Application Repository service encountered an internal error.

Schemas

Request bodies

PUT schema

```
{
  "templateBody": "string",
  "templateUrl": "string",
  "sourceCodeUrl": "string"
}
```

Response bodies

Version schema

```
{
  "applicationId": "string",
  "semanticVersion": "string",
  "sourceCodeUrl": "string",
  "templateUrl": "string",
  "creationTime": "string",
  "parameterDefinitions": [
    {
```

```
      "name": "string",
      "defaultValue": "string",
      "description": "string",
      "type": "string",
      "noEcho": boolean,
      "allowedPattern": "string",
      "constraintDescription": "string",
      "minValue": integer,
      "maxValue": integer,
      "minLength": integer,
      "maxLength": integer,
      "allowedValues": [
        "string"
      ],
      "referencedByResources": [
        "string"
      ]
    }
  ]
}
```

BadRequestException schema

```
{
  "message": "string",
  "errorCode": "string"
}
```

ForbiddenException schema

```
{
  "message": "string",
  "errorCode": "string"
}
```

ConflictException schema

```
{
  "message": "string",
  "errorCode": "string"
}
```

TooManyRequestsException schema

```
{
  "message": "string",
  "errorCode": "string"
}
```

InternalServerErrorException schema

```
{
  "message": "string",
  "errorCode": "string"
}
```

Properties

BadRequestException

One of the parameters in the request is invalid.

message

One of the parameters in the request is invalid.

Type: string
Required: False

errorCode

400

Type: string
Required: False

ConflictException

The resource already exists.

message

The resource already exists.

Type: string
Required: False

errorCode

409

Type: string
Required: False

CreateApplicationVersionInput

Create a version request.

templateBody

The raw packaged AWS SAM template of your application.

Type: string
Required: False

templateUrl

A link to the packaged AWS SAM template of your application.

Type: string

Required: False

sourceCodeUrl

A link to a public repository for the source code of your application.

Type: string
Required: False

ForbiddenException

The client is not authenticated.

message

The client is not authenticated.

Type: string
Required: False

errorCode

403

Type: string
Required: False

InternalServerErrorException

The AWS Serverless Application Repository service encountered an internal error.

message

The AWS Serverless Application Repository service encountered an internal error.

Type: string
Required: False

errorCode

500

Type: string
Required: False

ParameterDefinition

Parameters supported by the application.

name

The name of the parameter.

Type: string

Required: True

defaultValue

A value of the appropriate type for the template to use if no value is specified when a stack is created. If you define constraints for the parameter, you must specify a value that adheres to those constraints.

Type: string
Required: False

description

A string of up to 4,000 characters that describes the parameter.

Type: string
Required: False

type

The type of the parameter.

Valid values: `String | Number | List<Number> | CommaDelimitedList`

`String`: A literal string.

For example, users can specify `"MyUserName"`.

`Number`: An integer or float. AWS CloudFormation validates the parameter value as a number. However, when you use the parameter elsewhere in your template (for example, by using the `Ref` intrinsic function), the parameter value becomes a string.

For example, users might specify `"8888"`.

`List<Number>`: An array of integers or floats that are separated by commas. AWS CloudFormation validates the parameter value as numbers. However, when you use the parameter elsewhere in your template (for example, by using the `Ref` intrinsic function), the parameter value becomes a list of strings.

For example, users might specify "80,20", and then `Ref` results in `["80","20"]`.

`CommaDelimitedList`: An array of literal strings that are separated by commas. The total number of strings should be one more than the total number of commas. Also, each member string is space-trimmed.

For example, users might specify "test,dev,prod", and then `Ref` results in `["test","dev","prod"]`.

Type: string
Required: False

noEcho

Whether to mask the parameter value whenever anyone makes a call that describes the stack. If you set the value to true, the parameter value is masked with asterisks (*****).

Type: boolean
Required: False

allowedPattern

A regular expression that represents the patterns to allow for `String` types.

Type: string
Required: False

constraintDescription

A string that explains a constraint when the constraint is violated. For example, without a constraint description, a parameter that has an allowed pattern of `[A-Za-z0-9]+` displays the following error message when the user specifies an invalid value:

```
Malformed input-Parameter MyParameter must match pattern [A-Za-z0-9]+
```

By adding a constraint description, such as "must contain only uppercase and lowercase letters and numbers," you can display the following customized error message:

```
Malformed input-Parameter MyParameter must contain only uppercase and lowercase
letters and numbers.
```

Type: string
Required: False

minValue

A numeric value that determines the smallest numeric value that you want to allow for `Number` types.

Type: integer
Required: False

maxValue

A numeric value that determines the largest numeric value that you want to allow for `Number` types.

Type: integer
Required: False

minLength

An integer value that determines the smallest number of characters that you want to allow for `String` types.

Type: integer
Required: False

maxLength

An integer value that determines the largest number of characters that you want to allow for `String` types.

Type: integer
Required: False

allowedValues

An array containing the list of values allowed for the parameter.

Type: Array of type string
Required: False

referencedByResources

A list of AWS SAM resources that use this parameter.

Type: Array of type string
Required: True

TooManyRequestsException

The client is sending more than the allowed number of requests per unit of time.

message

The client is sending more than the allowed number of requests per unit of time.

Type: string
Required: False

errorCode

429

Type: string
Required: False

Version

Application version details.

applicationId

The application Amazon Resource Name (ARN).

Type: string
Required: True

semanticVersion

The semantic version of the application:

https://semver.org/

Type: string
Required: True

sourceCodeUrl

A link to a public repository for the source code of your application.

Type: string
Required: False

templateUrl

A link to the packaged AWS SAM template of your application.

Type: string
Required: True

creationTime

The date and time this resource was created.

Type: string
Required: True

parameterDefinitions

An array of parameter types supported by the application.

Type: Array of type ParameterDefinition (p. 111)
Required: True

See also

For more information about using this API in one of the language-specific AWS SDKs and references, see the following:

CreateApplicationVersion

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

Historique du document

- Version de l'API : dernière en date
- Mise à jour de la documentation : 10 mars 2020

Le tableau suivante décrit les modifications importantes apportées à chaque édition du Manuel du développeur AWS Serverless Application Repository. Pour recevoir les notifications des mises à jour de cette documentation, abonnez-vous à un flux RSS.

update-history-change	update-history-description	update-history-date
Mises à jour du partage et de la restriction de l'accès aux applications (p. 116)	Ajout de la prise en charge du partage d'applications sur des comptes dansAWSOrganisation, et de restreindre l'accès aux applications publiques pourAWSComptes etAWSOrganizations. Pour plus d'exemples de partage d'applications avec des utilisateurs d'une organisation, consultez Exemples de stratégies AWS Serverless Application Repository basées sur les ressources. Pour obtenir des exemples de restriction de l'accès aux applications publiques, consultez Exemples de stratégies AWS Serverless Application Repository basées sur l'identité.	March 10, 2020
Nouvelles ressources prises en charge (p. 116)	Ajout de la prise en charge d'un certain nombre de ressources supplémentaires. Pour obtenir la liste complète des ressources prises en charge, consultez.Liste des prises en chargeAWSRessources.	January 17, 2020
Régions de Chine (p. 116)	AWS Serverless Application Repository est désormais disponible dans la région Régions de Chine, Pékin et Ningxia. Pour de plus amples information sur les régions et points de terminaison AWS Serverless Application Repository, veuillez consulter Régions et points de terminaison dans le AWS General Reference.	January 15, 2020
Mise à jour de la section Sécurité pour plus de cohérence avec les autres services AWS. (p. 116)	Pour de plus amples informations, veuillez consulter Sécurité.	January 2, 2020

Processus simplifié de publication des applications (p. 116)	La nouvelle commande `sam publish` de l'interface de ligne de commande AWS SAM simplifie le processus de publication des applications sans serveur dans le AWS Serverless Application Repository. Pour obtenir un didacticiel de bout en bout sur le téléchargement et la publication d'un exemple d'application, veuillez consulter.Quick Start : Publication des applications. Pour obtenir des instructions sur la publication d'une application que vous avez déjà développée et testée dans l'AWSCloud, voirPublication d'une application via l'AWS SAMCLI.	December 21, 2018
Prise en charge des applications imbriquées et des couches (p. 116)	Ajout de la prise en charge des applications imbriquées et des couches. Cela inclut les mises à jour deprises en chargeAWSRessourcesandConfirmation des capacités d'une application.	November 29, 2018
Publication d'applications avec des rôles IAM et des stratégies de ressources personnalisés (p. 116)	Ajout de la prise en charge de la publication d'applications avec des rôles IAM et des stratégies de ressources personnalisés. Cela inclut les mises à jour de laUtilisation des applicationsandPublication des applicationsWorkflows et mises à jour deprises en chargeAWSRessourcesandAPI Referencedans leManuel du développeur AWS Serverless Application Repository.	November 16, 2018
Mises à jour des modèles de stratégie (p. 116)	Mises à jour des modèles de stratégie pris en charge dans le Manuel du développeur AWS Serverless Application Repository.	September 26, 2018
Mises à jour de la documentation (p. 116)	Ajout de la rubrique Authentification et contrôle d'accès dans Manuel du développeur AWS Serverless Application Repository.	July 2, 2018

Publication (p. 116)	Publication duAWS Serverless Application Repository, désormais disponible dans 14AWSRégions. Pour en savoir plus sur l'AWSRégions où l'AWS Serverless Application Repositoryest disponible etAWS Serverless Application RepositoryPoints de terminaison, consultez.Régions et points de terminaisondans leAWS General Reference.	February 20, 2018
Nouveau guide (p. 116)	Il s'agit de la première version préliminaire du Manuel du développeur AWS Serverless Application Repository.	November 30, 2017

Glossaire AWS

Pour la terminologie AWS la plus récente, consultez le Glossaire AWS dans le document AWS General Reference.

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.