aws

開発者ガイド

AWS Serverless Application Repository

AWS Serverless Application Repository: 開発者ガイド

Table of Contents

AWS Serverless Application Repository とは

AWS Serverless Application Repository により、デベロッパーや企業は AWS クラウドでサーバーレスアプリケーションをすばやく見つけてデプロイおよび公開することが簡単になります。サーバーレスアプリケーションの詳細については、AWS ウェブサイトのサーバーレスコンピューティングとアプリケーションを参照してください。

アプリケーションの発行、コミュニティ全体との公開共有、またはチーム内や組織内での非公開共有を簡単に行うことができます。サーバーレスアプリケーション (またはアプリ) を発行するには、AWS Management Console、AWS SAM コマンドラインインターフェイス (AWS SAM CLI)、または AWS SDK を使用してコードをアップロードします。コードと共に、AWS Serverless Application Model (AWS SAM) テンプレートとして知られる、単純なマニフェストファイルもアップロードする必要があります。AWS SAM の詳細については、AWS Serverless Application Model デベロッパーガイドを参照してください。

AWS Serverless Application Repository は、AWS Lambda コンソールと緊密に統合されています。この統合のため、あらゆるレベルの開発者が新しいことを学ぶことなくサーバーレスコンピューティングを開始できます。カテゴリキーワードを使用して、ウェブおよびモバイルバックエンド、データ処理アプリケーション、またはチャットボットなどのアプリケーションを参照できます。名前、発行者、またはイベントソースでアプリケーションを検索することもできます。アプリケーションを使用するには、それを選択し、必須フィールドを設定して数回クリックしてデプロイするだけです。

このガイドでは、AWS Serverless Application Repository を使用する 2 つの方法について説明します。

- アプリケーションの公開 - アプリケーションを設定してアップロードし、他のデベロッパーが利用できるようにして、新しいバージョンのアプリケーションを発行します。
- アプリケーションをデプロイする - アプリケーションを参照し、アプリケーションのソースコードや readme ファイルなどの関連情報を表示します。また、選択したアプリケーションをインストール、設定、およびデプロイします。

次のステップ

- AWS Serverless Application Repository へのサンプルアプリケーションの公開に関するチュートリアルについては、クイックスタート: アプリケーションの発行を参照してください。
- AWS Serverless Application Repository からアプリケーションをデプロイする手順については、アプリケーションをデプロイする方法を参照してください。

クイックスタート: アプリケーションの発行

このガイドでは、AWS SAM CLI を使用して、サーバレスアプリケーションのサンプルをダウンロード、構築、テストして、AWS Serverless Application Repository に発行する手順を示します。このサンプルアプリケーションを参考にして、独自のサーバーレスアプリケーションを開発して発行できます。

概要

次の手順は、サーバーレスアプリケーションのサンプルをダウンロード、構築、および発行する方法の概要です。

1. 初期化する。`sam init` を使用してテンプレートからサンプルアプリケーションをダウンロードします。
2. ローカルでテストする。`sam local invoke` または `sam local start-api` を使用して、アプリケーションをローカルでテストします。これらのコマンドでは、Lambda 関数をローカルに呼び出しますが、読み取りと書き込みは AWS クラウド内の AWS リソースに対して行われます。
3. パッケージ化する。Lambda 関数に問題がなければ、`sam package` を使用して Lambda 関数、AWS SAM テンプレート、および依存関係を AWS CloudFormation デプロイパッケージとしてバンドルします。このステップでは、AWS Serverless Application Repository にアップロードするアプリケーションに関する情報も含めます。
4. 発行する。`sam publish` を使用してアプリケーションを AWS Serverless Application Repository に発行します。このステップの最後で、AWS Serverless Application Repository でアプリケーションを表示し、AWS Serverless Application Repository を使用してアプリケーションを AWS クラウドにデプロイできます。

次のセクションの例 Hello World アプリケーション では、サーバーレスアプリケーションの構築と発行の手順を示します。

Hello World アプリケーション

この演習では、単純な API バックエンドを示す Hello World サーバーレスアプリケーションをダウンロードしてテストします。これには、GET オペレーションと Lambda 関数をサポートする Amazon API Gateway エンドポイントが含まれています。エンドポイントに GET リクエストを送信する

と、API Gateway により Lambda 関数が呼び出されます。次に、AWS Lambda は関数を実行します。この関数は、単に `hello world` メッセージを返します。

アプリケーションには次のコンポーネントがあります。

- Hello World アプリケーションの 2AWS SAMAWS つのリソースを定義するテンプレート。1 つは GET オペレーションを使用する API Gateway サービスで、もう 1 つは Lambda 関数です。このテンプレートは、API Gateway GET オペレーションと Lambda 関数の間のマッピングも定義します。
- Python で書かれたアプリケーションコード。

開始する前に

この演習に必要な設定が整っていることを確認します。

- 管理者権限を持つ IAM ユーザーが含まれている有効な AWS アカウントが必要です。AWS アカウントのセットアップを参照してください。
- AWS SAM CLI (コマンドラインインターフェイス) をインストールしている必要があります。「AWS SAM CLI のインストール」を参照してください。
- バージョン 1.16.77 以降の AWS CLI をインストールしている必要があります。「AWS Command Line Interface のインストール」を参照してください。

ステップ 1: アプリケーションの初期化

このセクションでは、AWS SAM テンプレートとアプリケーションコードで構成されるサンプルアプリケーションをダウンロードします。

アプリケーションを初期化する

1. AWS SAM CLI コマンドプロンプトで以下のコマンドを実行します。

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

2. コマンドで作成されたディレクトリの内容を確認します (`sam-app/`)。

   - `template.yaml` – Hello World アプリケーションが必要とする 2 つの AWS リソース (Lambda 関数、および GET オペレーションをサポートする API Gateway エンドポイント) を定義します。このテンプレートは、2 つのリソース間のマッピングも定義します。

- Hello World アプリケーションコードに関連するコンテンツ:
  - `hello_world/` ディレクトリ - アプリケーションコードが含まれています。このコードを実行すると、`hello world` が返されます。

> **Note**
>
> この演習では、アプリケーションコードを Python で記述し、`init` コマンドでランタイムを指定します。AWS Lambda は、アプリケーションコードを作成するための追加の言語をサポートしています。別のサポートされているランタイムを指定すると、`init` コマンドは、指定した言語での Hello World コードと、その言語で参照できる `README.md` ファイルを提供します。サポートされているランタイムの詳細については、Lambda 実行環境と使用可能なライブラリを参照してください。

ステップ 2: アプリケーションのローカルテスト

これで、ローカルマシンに AWS SAM アプリケーションが準備されました。次に、以下の手順に従ってアプリケーションをローカルでテストします。

アプリケーションをローカルでテストするには

1. API ゲートウェイエンドポイントをローカルで起動します。`template.yaml` ファイルがあるディレクトリから次のコマンドを実行する必要があります。

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

   このコマンドは API Gateway エンドポイントを返します。このエンドポイントにローカルテストのためのリクエストを送信できます。
2. アプリケーションをテストします。API Gateway エンドポイント URL をコピーしてブラウザに貼り付け、Enter を選択します。API Gateway エンドポイント URL の一例は、`http://127.0.0.1:3000/hello` です。

   API Gateway は、エンドポイントのマッピング先の Lambda 関数をローカルに呼び出します。Lambda 関数は、ローカルの Docker コンテナで実行され、`hello world` を返します。API Gateway は、テキストが含まれているブラウザへのレスポンスを返します。

演習: メッセージの文字列を変更する

サンプルアプリケーションを正常にテストしたら、簡単な変更を試すために、返されたメッセージの文字列を変更します。

1. /hello_world/app.py ファイルを編集して、メッセージの文字列を 'hello world' から 'Hello World!' に変更します。
2. ブラウザでテスト URL をリロードし、新しい文字列を確認します。

sam local プロセスを再起動することなく、新しいコードが動的にロードされることがわかります。

ステップ 3: アプリケーションのパッケージ化

アプリケーションをローカルでテストしたら、AWS SAM CLI を使用してデプロイパッケージとパッケージ化された AWS SAM テンプレートを作成します。

> **Note**
>
> 次の手順では、アプリケーションコードを含む hello_world/ ディレクトリの内容の .zip ファイルを作成します。この .zip ファイルは、サーバーレスアプリケーションのデプロイパッケージです。詳細については、AWS Lambda デベロッパーガイドのデプロイパッケージの作成 (Python)を参照してください。

Lambda デプロイパッケージを作成する

1. Metadata セクションを AWS SAM テンプレートファイルに追加し、必要なアプリケーション情報を指定します。AWS SAM テンプレートの Metadata セクションの詳細については、AWS Serverless Application Model デベロッパーガイドの AWS SAM テンプレートのメタデータセクションのプロパティを参照してください。

   次に、Metadata セクションの例を示します。

   ```
   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
```

LicenseUrl プロパティと ReadmeUrl プロパティは、ローカルファイルへの参照 (上の例を参照) であるか、これらのアーティファクトをすでにホストしている Amazon S3 バケットへのリンクとなります。

2. パッケージ化されたコードを保存する場所に S3 バケットを作成します。既存の S3 バケットを使用する場合は、このステップをスキップします。

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

3. 次の package AWS SAM CLI コマンドを実行して、Lambda 関数のデプロイパッケージを作成します。

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

コマンドが以下の操作を行います。

- aws-sam/hello_world/ ディレクトリの内容を圧縮して Amazon S3 にアップロードします。
- デプロイパッケージ、README ファイル、および LICENSE ファイルを --s3-bucket オプションで指定した Amazon S3 バケットにアップロードします。
- 新しいテンプレートファイル (packaged.yaml) を出力します。このファイルは、次のステップでアプリケーションを AWS Serverless Application Repository に発行するために使用します。packaged.yaml テンプレートファイルは元のテンプレートファイル (template.yaml) と似ていますが、大きな違いがあります。それは、CodeUri、LicenseUrl、および ReadmeUrl プロパティはそれぞれのアーティファクトを含む Amazon S3 バケットとオブジェクトを指すことです。packaged.yaml テンプレートファイルの例から次のスニペットは、CodeUri プロパティを示しています。

```
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/fbd77a3647a4f47a352fc0bjectGUID

...
```

ステップ 4: アプリケーションを発行する

デプロイパッケージを作成したので、これを使用してアプリケーションを AWS Serverless Application Repository に発行します。

サーバーレスアプリケーションを AWS Serverless Application Repository に発行するには

- 次のコマンドを実行して、AWS Serverless Application Repository で新しいアプリケーションを発行します。最初に作成するバージョンは 0.0.1 とします。

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

> **Note**
>
> アプリケーションは、デフォルトで非公開として作成されます。アプリケーションの表示とデプロイを他の AWS アカウントに許可する前に、アプリケーションを共有する必要があります。アプリケーションの共有の詳細については、次のステップ を参照してください。

次のステップ

サンプルアプリケーションを発行したので、次にこれを使用していくつかの操作を行います。

- AWS Serverless Application Repository でアプリケーションを表示する – `sam publish` コマンドの出力には、アプリケーションの詳細ページを直接表示するための AWS Serverless Application

Repository へのリンクが含まれています。AWS Serverless Application Repository ランディングページに移動して、アプリケーションを検索することもできます。

- アプリケーションを共有する – アプリケーションはデフォルトで非公開に設定されるため、他の AWS アカウントでは表示できません。アプリケーションを他のユーザーと共有するには、アプリケーションを公開とするか、一連の特定の AWS アカウントにアクセス許可を付与する必要があります。AWS CLI を使用してアプリケーションを共有する方法については、「AWS Serverless Application Repositoryアプリケーションポリシーの例」を参照してください。コンソールを使用してアプリケーションを共有する方法については、「アプリケーションの共有」を参照してください。

詳細情報

AWS SAM テンプレートの `Metadata` セクション、AWS SAM CLI の `sam package` および `sam publish` コマンドの詳細については、AWS Serverless Application Model デベロッパーガイドの AWS SAM CLI を使用したアプリケーションの発行を参照してください。

アプリケーションの公開

サーバレスアプリケーションを AWS Serverless Application Repository に発行すると、他のユーザーはそのアプリケーションを見つけてデプロイできます。

最初に AWS Serverless Application Model (AWS SAM) テンプレートを使用してアプリケーションを定義します。アプリケーションを定義するときは、アプリケーションのコンシューマーがアプリケーションの機能を承認する必要があるかどうかを考慮します。AWS SAM と承認機能を使用する方法の詳細については、「AWS SAM を AWS Serverless Application Repository に使用する」を参照してください。

サーバーレスアプリケーションを発行するには、AWS Management Console、AWS SAM コマンドラインインターフェイス (AWS SAM CLI)、または AWS SDK を使用できます。アプリケーションを AWS Serverless Application Repository に発行する手順の詳細については、「アプリケーションを発行する方法」を参照してください。

アプリケーションを発行すると、最初は非公開に設定されます。この場合、アプリケーションはその作成元の AWS アカウントでのみ使用できます。アプリケーションを他のユーザーと共有するには、アプリケーションを非公開共有 (一連の特定の AWS アカウントとのみ共有) に設定するか、公開共有 (すべてのユーザーと共有) に設定する必要があります。

アプリケーションを AWS Serverless Application Repository に発行して公開として設定すると、アプリケーションはすべてのリージョンのコンシューマーが使用できます。コンシューマーが、アプリケーションを最初に発行したリージョン以外のリージョンに公開アプリケーションをデプロイすると、AWS Serverless Application Repository はアプリケーションのデプロイアーティファクトをデプロイ先リージョンの Amazon S3 バケットにコピーします。これらのアーティファクトを使用する AWS SAM テンプレート内のリソースは、デプロイ先リージョンの Amazon S3 バケット内のファイルを参照するように更新されます。デプロイアーティファクトには、Lambda 関数コード、API 定義ファイルなどを含めることができます。

> **Note**
>
> 非公開アプリケーションおよび非公開共有アプリケーションは、作成元の AWS リージョンでのみ使用できます。公開共有アプリケーションは、すべての AWS リージョンで使用できます。アプリケーションの共有の詳細については、「AWS Serverless Application Repository アプリケーションポリシーの例」を参照してください。

トピック

AWS SAM を AWS Serverless Application Repository に使用する

AWS Serverless Application Model (AWS SAM) は、AWS でサーバーレスアプリケーションを構築するために使用できるオープンソースのフレームワークです。サーバーレスアプリケーションを構築するために AWS SAM を使用する方法の詳細については、『AWS Serverless Application Model 開発者ガイド』を参照してください。

AWS Serverless Application Repository に発行するアプリケーションを構築する場合、サポートされている AWS リソースおよびポリシーテンプレートのセットの使用を考慮する必要があります。以下のセクションでは、これらのトピックについて詳しく説明します。

AWS Serverless Application Repository でサポートされている AWS リソース

AWS Serverless Application Repository は、AWS SAM および AWS CloudFormation の多くのリソースで構成されているサーバレスアプリケーションをサポートしています。AWS Serverless Application Repository でサポートされている AWS リソースの詳細なリストについては、サポートされる AWS リソースのリストを参照してください。

追加の AWS リソースのサポートをリクエストする場合は、AWS サポートにお問い合わせください。

> **Important**
>
> アプリケーションテンプレートに、次のいずれかのカスタム IAM ロールやリソースポリシーが含まれている場合、デフォルトではアプリケーションが検索結果に表示されません。また、お客様は、アプリケーションをデプロイする前に、アプリケーションのカスタム IAM ロールまたはリソースポリシーを承認する必要があります。詳細については、「アプリケーション機能の承認」を参照してください。
>
> これが適用されるリソースのリストは次のとおりです。

- IAM ロール:AWS::IAM::Group,AWS:: IAM:: IAM::InstanceProfile,AWS::IAM::Policy, および AWS::IAM::Role。
- リソースポリシー AWS:: Lambda::LayerVersionアクセス権限,AWS::Lambda::Permission,AWS:: Event:: EventEventBusポリシー,AWS:: IAM: ポリシー,AWS::ApplicationAutoスケール変更::ScalingPolicy,AWS:: S3:: S3::BucketPolicy,AWS:: SQS:: SQS::QueuePolicy, および AWS:: SNS:TopicPolicy。

アプリケーションに AWS::Serverless::Application リソースが含まれている場合は、アプリケーションをデプロイする前に、アプリケーションにネストされたアプリケーションが含まれていることを承認する必要があります。ネストされたアプリケーションの詳細については、AWS Serverless Application Model デベロッパーガイドのネストされたアプリケーションを参照してください。機能の承認の詳細については、「アプリケーション機能の承認」を参照してください。

ポリシーテンプレート

AWS SAM には、アプリケーションで使用するリソースに対する Lambda 関数のアクセス許可を絞り込むためのポリシーテンプレートのリストが用意されています。ポリシーテンプレートを使用すると、アプリケーションを検索、参照、またはデプロイするための追加の承認は不要です。

標準の AWS SAM ポリシーテンプレートのリストについては、AWS Serverless Application Modelデベロッパーガイドの AWS SAM ポリシーテンプレートを参照してください。

サポートされる AWS リソースのリスト

これは、AWS Serverless Application Repository でサポートされる AWS リソースの詳細なリストです。

- `AWS::AccessAnalyzer::Analyzer`
- `AWS::AmazonMQ::Broker`
- `AWS::AmazonMQ::Configuration`
- `AWS::AmazonMQ::ConfigurationAssociation`
- `AWS::ApiGateway::Account`
- `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::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::AutoScalingGroup
- AWS::ApplicationAutoScaling::LaunchConfiguration
- AWS::ApplicationAutoScaling::ScalableTarget
- AWS::ApplicationAutoScaling::ScalingPolicy
- AWS::Athena::NamedQuery
- AWS::Athena::WorkGroup
- AWS::CertificateManager::Certificate
- AWS::Chatbot::SlackChannelConfiguration
- AWS::CloudFormation::CustomResource
- AWS::CloudFormation::Interface
- AWS::CloudFormation::Macro
- AWS::CloudFormation::WaitConditionHandle
- AWS::CloudFront::CachePolicy
- AWS::CloudFront::CloudFrontOriginAccessIdentity
- AWS::CloudFront::Distribution
- AWS::CloudFront::Function
- AWS::CloudFront::OriginRequestPolicy
- AWS::CloudFront::ResponseHeadersPolicy
- 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::CodeStar::GitHubRepository
- AWS::CodeStarNotifications::NotificationRule
- 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::EC2::EIP
- AWS::EC2::InternetGateway
- AWS::EC2::NatGateway
- AWS::EC2::Route

- AWS::EC2::RouteTable
- AWS::EC2::SecurityGroup
- AWS::EC2::SecurityGroupEgress
- AWS::EC2::SecurityGroupIngress
- AWS::EC2::Subnet
- AWS::EC2::SubnetRouteTableAssociation
- AWS::EC2::VPC
- AWS::EC2::VPCGatewayAttachment
- AWS::EC2::VPCPeeringConnection
- AWS::ECR::Repository
- AWS::Elasticsearch::Domain
- AWS::Events::EventBus
- 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::SecurityConfiguration
- AWS::Glue::Table
- AWS::Glue::Trigger
- AWS::Glue::Workflow

- AWS::IAM::Group
- AWS::IAM::InstanceProfile
- AWS::IAM::ManagedPolicy
- AWS::IAM::OIDCProvider
- AWS::IAM::Policy
- AWS::IAM::Role
- AWS::IAM::ServiceLinkedRole
- 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::Location::GeofenceCollection
- AWS::Location::Map
- AWS::Location::PlaceIndex
- AWS::Location::RouteCalculator
- AWS::Location::Tracker
- AWS::Location::TrackerConsumer
- 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::SecretsManager::ResourcePolicy
- AWS::SecretsManager::RotationSchedule
- AWS::SecretsManager::Secret
- AWS::SecretsManager::SecretTargetAttachment
- AWS::Serverless::Api
- AWS::Serverless::Application
- AWS::Serverless::Function
- AWS::Serverless::HttpApi
- AWS::Serverless::LayerVersion
- AWS::Serverless::SimpleTable
- AWS::Serverless::StateMachine
- AWS::ServiceDiscovery::HttpNamespace
- AWS::ServiceCatalog::CloudFormationProvisionedProduct
- AWS::ServiceDiscovery::Instance
- AWS::ServiceDiscovery::PrivateDnsNamespace
- AWS::ServiceDiscovery::PublicDnsNamespace
- AWS::ServiceDiscovery::Service
- AWS::SES::ReceiptRule
- AWS::SES::ReceiptRuleSet
- AWS::StepFunctions::Activity
- AWS::StepFunctions::StateMachine
- AWS::Wisdom::Assistant
- AWS::Wisdom::AssistantAssociation
- AWS::Wisdom::KnowledgeBase

アプリケーションを発行する方法

このセクションでは、AWS SAM CLI または AWS Management Console を使用してサーバレスアプリケーションを AWS Serverless Application Repository に発行する手順について説明しま

す。また、アプリケーションを共有して他のユーザーがデプロイできるようにする方法と、AWS Serverless Application Repository からアプリケーションを削除する方法についても説明します。

> **Important**
>
> アプリケーションの発行時に入力した情報は暗号化されません。この情報には、作成者名などのデータが含まれます。個人を特定できる情報を保存または公開しない場合は、アプリケーションの発行時に、この情報を入力しないことをお勧めします。

アプリケーションの発行 (AWS CLI)

アプリケーションを AWS Serverless Application Repository に発行する最も簡単な方法は、一連の AWS SAM CLI コマンドを使用することです。詳細については、AWS Serverless Application Model (AWS SAM) デベロッパーガイドの AWS SAM CLI を使用したアプリケーションの発行を参照してください。

新しいアプリケーションの発行 (コンソール)

このセクションでは、AWS Management Console を使用して新しいアプリケーションを AWS Serverless Application Repository に発行する方法について説明します。既存のアプリケーションの新しいバージョンを発行する手順については、「既存アプリケーションの新しいバージョンの発行」を参照してください。

前提条件

AWS Serverless Application Repository にアプリケーションを発行する前に、以下のものが必要です。

- 有効な AWS アカウント。
- 使用する AWS リソースを定義する有効な AWS Serverless Application Model (AWS SAM) テンプレート。AWS SAM テンプレートの詳細については、「AWS SAM テンプレートの基礎」を参照してください。
- AWS CLI の AWS CloudFormation `package` コマンドを使用して作成したアプリケーション用のパッケージ。このコマンドは、お客様の AWS SAM テンプレートが参照するローカルアーティファクト (ローカルパス) をパッケージ化します。詳細については、ドキュメントの「パッケージ AWS CloudFormation」を参照してください。

- アプリケーションのソースコードを指す URL (アプリケーションを公開で発行する場合)。
- readme.txt ファイル。このファイルには、お客様がアプリケーションを使用する方法、およびお客様独自の AWS アカウントでアプリケーションをデプロイする前の設定方法が説明されている必要があります。
- SPDX ウェブサイトの license.txt ファイルまたは有効なライセンス識別子。ライセンスは、アプリケーションを公開共有する場合にのみ必要です。アプリケーションを非公開にしたり、非公開共有したりする場合は、ライセンスを指定する必要はありません。
- アプリケーションをパッケージ化したときに Amazon S3 にアップロードしたアーティファクトに対する読み取りアクセス許可をサービスに付与するための有効な Amazon S3 バケットポリシー。このポリシーを設定するには、次の手順に従います。

  1. https://console.aws.amazon.com/s3/ で Simple Storage Service (Amazon S3) コンソールを開きます。
  2. アプリケーションのパッケージ化に使用した Amazon S3 バケットを選択します。
  3. [Permissions] (アクセス許可) タブを選択します。
  4. [バケットポリシー] ボタンを選択します。
  5. 次のポリシーステートメントを [バケットポリシーエディタ] に貼り付けます。`Resource` エレメントのバケット名、および `Condition` エレメントの AWS アカウント ID を必ず置き換えてください。`Condition` エレメントの式は、AWS Serverless Application Repository が、指定された AWS アカウントからのアプリケーションのみにアクセスする許可を持っていることを保証します。ポリシーステートメントの詳細については、IAM ユーザーガイドの「IAM JSON ポリシーの要素のリファレンス」を参照してください。

```
{
    "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. [保存] ボタンを選択します。

手順

AWS Serverless Application Repository でアプリケーションを新規作成するには、次の手順を使用します。

AWS Serverless Application Repository で新しいアプリケーションを作成するには

1. AWS Serverless Application Repository コンソールを開き、[Publish applications] を選択します。
2. [Publish an application] ページで、次のアプリケーション情報を入力し、[Publish application] を選択します。

プロパティ	必要	説明
アプリケーション名	TRUE	アプリケーションの名前。 最小長 = 1。最大長 = 140。 パターン: "[a-zA-Z0-9\\-]+";
筆者	TRUE	アプリケーションを公開する作成者の名前。 最小長 = 1。最大長 = 127。 パターン: "^[a-z0-9](([a-z0-9]|-(?!-))*[a-z0-9])?$";
ホームページ	FALSE	アプリケーションに関する詳細情報が含まれる URL。例えば、 GitHub アプリケーションのリポジトリ。
説明	TRUE	アプリケーションの説明。

プロパティ	必要	説明
		最小長 = 1。最大長 = 256。
ラベル	FALSE	検索結果でアプリケーションを見つけやすくするためのラベル。 最小長 = 1。最大長 = 127。ラベルの最大数: 10. パターン: "^[a-zA-Z0-9+\\-_:\\/@]+$";
Spdx ライセンス (ドロップダウンリスト)	FALSE	SPDX ウェブサイトで使用できるライセンスを一覧表示するドロップダウンから有効なライセンス識別子を選択します。ドロップダウンで項目を選択すると、その下の [ライセンス] テキストボックスに値が入力されます。[Note:] (メモ:) ドロップダウンでライセンスを選択すると、ライセンステキストボックスが表示され、手動で行った編集はすべて破棄されます。

プロパティ	必要	説明
ライセンス	FALSE	.txt ライセンスファイルをアップロードするか、前述した [Spdx ライセンス] ドロップダウンからライセンスを選択します。[Spdx ライセンス] ドロップダウンからライセンスを選択すると、[ライセンス] テキストボックスに自動的に値が入力されます。ライセンスファイルをアップロードするか、[Spdx ライセンス] ドロップダウンからライセンスを選択した後で、このテキストボックスの内容を手動で編集できます。ただし、ドロップダウンから別の Spdx ライセンスを選択すると、以前に手動で行った編集はすべて破棄されます。 これはオプションフィールドですが、アプリケーションを公開共有するには、ライセンスを指定する必要があります。

プロパティ	必要	説明
Readme	FALSE	Readme ファイルの内容をアップロードします。このファイルは、テキスト形式またはマークダウン形式です。これらの内容は、AWS Serverless Application Repository でアプリケーションの詳細ページに表示されます。ファイルをアップロードした後で、このテキストボックスの内容を手動で編集できます。
Semantic version	FALSE	アプリケーションのセマンティックバージョンです。詳細については、セマンティックバージョニングのウェブサイトを参照してください。 アプリケーションをパブリックにするには、このプロパティに値を指定する必要があります。
Source code Url	FALSE	アプリケーションのソースコードを含むパブリックリポジトリへのリンク。
SAM template	TRUE	使用する AWS リソースを定義する有効な AWS Serverless Application Model (AWS SAM) テンプレート。

アプリケーションの共有

公開アプリケーションには、次の 3 つのカテゴリのいずれかでアクセス許可を設定できます。

- 非公開 (デフォルト) - 同じアカウントで作成され、他の AWS アカウントと共有されていないアプリケーション。非公開アプリケーションをデプロイするアクセス許可を持つのは、AWS アカウントを共有するコンシューマーのみです。
- 非公開共有 - パブリッシャーが一連の特定の AWS アカウントまたは AWS Organization の AWS アカウントと明示的に共有しているアプリケーション。コンシューマーは、これらの AWS アカウントまたは AWS Organization と共有されているアプリケーションをデプロイするためのアクセス許可を付与されます。AWS Organizations の詳細については、「AWS Organizations ユーザーガイド」を参照してください。
- 公開共有 – パブリッシャーがすべてのユーザーと共有しているアプリケーション。すべてのコンシューマーは、すべての公開共有アプリケーションをデプロイするためのアクセス許可を付与されます。

アプリケーションを AWS Serverless Application Repository に公開すると、デフォルトでは 非公開に設定されます。このセクションでは、特定の AWS アカウントまたは AWS Organization とアプリケーションを非公開で共有する方法、またはすべてのユーザーと公開で共有する方法について説明します。

コンソールからのアプリケーションの共有

アプリケーションを他のユーザーと共有するには、次の 2 つのオプションがあります。1) 特定と共有するAWSアカウントまたはAWS内のアカウントAWS組織、または 2) 全員とパブリックに共有します。AWS Organizations の詳細については、「AWS Organizations ユーザーガイド」を参照してください。

オプション 1: アプリケーションを特定のユーザーと共有するにはAWSアカウントまたはお客様の内のアカウントAWS会社

1. AWS Serverless Application Repository コンソールを開きます。
2. ナビゲーションペインで、[Published Applications (後悔アプリケーション)] を選択して、作成したアプリケーションの一覧を表示します。
3. 共有するアプリケーションを選択します。
4. [共有] タブを選択します。

5. [Application policy statements (アプリケーションポリシーステートメント)] セクションで、[Create Statement (ステートメントの作成)] ボタンを選択します。
6. [Statement Configuration (ステートメント設定)] ウィンドウで、アプリケーションの共有方法に基づいてフィールドに入力します。

> **Note**
>
> 組織と共有する場合は、AWS アカウントがメンバーである組織のみを指定できます。メンバーではない AWS Organization を指定しようとすると、エラーが発生します。アプリケーションを AWS Organization と共有するには、将来共有を取り消す必要がある場合に備えて、`UnshareApplication` アクションがポリシーステートメントに追加されることを承認する必要があります。

7. [保存] ボタンを選択します。

オプション 2: アプリケーションをすべてのユーザーと公開するには

1. AWS Serverless Application Repository コンソールを開きます。
2. ナビゲーションペインで、[Published Applications (後悔アプリケーション)] を選択して、作成したアプリケーションの一覧を表示します。
3. 共有するアプリケーションを選択します。
4. [共有] タブを選択します。
5. [Public Sharing (公開共有)] セクションで、[編集] ボタンを選択します。
6. [Public Sharing (公開共有)] で、[有効] ラジオボタンを選択します。
7. テキストボックスにアプリケーション名を入力し、[保存] ボタンを選択します。

> **Note**
>
> アプリケーションを公開共有するには、`SemanticVersion` プロパティ `LicenseUrl` とプロパティの両方が設定されている必要があります。

AWS CLI からのアプリケーションの共有

AWS CLI を使用してアプリケーションを共有するには、`put-application-policy` コマンドを使用してアクセス許可を付与し、プリンシパルとして共有する AWS アカウントを指定します。

AWS CLI を使用したアプリケーションの共有の詳細については、AWS Serverless Application Repositoryアプリケーションポリシーの例を参照してください。

アプリケーションの共有解除

AWS Organization からアプリケーションの共有を解除するには、次の 2 つのオプションがあります。

1. アプリケーションの発行者は、`put-application-policy` コマンドを使用してアクセス許可を削除できます。
2. AWS Organization の管理アカウントのユーザーは、アプリケーションが別のアカウントのユーザーによって公開された場合でも、組織と共有されている任意のアプリケーションに対してアプリケーションの共有解除オペレーションを実行できます。

   > **Note**
   >
   > アプリケーションを「アプリケーションの共有解除」オペレーションで AWS Organization から共有解除すると、再び AWS Organization と共有することはできません。

   AWS Organizations の詳細については、「AWS Organizations ユーザーガイド」を参照してください。

アクセス許可を削除するパブリッシャー

コンソールからアクセス許可を削除するパブリッシャー

AWS Management Console を使用してアプリケーションの共有を解除するには、他の AWS アカウントと共有するポリシーステートメントを削除します。これを実行するには、以下の手順を実行します。

1. AWS Serverless Application Repository コンソールを開きます。
2. 左側のナビゲーションペインで、[使用可能なアプリケーション] を選択します。
3. 共有解除するアプリケーションを選択します。
4. [共有] タブを選択します。

5. [Application policy statements (アプリケーションポリシーステートメント)] セクションで、共有解除するアカウントとアプリケーションを共有しているポリシーステートメントを選択します。
6. [Delete] (削除) を選択します。
7. 確認メッセージが表示されます。[削除] をもう一度選択します。

AWS CLI からアクセス許可を削除するパブリッシャー

AWS CLI を使用してアプリケーションの共有を解除するには、パブリッシャーは `put-application-policy` コマンドを使用してアクセス許可を削除または変更し、アプリケーションを非公開にしたり、別の AWS アカウントと共有することができます。

AWS CLI を使用したアクセス許可の変更の詳細については、AWS Serverless Application Repository アプリケーションポリシーの例を参照してください。

アプリケーションを共有解除する管理アカウント

コンソールを使用して AWS Organization のアプリケーションを共有解除する管理アカウント

AWS Management Console を使用して AWS Organization からアプリケーションを共有解除するには、管理アカウントのユーザーは次の操作を実行できます。

1. AWS Serverless Application Repository コンソールを開きます。
2. 左側のナビゲーションペインで、[使用可能なアプリケーション] を選択します。
3. アプリケーションのタイルで、[共有解除] を選択します。
4. [共有解除] メッセージボックスで、組織 ID とアプリケーション名を入力し、[保存] を選択して、アプリケーションの共有解除を確認します。

AWS CLI を使用して AWS Organization のアプリケーションを共有解除する管理アカウント

AWS Organization からアプリケーションの共有を解除するには、管理アカウントのユーザーは `aws serverlessrepo unshare-application` コマンドを実行できます。

次のコマンドは、AWS Organization からアプリケーションを共有解除します。ここで、*`###### # ID`* はアプリケーションの Amazon リソースネーム (ARN) で、*`## ID`* は AWS Organization ID です。

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

アプリケーションの削除

アプリケーションを AWS Serverless Application Repository から削除するには、AWS Management Console または AWS SAM CLI を使用します。

アプリケーションの削除 (コンソール)

AWS Management Console から公開されたアプリケーションを削除するには、次の操作を行います。

1. AWS Serverless Application Repository コンソールを開きます。
2. [My Applications (マイアプリケーション)] に、削除するアプリケーションを選択します。
3. アプリケーションの詳細ページで、[Delete application (アプリケーションの削除)] を選択します。
4. [Delete application (アプリケーションの削除)] を選択して、削除を完了します。

アプリケーションの削除 (AWS CLI)

AWS CLI を使用して発行済みのアプリケーションを削除するには、`aws serverlessrepo delete-application` コマンドを実行します。

次のコマンドはアプリケーションを削除します。ここで、*`application-id`* はアプリケーションの Amazon リソースネーム (ARN) です。

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

既存アプリケーションの新しいバージョンの発行

このセクションでは、AWS SAM CLI または AWS Management Console を使用して、既存のアプリケーションの新しいバージョンを AWS Serverless Application Repository に発行する方法について説明します。新しいアプリケーションを発行する手順については、「アプリケーションを発行する方法」を参照してください。

既存アプリケーションの新しいバージョンの発行 (AWS CLI)

既存のアプリケーションの新しいバージョンを発行する最も簡単な方法は、一連の AWS SAM CLI コマンドを使用することです。詳細については、AWS Serverless Application Model (AWS SAM) デベロッパーガイドの AWS SAM CLI を使用したアプリケーションの発行を参照してください。

既存アプリケーションの新しいバージョンの発行 (コンソール)

以前に発行したアプリケーションの新しいバージョンを発行するには、次の手順に従います。

1. AWS Serverless Application Repository コンソールを開きます。
2. ナビゲーションペインで、[My Applications] を選択し、作成済みのアプリケーションを一覧表示します。
3. 新しいバージョンを発行するアプリケーションを選択します。
4. [Publish new version] (新しいバージョンを発行) を選択します。
5. [Versions] に、次のアプリケーション情報を入力します。

プロパティ	必要	説明
Semantic version	TRUE	アプリケーションのセマンティックバージョンです。詳細については、セマンティックバージョニングのウェブサイトを参照してください。 アプリケーションをパブリックにするには、このプロパティに値を指定する必要があります。
Source code Url	FALSE	アプリケーションのソースコードを含むパブリックリポジトリへのリンク。
SAM template	TRUE	使用する AWS リソースを定義する有効な AWS Serverless Application Model (AWS SAM) テンプレート。

6. [Publish version] を選択します。

検証済み作成者バッジ

AWS が依頼者から提供された情報を信義誠実の原則に基づいて審査した結果、適切で良識的なサービスプロバイダーであると判断し、さらに依頼者のアイデンティティが主張どおりであることを確認した場合、作成者は AWS Serverless Application Repository の検証済み作成者とみなされます。

検証済み作成者のアプリケーションには、検証済み作成者バッジと作成者のパブリックプロファイルへのリンクが表示されます。検証済み作成者バッジは、検索結果とアプリケーションの詳細ページの両方に表示されます。

検証済み作成者バッジのリクエスト

検証済み作成者としての承認をリクエストするには、AWS Serverless Application Repositoryにメールを送信するserverlessrepo-verified-author@amazon .com。以下の情報を指定する必要があります。

- 作成者名
- AWS アカウント ID
- パブリックアクセスが可能なプロファイルリンク (ユーザーなど)GitHubまたはLinkedInプロフィール

検証済み作成者バッジのリクエストを送信すると、数日以内に AWS から回答があります。リクエストが承認される前に、追加情報の提供を求められる場合があります。

リクエストが承認されると、検証済み作成者バッジが 1 日以内にアプリケーションに表示されます。

> **Note**
>
> 検証済み作成者バッジは、AWS アカウントと作成者名の両方に一致するすべてのアプリケーションに表示されます。AWS アカウントは複数の作成者を持つことができるため、作成者名が異なるアプリケーションにはバッジが表示されません。作成者名が異なるアプリケーションに作成者バッジを表示するには、その作成者用の別のリクエストを送信する必要があります。

Lambda レイヤーの共有

Lambda レイヤーに機能を実装している場合は、レイヤーのグローバルインスタンスをホストしなくても、レイヤーを共有できます。この方法でレイヤーを共有すると、他のユーザーは各自のアカウントにレイヤーのインスタンスをデプロイできます。クライアントアプリケーションは、レイヤーのグローバルインスタンスに依存する必要がなくなります。AWS Serverless Application Repository を使用すると、この方法で簡単に Lambda レイヤーを共有できます。

Lambda レイヤーの詳細については、AWS Lambda デベロッパーガイドのAWS Lambda レイヤーを参照してください。

仕組み

AWS Serverless Application Repository を使用してレイヤーを共有する手順は次のとおりです。この手順により、レイヤーのコピーをユーザーの AWS アカウントに作成できます。

1. レイヤーをリソース (つまり、`AWS::Serverless::LayerVersion` リソースまたは `AWS::Lambda::LayerVersion` リソースのいずれか) として含む AWS SAM テンプレートを使用してサーバーレスアプリケーションを定義します。
2. アプリケーションを AWS Serverless Application Repository に発行し、公開または非公開として共有します。
3. ユーザーは、アプリケーションをデプロイし、各自の AWS アカウントにレイヤーのコピーを作成します。これで、ユーザーは各自のクライアントアプリケーションを使用して、各自の AWS アカウント内のレイヤーの Amazon リソースネーム (ARN) を参照できます。

例

共有する Lambda レイヤーが含まれているアプリケーション用の AWS SAM テンプレートの例を次に示します。

```
Resources:
  SharedLayer:
    Type: AWS::Serverless::LayerVersion
    Properties:
      LayerName: shared-layer
      ContentUri: source/layer-code/
      CompatibleRuntimes:
        - python3.7
```

```
Outputs:
  LayerArn:
    Value: !Ref SharedLayer
```

ユーザーが AWS Serverless Application Repository からアプリケーションをデプロイすると、ユーザーの AWS アカウントにレイヤーが作成されます。レイヤーの ARN は次のようになります。

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

これで、ユーザーは、次の例に示すように、この ARN を各自のクライアントアプリケーションで参照できます。

```
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
```

アプリケーションをデプロイする

このセクションでは、AWS Serverless Application Repository に発行したサーバーレスアプリケーションを見つけてデプロイする方法について説明します。公開サイトにアクセスすることで、一般公開されているアプリケーションを AWS アカウントを持っていなくても参照することができます。または、AWS Lambda コンソール内からアプリケーションを参照できます。

一部のアプリケーションには、作成者のプロファイルにリンクされた検証済み作成者バッジがあります。AWS が依頼者から提供された情報を信義誠実の原則に基づいて審査した結果、適切で良識的なサービスプロバイダーであると判断し、さらに依頼者のアイデンティティが主張どおりであることを確認した場合、作成者は検証済み作成者とみなされます。

AWS Serverless Application Repository からアプリケーションをデプロイする前に、以下のトピックを参照し、アプリケーションをデプロイするためのアクセス許可とアプリケーションの機能を確認してください。

トピック

- アプリケーションをデプロイするためのアクセス許可
- アプリケーションの機能: IAM ロール、リソースポリシー、ネストされたアプリケーション
- アプリケーションをデプロイする方法

アプリケーションをデプロイするためのアクセス許可

AWS Serverless Application Repository でアプリケーションをデプロイするには、そのためのアクセス許可が必要です。デプロイのアクセス許可が付与されるアプリケーションには、次の 3 つのカテゴリがあります。

- 非公開 – 同じアカウントで作成され、他のアカウントと共有されていないアプリケーション。この AWS アカウントを使用して作成されたアプリケーションをデプロイするためのアクセス許可が付与されます。
- 非公開共有 – パブリッシャーが一連の特定の AWS アカウントと明示的に共有しているアプリケーション。これらの AWS アカウントと共有されているアプリケーションをデプロイするためのアクセス許可が付与されます。
- 公開共有 – パブリッシャーがすべてのユーザーと共有しているアプリケーション。すべての公開共有アプリケーションをデプロイするためのアクセス許可が付与されます。

検索および参照できるのは、アクセス許可を付与されているアプリケーションのみです。これには、AWS アカウントを使用して作成されたアプリケーション、AWS アカウントと非公開共有されているアプリケーション、および公開共有されているアプリケーションが含まれます。その他すべてのアプリケーションは表示されません。

> **Important**
>
> ネストされたアプリケーションを含むアプリケーションは、ネストされたアプリケーションの共有制限を継承します。例えば、あるアプリケーションが一般公開されていて、それにこの親アプリケーションを作成した AWS アカウントとプライベート限定で共有されている、ネストされたアプリケーションが含まれているとします。この場合、このネストされたアプリケーションをデプロイするアクセス許可が AWS アカウントにないときに、親アプリケーションをデプロイすることはできません。ネストされたアプリケーションの詳細については、AWS Serverless Application Model デベロッパーガイドのネストされたアプリケーションを参照してください。

アプリケーションの機能: IAM ロール、リソースポリシー、ネストされたアプリケーション

アプリケーションをデプロイする前に、AWS Serverless Application Repository は、アプリケーションのテンプレートを参照し、作成する必要がある IAM ロール、AWS リソースポリシー、およびネストされたアプリケーションがテンプレートに指定されていることを確認します。IAM リソース (フルアクセス権を持つ IAM ロールなど) は、AWS アカウントの任意のリソースを変更できます。したがって、拡大した権限で意図していないリソースを作成しないように、続行する前にアプリケーションに関連付けられたアクセス許可を確認することをお勧めします。そのように実行したことを確認するには、ユーザーの代わりに AWS Serverless Application Repository がアプリケーションをデプロイする前に、アプリケーションが機能を備えていることを確認する必要があります。

アプリケーションに
は、CAPABILITY_IAM、CAPABILITY_NAMED_IAM、CAPABILITY_RESOURCE_POLICY、および CAPABILITY_AUTO_EXPAND の 4 つの機能のうち、いずれでも含めることができます。

次のリソースで
は、、AWS::IAM::Group、、AWS::IAM::InstanceProfileAWS::IAM::Policy、CAPABILITY_IAMCAPABILITY_
たはを指定する必要があります。アプリケーションにカスタム名を持つ IAM リソースがある場合

は、CAPABILITY_NAMED_IAM を指定する必要があります。機能を指定する方法の例については、「アプリケーション機能の検索と承認 (AWS CLI)」を参照してください。

次のリソースでは、AWS::Lambda::LayerVersionアクセス許可、AWS::Events::EventBusポリシーAWS::Lambda::Permission、AWS:: iam: Policy、、、AWS::ApplicationAutoScaling::ScalingPolicyAWS::S3::BucketPolicyAWS::SQS::QueuePolicy、C よびを指定する必要があります。

1 つまたは複数のネストされたアプリケーションが含まれているアプリケーションで は、CAPABILITY_AUTO_EXPAND を指定する必要があります。ネストされたアプリケーションの詳細については、AWS Serverless Application Model デベロッパーガイドのネストされたアプリケーションを参照してください。

アプリケーションの機能の検索と承認 (コンソール)

AWS Serverless Application Repository で使用できるアプリケーションは AWS Serverless Application Repository ウェブサイト、または、Lambda コンソール ([AWS Serverless Application Repository] タブの [Create Function] (関数の作成) ページ) で確認できます。

カスタム IAM ロールまたはリソースポリシーを作成するための機能の承認を必要とするアプリケーションは、デフォルトでは検索結果に表示されません。これらの機能が含まれているアプリケーションを検索するには、[Show apps that create custom IAM roles or resource policies (カスタム IAM ロールまたはリソースポリシーを作成するアプリを表示)] チェックボックスをオンにする必要があります。

アプリケーションの機能は、アプリケーションを選択するときに [アクセス許可] タブの下で確認できます。アプリケーションをデプロイするには、[I acknowledge this application creates custom IAM roles or resource policies (このアプリケーションがカスタム IAM ロールまたはリソースポリシーを作成することを承認します)] チェックボックスをオンにする必要があります。これらの機能を承認しない場合は、エラーメッセージ [Acknowledgement required. デプロイするには、「アプリケーションパラメータの設定」セクションのボックスにチェックを入れます。

アプリケーションの機能の表示 (AWS CLI)

AWS CLI を使用してアプリケーションの機能を表示するには、まずアプリケーションの Amazon リソースネーム (ARN) が必要です。その後、次のコマンドを実行できます。

```
aws serverlessrepo get-application \
```

```
--application-id application-arn
```

requiredCapabilities レスポンスプロパティには、アプリケーションをデプロイする前に、承認する必要があるアプリケーション機能が一覧表示されます。requiredCapabilities プロパティが空の場合、アプリケーションには必要な機能がないことに注意してください。

アプリケーションをデプロイする方法

このセクションでは、AWS Management Console または AWS CLI を使用して AWS Serverless Application Repository からサーバレスアプリケーションをデプロイする手順を示します。

新しいアプリケーションのデプロイ (コンソール)

このセクションでは、AWS Management Console を使用して AWS Serverless Application Repository から新しいアプリケーションをデプロイする方法を示します。既存のアプリケーションの新しいバージョンをデプロイする手順については、「アプリケーションの更新」を参照してください。

アプリケーションの参照、検索、およびデプロイ

AWS Serverless Application Repository でアプリケーションを検索、設定、およびデプロイするには、次の手順を使用します。

AWS Serverless Application Repository でアプリケーションを検索して設定するには

1. AWS Serverless Application Repository のパブリックホームページ を開くか、AWS Lambda コンソールを開きます。[Create function (関数の作成)] を選択し、[Browse serverless app repository (サーバーレスアプリケーションリポジトリを参照)] を選択します。
2. アプリケーションを参照または検索します。

> **Note**
>
> カスタム IAM ロールまたはリソースポリシーが含まれているアプリケーションを表示するには、[Show apps that create custom IAM roles or resource policies (カスタム IAM ロールまたはリソースポリシーを作成するアプリを表示)] チェックボックスをオンにします。カスタム IAM ロールとリソースポリシーの詳細については、「Acknowledging Application Capabilities (アプリケーション承認機能)」を参照してください。

3. アプリケーションを選択して、そのアクセス許可、機能、AWS のお客様がデプロイした回数などの詳細を表示します。

   アプリケーションをデプロイしようとしている AWS リージョンのデプロイ数が表示されます。

4. アプリケーションの詳細ページで、AWS SAM テンプレート、ライセンス、および readme ファイルを表示して、アプリケーションのアクセス許可およびリソースを確認します。このページで、公開共有されているアプリケーションの [Source code URL (ソースコード URL)] リンクを見つけることもできます。アプリケーションにネストされたアプリケーションが含まれている場合、このページでネストされたアプリケーションの詳細を表示することもできます。

5. [Application settings (アプリケーションの設定)] セクションでアプリケーションを設定します。特定のアプリケーションを設定する際のガイダンスについては、アプリケーションの readme ファイルを参照してください。

   たとえば、設定要件には、アプリケーションにアクセスさせるリソースの名前の指定が含まれる場合があります。Amazon DynamoDB テーブル、Amazon S3 バケット、または Amazon API Gateway API などのリソースです。

6. [Deploy] (デプロイ) をクリックします。これにより、[Deployment status] (デプロイのステータス) ページに移動します。

   > **Note**
   >
   > アプリケーションに承認を必要とする機能がある場合は、アプリケーションをデプロイする前に [I acknowledge this application creates custom IAM roles or resource polices (このアプリケーションがカスタム IAM ロールまたはリソースポリシーを作成することを承認します)] チェックボックスをオンにします。そうしないと、エラーが発生します。カスタム IAM ロールとリソースポリシーの詳細については、「Acknowledging Application Capabilities (アプリケーション承認機能)」を参照してください。

7. [Deployment status (デプロイのステータス)] ページで、デプロイの進捗状況を表示できます。デプロイが完了するのを待っている間に、他のアプリケーションを検索して参照し、Lambda コンソールからこのページに戻ることができます。

アプリケーションが正常にデプロイされたら、既存の AWS ツールを使用して作成済みのリソースを確認し、管理できます。

新しいアプリケーションのデプロイ (AWS CLI)

このセクションでは、AWS CLI を使用して AWS Serverless Application Repository から新しいアプリケーションをデプロイする方法を示します。既存のアプリケーションの新しいバージョンをデプロイする手順については、「アプリケーションの更新」を参照してください。

アプリケーション機能の検索と承認 (AWS CLI)

AWS CLI を使用してアプリケーションの機能を確認するには、次の手順に従います。

1. アプリケーションの機能を確認してください。AWS CLI次のコマンドを使用して、アプリケーションの機能を確認します。

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

   requiredCapabilities レスポンスプロパティには、アプリケーションをデプロイする前に、承認する必要があるアプリケーション機能が一覧表示されます。AWSSDK の GetApplication API を使用してこのデータを取得することもできます。

2. チェンジセットを作成します。AWS CloudFormation 変更セットを作成する際に、必要な機能のセットを提供する必要があります。たとえば、次の AWS CLI コマンドを使用し、アプリケーションの機能を承認して、デプロイします。

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

   このコマンドが正常に実行されると、変更セット ID が返されます。この変更セット ID は次のステップで必要になります。また、AWS SDK で CreateCloudFormationChangeSet API を使用して変更セットを作成することもできます。

   たとえば、AWS CLI次のコマンドは、AWS::IAM::Roleカスタム名のリソースと 1 つ以上のネストされたアプリケーションを含むアプリケーションを承認します。

   ```
   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. チェンジセットを実行します。変更セットを実行すると、デプロイが実際に行われます。前のステップで変更セットを作成したときに返された変更セット ID を指定します。

   次の AWS CLI コマンド例では、アプリケーションの変更セットを実行してアプリケーションをデプロイします。

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

   また、AWS SDK で ExecuteChangeSet API を使用して変更セットを実行することもできます。

アプリケーションスタックの削除

AWS Serverless Application Repository を使用して以前にデプロイされたアプリケーションを削除するには、AWS CloudFormation スタックを削除する場合と同じ手順に従います。

- AWS Management Console: AWS Management Console を使用してアプリケーションを削除する場合は、AWS CloudFormation ユーザーガイドの AWS CloudFormation コンソールでのスタックの削除を参照してください。
- AWS CLI: AWS CLI を使用してアプリケーションを削除する場合は、AWS CloudFormation ユーザーガイドのスタックの削除を参照してください。

アプリケーションの更新

AWS Serverless Application Repository からアプリケーションをデプロイした後で、アプリケーションの更新が必要になる場合があります。たとえば、アプリケーションの設定を変更したり、アプリケーションを発行済みの最新バージョンに更新したりする場合があります。

以下のセクションでは、AWS Management Console または AWS CLI を使用して、新しいバージョンのアプリケーションをデプロイする方法を示します。

アプリケーションの更新 (コンソール)

以前にデプロイしたアプリケーションを更新するには、新しいアプリケーションをデプロイするのと同じ手順を使用し、最初にデプロイしたときと同じアプリケーション名を指定します。この場合、AWS Serverless Application Repository はアプリケーション名の先頭に `serverlessrepo-` に付加します。ただし、新しいバージョンのアプリケーションをデプロイする場合は、先頭に `serverlessrepo-` を付加せずに元のアプリケーション名を指定します。

たとえば、MyApplication という名前のアプリケーションをデプロイした場合、スタック名は serverlessrepo-MyApplication となります。アプリケーションを更新するには、名前 MyApplication をもう一度指定します。serverlessrepo-MyApplication の完全なスタック名を指定しないでください。

他のすべてのアプリケーション設定では、以前のデプロイと同じ値を保持するか、新しい値を指定できます。

アプリケーションの更新 (AWS CLI)

以前にデプロイしたアプリケーションを更新するには、新しいアプリケーションをデプロイするのと同じ手順を使用し、最初にデプロイしたときと同じ *--stack-name* を指定します。この場合、AWS Serverless Application Repository はスタック名の先頭に serverlessrepo- を付加します。ただし、新しいバージョンのアプリケーションをデプロイする場合は、先頭に serverlessrepo- を付加せずに元のスタック名を指定します。

たとえば、MyApplication というスタック名のアプリケーションをデプロイした場合、作成されるスタック名は serverlessrepo-MyApplication となります。アプリケーションを更新するには、名前 MyApplication をもう一度指定します。serverlessrepo-MyApplication の完全なスタック名を指定しないでください。

AWS Serverless Application Repository でのセキュリティ

AWSでは、クラウドのセキュリティが最優先事項です。AWSのお客様は、セキュリティを最も重視する組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャから利点を得られます。

セキュリティは、AWSお客様の間の共有責任です。共有責任モデルでは、これをクラウドのセキュリティおよびクラウド内のセキュリティとして説明しています。

- クラウドのセキュリティ - AWS は、AWS クラウドで AWS サービスを実行するインフラストラクチャを保護する責任を負います。また、AWS は、安全に使用できるサービスを供給します。AWS コンプライアンスプログラムの一環として、サードパーティーの監査が定期的にセキュリティの有効性をテストおよび検証しています。AWS Serverless Application Repository に適用されるコンプライアンスプログラムの詳細については、「コンプライアンスプログラム対象範囲内の AWS のサービス」を参照してください。
- クラウド内のセキュリティ–お客様の責任は、使用するAWSのサービスに応じて判断されます。また、お客様は、データの機密性、お客様の会社の要件、および適用可能な法律および規制など、その他の要因についても責任を担います。

このドキュメントは、AWS Serverless Application Repository を使用する際の責任共有モデルの適用方法を理解するのに役立ちます。以下のトピックでは、セキュリティおよびコンプライアンスの目的を達成するために AWS Serverless Application Repository を設定する方法を示します。また、AWS リソースのモニタリングや保護に役立つ、他の AWS Serverless Application Repository サービスの使用方法についても説明します。

トピック

AWS Serverless Application Repository でのデータ保護

AWS責任共有モデルは、AWS Serverless Application Repository でのデータ保護に適用されます。このモデルで説明されているように、AWS は、AWS クラウド のすべてを実行するグローバルインフラストラクチャを保護する責任を負います。顧客は、このインフラストラクチャでホストされているコンテンツに対する管理を維持する責任があります。また、使用する AWS のサービスのセキュリティ設定と管理タスクにも責任があります。データプライバシーの詳細については、データプライバシーのよくある質問を参照してください。欧州でのデータ保護の詳細については、「AWS セキュリティブログ」に投稿された「AWS 責任共有モデルおよび GDPR」のブログ記事を参照してください。

データを保護するため、AWS アカウント の認証情報を保護し、AWS IAM Identity Center または AWS Identity and Access Management (IAM) を使用して個々のユーザーをセットアップすることをお勧めします。こうすると、それぞれのジョブを遂行するために必要なアクセス許可のみを各ユーザーに付与できます。また、次の方法でデータを保護することをお勧めします。

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 および TLS 1.3 をお勧めします。
- AWS CloudTrail で API とユーザーアクティビティログをセットアップします。
- AWS のサービス 内でデフォルトである、すべてのセキュリティ管理に加え、AWS の暗号化ソリューションを使用します。
- Amazon Macie などの高度なマネージドセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API を使用して AWS にアクセスするときに FIPS 140-2 検証済みの暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-2」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報は、タグ、または名前フィールドなどの自由形式のテキストフィールドに配置しないことを強くお勧めします。これは、コンソール、API、AWS CLI、または AWS SDK で AWS Serverless Application Repository または他の AWS のサービス を使用する場合も同様です。タグ、または名前に使用される自由形式のテキストフィールドに入力されるデータは、請求または診断ログに使用される場合があります。外部サーバーへ URL を供給する場合は、そのサーバーへのリクエストを検証するために、認証情報を URL に含めないことを強くお勧めします。

転送時の暗号化

AWS Serverless Application Repository API エンドポイントでは、HTTPS 経由の安全な接続のみがサポートされます。AWS Serverless Application Repository リソースを AWS Management Console、AWS SDK、または AWS Serverless Application Repository API を使用して管理する場合、すべての通信は Transport Layer Security (TLS) で暗号化されます。

API エンドポイントの全リストについては、の「AWSリージョンとエンドポイント」を参照してください。AWS 全般のリファレンス

保管時の暗号化

AWS Serverless Application Repository は、デプロイパッケージやレイヤーアーカイブなど、AWS Serverless Application Repository にアップロードされるファイルを暗号化します。

AWS Serverless Application Repositoryの Identity and Access Management

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御 AWS のサービス するのに役立つ です。IAM 管理者は、誰を認証 (サインイン) し、誰に AWS Serverless Application Repository リソースの使用を承認する (アクセス許可を付与する) かを制御します。IAM は、追加料金なしで AWS のサービス 使用できる です。

IAM の仕組みの概要については、IAM ユーザーガイドの IAM の仕組みについてを参照してください。

トピック

- 対象者
- アイデンティティを使用した認証
- ポリシーを使用したアクセスの管理
- AWS Serverless Application Repository で IAM を使用する方法
- AWS Serverless Application Repository アイデンティティベースのポリシーの例
- AWS Serverless Application Repositoryアプリケーションポリシーの例
- AWS Serverless Application RepositoryAPI のアクセス権限: アクションとリソースのリファレンス
- AWS Serverless Application Repository Identity and Access のトラブルシューティング

対象者

AWS Identity and Access Management (IAM) の使用方法は、 で行う作業によって異なります AWS Serverless Application Repository。

サービスユーザー – AWS Serverless Application Repository サービスを使用してジョブを実行する場合、管理者から必要な認証情報とアクセス許可が与えられます。さらに多くの AWS Serverless Application Repository 機能を使用して作業を行う場合は、追加のアクセス許可が必要になることがあります。アクセスの管理方法を理解すると、管理者から適切な権限をリクエストするのに役に立ちます。 AWS Serverless Application Repository機能にアクセスできない場合は、「AWS Serverless Application Repository Identity and Access のトラブルシューティング」を参照してください。

サービス管理者 – 社内の AWS Serverless Application Repository リソースを担当している場合は、通常、 へのフルアクセスがあります AWS Serverless Application Repository。サービスユーザーがどの AWS Serverless Application Repository 機能やリソースにアクセスするかを決めるのは管理者の仕事です。その後、IAM 管理者にリクエストを送信して、サービスユーザーの権限を変更する必要があります。このページの情報を点検して、IAM の基本概念を理解してください。会社で IAM をで使用する方法の詳細については、 AWS Serverless Application Repository「」を参照してくださいAWS Serverless Application Repository で IAM を使用する方法。

IAM 管理者 - 管理者は、 AWS Serverless Application Repositoryへのアクセスを管理するポリシーの書き込み方法の詳細について確認する場合があります。IAM で使用できる AWS Serverless Application Repository アイデンティティベースのポリシーの例を表示するには、「」を参照してくださいAWS Serverless Application Repository アイデンティティベースのポリシーの例。

アイデンティティを使用した認証

認証とは、ID 認証情報 AWS を使用して にサインインする方法です。として、IAM ユーザーとして AWS アカウントのルートユーザー、または IAM ロールを引き受けて認証（ にサインイン AWS) される必要があります。

ID ソースを介して提供された認証情報を使用して、フェデレーティッド ID AWS として にサインインできます。 AWS IAM Identity Center（ IAM Identity Center) ユーザー、会社のシングルサインオン認証、Google または Facebook の認証情報は、フェデレーション ID の例です。フェデレーティッド ID としてサインインする場合、IAM ロールを使用して、前もって管理者により ID フェデレーションが設定されています。フェデレーション AWS を使用して にアクセスすると、間接的にロールを引き受けることになります。

ユーザーのタイプに応じて、 AWS Management Console または AWS アクセスポータルにサインインできます。へのサインインの詳細については AWS、「 ユーザーガイド」の「 へのサインイン AWS アカウント方法AWS サインイン 」を参照してください。

AWS プログラムで にアクセスする場合、 は Software Development Kit (SDK) とコマンドラインインターフェイス (CLI) AWS を提供し、認証情報を使用してリクエストに暗号で署名します。 AWS ツールを使用しない場合は、リクエストに自分で署名する必要があります。推奨される方法を使用してリクエストを自分で署名する方法の詳細については、IAM ユーザーガイドの API AWS リクエストの署名を参照してください。

使用する認証方法を問わず、追加セキュリティ情報の提供をリクエストされる場合もあります。例えば、 AWS では、多要素認証 (MFA) を使用してアカウントのセキュリティを向上させることをお勧めします。詳細については、「AWS IAM Identity Center ユーザーガイド」の「Multi-factor authentication」(多要素認証) および「IAM ユーザーガイド」の「AWSでの多要素認証 (MFA) の使用」を参照してください。

AWS アカウント ルートユーザー

を作成するときは AWS アカウント、アカウント内のすべての AWS のサービス およびリソースへの完全なアクセス権を持つ 1 つのサインインアイデンティティから始めます。この ID は AWS アカウント ルートユーザーと呼ばれ、アカウントの作成に使用した E メールアドレスとパスワードでサインインすることでアクセスできます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザーの認証情報は保護し、ルートユーザーでしか実行できないタスクを実行するときに使用します。ルートユーザーとしてサインインする必要があるタスクの完全なリストについては、IAM ユーザーガイドの「ルートユーザー認証情報が必要なタスク」を参照してください。

IAM ユーザーとグループ

IAM ユーザーは、単一のユーザーまたはアプリケーションに対して特定のアクセス許可 AWS アカウント を持つ 内のアイデンティティです。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を保有する IAM ユーザーを作成する代わりに、一時的な認証情報を使用することをお勧めします。ただし、IAM ユーザーでの長期的な認証情報が必要な特定のユースケースがある場合は、アクセスキーをローテーションすることをお勧めします。詳細については、IAM ユーザーガイドの長期的な認証情報を必要とするユースケースのためにアクセスキーを定期的にローテーションするを参照してください。

IAM グループは、IAM ユーザーの集団を指定するアイデンティティです。グループとしてサインインすることはできません。グループを使用して、複数のユーザーに対して一度に権限を指定できま

す。多数のユーザーグループがある場合、グループを使用することで権限の管理が容易になります。例えば、IAMAdminsという名前のグループを設定して、そのグループにIAM リソースを管理する許可を与えることができます。

ユーザーは、ロールとは異なります。ユーザーは 1 人の人または 1 つのアプリケーションに一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユーザーには永続的な長期の認証情報がありますが、ロールでは一時的な認証情報が提供されます。詳細については、「IAM ユーザーガイド」の「IAM ユーザー (ロールではなく) の作成が適している場合」を参照してください。

IAM ロール

IAM ロールは、特定のアクセス許可 AWS アカウント を持つ 内のアイデンティティです。これは IAM ユーザーに似ていますが、特定のユーザーには関連付けられていません。ロール を切り替える AWS Management Console ことで、 で IAM ロールを一時的に引き受けることができます。ロールを引き受けるには、 または AWS API AWS CLI オペレーションを呼び出すか、カスタム URL を使用します。ロールを使用する方法の詳細については、「IAM ユーザーガイド」の「IAM ロールの使用」を参照してください。

IAM ロールと一時的な認証情報は、次の状況で役立ちます:

- フェデレーションユーザーアクセス – フェデレーティッド ID に許可を割り当てるには、ロールを作成してそのロールの許可を定義します。フェデレーティッド ID が認証されると、その ID はロールに関連付けられ、ロールで定義されている許可が付与されます。フェデレーションの詳細については、「IAM ユーザーガイド」の「Creating a role for a third-party Identity Provider」(サードパーティーアイデンティティプロバイダー向けロールの作成) を参照してください。IAM Identity Center を使用する場合は、許可セットを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM Identity Center は、権限セットを IAM のロールに関連付けます。アクセス許可セットの詳細については、「AWS IAM Identity Center ユーザーガイド」の「アクセス許可セット」を参照してください。
- 一時的な IAM ユーザー権限 - IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる権限を一時的に IAM ロールで引き受けることができます。
- クロスアカウントアクセス - IAM ロールを使用して、自分のアカウントのリソースにアクセスすることを、別のアカウントの人物 (信頼済みプリンシパル) に許可できます。クロスアカウントアクセス権を付与する主な方法は、ロールを使用することです。ただし、一部の では AWS のサービス、 (ロールをプロキシとして使用する代わりに) ポリシーをリソースに直接アタッチできます。クロスアカウントアクセスにおけるロールとリソースベースのポリシーの違いについては、

「IAM ユーザーガイド」の「IAM でのクロスアカウントのリソースへのアクセス」を参照してください。

- クロスサービスアクセス — 一部の は、他の の機能 AWS のサービス を使用します AWS のサービス。例えば、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。サービスでは、呼び出し元プリンシパルの許可、サービスロール、またはサービスリンクロールを使用してこれを行う場合があります。
  - 転送アクセスセッション (FAS) — IAM ユーザーまたはロールを使用して でアクションを実行する場合 AWS、ユーザーはプリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、 を呼び出すプリンシパルのアクセス許可を AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストのリクエストと組み合わせて使用します。FAS リクエストは、サービスが他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。
  - サービスロール - サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、「IAM ユーザーガイド」の「AWS のサービスにアクセス許可を委任するロールの作成」を参照してください。
  - サービスにリンクされたロール – サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールのアクセス許可を表示できますが、編集することはできません。
- Amazon EC2 で実行されているアプリケーション – IAM ロールを使用して、EC2 インスタンスで実行され、 AWS CLI または AWS API リクエストを行うアプリケーションの一時的な認証情報を管理できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。 AWS ロールを EC2 インスタンスに割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスにアタッチされたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、IAM ユーザーガイドのAmazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用して許可を付与するを参照してください。

IAM ロールと IAM ユーザーのどちらを使用するかについては、IAM ユーザーガイドの(IAM ユーザーではなく) IAM ロールをいつ作成したら良いのか?を参照してください。

ポリシーを使用したアクセスの管理

でアクセスを制御する AWS には、ポリシーを作成し、 AWS ID またはリソースにアタッチします。ポリシーは AWS 、アイデンティティまたはリソースに関連付けられているときにアクセス許可を定義する のオブジェクトです。 は、プリンシパル (ユーザー、ルートユーザー、またはロールセッション) がリクエストを行うときに、これらのポリシー AWS を評価します。ポリシーでの権限により、リクエストが許可されるか拒否されるかが決まります。ほとんどのポリシーは JSON ドキュメント AWS として に保存されます。JSON ポリシードキュメントの構造と内容の詳細については、IAM ユーザーガイドのJSON ポリシー概要を参照してください。

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

デフォルトでは、ユーザーやロールに権限はありません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き継ぐことができます。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションの許可を定義します。例えば、`iam:GetRole`アクションを許可するポリシーがあるとします。そのポリシーを持つユーザーは、 AWS Management Console、、 AWS CLIまたは AWS API からロール情報を取得できます。

アイデンティティベースのポリシー

アイデンティティベースポリシーは、IAM ユーザーグループ、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。アイデンティティベースのポリシーを作成する方法については、IAM ユーザーガイドのIAM ポリシーの作成を参照してください。

アイデンティティベースのポリシーは、さらにインラインポリシーまたはマネージドポリシーに分類できます。インラインポリシーは、単一のユーザー、グループ、またはロールに直接埋め込まれています。管理ポリシーは、 内の複数のユーザー、グループ、ロールにアタッチできるスタンドアロンポリシーです AWS アカウント。管理ポリシーには、 AWS 管理ポリシーとカスタマー管理ポリシーが含まれます。マネージドポリシーまたはインラインポリシーのいずれかを選択する方法については、IAM ユーザーガイドのマネージドポリシーとインラインポリシーの比較を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシー や Amazon S3 バケットポリシー があげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーでは、IAM の AWS マネージドポリシーを使用できません。

アクセスコントロールリスト (ACL)

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための許可を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Amazon S3、 AWS WAF、および Amazon VPC は、ACLs。ACL の詳細については、Amazon Simple Storage Service デベロッパーガイドのアクセスコントロールリスト (ACL) の概要を参照してください。

その他のポリシータイプ

AWS は、一般的ではない追加のポリシータイプをサポートします。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の権限を設定できます。

- アクセス許可の境界 - アクセス許可の境界は、アイデンティティベースのポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる権限の上限を設定する高度な機能です。エンティティにアクセス許可の境界を設定できます。結果として得られる権限は、エンティティのアイデンティティベースポリシーとそのアクセス許可の境界の共通部分になります。`Principal` フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、アクセス許可の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。アクセス許可の境界の詳細については、IAM ユーザーガイドのIAM エンティティのアクセス許可の境界を参照してください。
- サービスコントロールポリシー (SCPs) – SCPs は、 の組織または組織単位 (OU) に対する最大アクセス許可を指定する JSON ポリシーです AWS Organizations。 AWS Organizations は、 AWS

アカウント ビジネスが所有する複数の をグループ化して一元管理するためのサービスです。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP は、各 を含むメンバーアカウントのエンティティのアクセス許可を制限します AWS アカウントのルートユーザー。Organizations と SCP の詳細については、AWS Organizations ユーザーガイドの「SCP の仕組み」を参照してください。

- セッションポリシー - セッションポリシーは、ロールまたはフェデレーションユーザーの一時的なセッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。結果としてセッションの権限は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポリシーの共通部分になります。また、リソースベースのポリシーから権限が派生する場合もあります。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。詳細については、IAM ユーザーガイドのセッションポリシーを参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成される権限を理解するのがさらに難しくなります。複数のポリシータイプが関与する場合にリクエストを許可するかどうか AWS を決定する方法については、IAM ユーザーガイドの「ポリシー評価ロジック」を参照してください。

AWS Serverless Application Repository で IAM を使用する方法

AWS Serverless Application Repository へのアクセスを管理するために IAM を使用する前に、AWS Serverless Application Repository でどの IAM 機能が使用できるかを理解しておく必要があります。

IAM の仕組みの概要については、IAM ユーザーガイドの IAM の仕組みについてを参照してください。AWS Serverless Application Repository およびその他の AWS のサービスが IAM と連携する方法の概要を把握するには、IAM ユーザーガイドの IAM と連携する AWS のサービスを参照してください。

トピック

- AWS Serverless Application Repository アイデンティティベースのポリシー
- AWS Serverless Application Repositoryアプリケーションポリシー
- AWS Serverless Application Repository タグに基づいた承認
- AWS Serverless Application Repository IAM ロール

AWS Serverless Application Repository アイデンティティベースのポリシー

IAM アイデンティティベースポリシーでは、許可または拒否するアクションとリソース、またアクションを許可または拒否する条件を指定できます。AWS Serverless Application Repository は、特定のアクション、リソース、および条件キーをサポートしています。JSON ポリシーで使用するすべての要素については、「IAM ユーザーガイド」の「IAM JSON ポリシーの要素のリファレンス」を参照してください。

以下に示しているのは、アクセス許可ポリシーの例です。

```
{

    "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"
        }
    ]
}
```

このポリシーには以下の 2 つのステートメントがあります。

- 最初のステートメントは、すべての AWS Serverless Application Repository リソースの AWS Serverless Application Repository アクション (`serverlessrepo:CreateApplication`) に、ワイルドカード文字 (*) を `Resource` 値として指定して、アクセス権限を付与します。

- 2 番目のステートメントでは、AWS Serverless Application Repository アプリケーションの Amazon リソースネーム (ARN) を使用して AWS リソースへの AWS Serverless Application Repository アクション `serverlessrepo:CreateApplicationVersion` に対してアクセス許可を付与します。アプリケーションは、`Resource` 値を使用して指定します。

アイデンティティベースのポリシーでは、アクセス許可の付与先のプリンシパルを指定しないため、`Principal` 要素は指定されません。ユーザーにポリシーをアタッチすると、そのユーザーが暗黙のプリンシパルになります。IAM ロールにアクセス許可ポリシーをアタッチすると、ロールの信頼ポリシーで識別されたプリンシパルがアクセス許可を得ることになります。

すべての AWS Serverless Application Repository API オペレーションとそれらが適用される AWS リソースの表については、AWS Serverless Application RepositoryAPI のアクセス権限: アクションとリソースのリファレンスを参照してください。

アクション

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どの プリンシパル がどの リソース に対してどのような 条件 下で アクション を実行できるかということです。

JSON ポリシーの `Action` 要素には、ポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。ポリシーアクションの名前は通常、関連する AWS API オペレーションと同じです。一致する API オペレーションのない許可のみのアクションなど、いくつかの例外があります。また、ポリシーに複数のアクションが必要なオペレーションもあります。これらの追加アクションは、依存アクションと呼ばれます。

このアクションは、関連付けられたオペレーションを実行するための許可を付与するポリシーで使用されます。

AWS Serverless Application Repository のポリシーアクションは、アクションの前にプレフィックス `serverlessrepo:` を使用します。たとえば、AWS Serverless Application Repository `SearchApplications` API オペレーションで AWS Serverless Application Repository インスタンスを実行するためのアクセス許可をユーザーに付与するには、ポリシーに `serverlessrepo:SearchApplications` アクションを含めます。ポリシーステートメントには、`Action` または `NotAction` 要素を含める必要があります。AWS Serverless Application Repository は、このサービスで実行できるタスクを記述する独自のアクションのセットを定義します。

単一のステートメントに複数のアクションを指定するには、次のようにカンマで区切ります。

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

ワイルドカード (*) を使用して複数のアクションを指定することができます。例えば、List という単語で始まるすべてのアクションを指定するには、次のアクションを含めます。

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

AWS Serverless Application Repository アクションのリストを表示するには、IAM ユーザーガイドのAWS Serverless Application Repository によって定義されたアクションを参照してください。

リソース

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

Resource JSON ポリシー要素は、オブジェクトあるいはアクションが適用されるオブジェクトを指定します。ステートメントには、Resource または NotResource 要素を含める必要があります。ベストプラクティスとして、Amazon リソースネーム (ARN) を使用してリソースを指定します。これは、リソースレベルの許可と呼ばれる特定のリソースタイプをサポートするアクションに対して実行できます。

オペレーションのリスト化など、リソースレベルの許可をサポートしないアクションの場合は、ステートメントがすべてのリソースに適用されることを示すために、ワイルドカード (*) を使用します。

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

AWS Serverless Application Repository の場合、プライマリ AWS リソースは AWS Serverless Application Repository アプリケーションです。AWS Serverless Application Repository アプリケーションには、次の表に示すように、一意の Amazon リソースネーム (ARN) が関連付けられています。

AWS リソースタイプ	Amazon リソースネーム (ARN) 形式
アプリケーション	arn:*partition* :serverlessrepo:*region*:*account-id* :applications/*application-name*

ARN の形式の詳細については、「Amazon リソースネーム (ARN) と AWS サービスの名前空間」を参照してください。

すべての AWS リソースの `serverlessrepo:ListApplications` アクションのアクセス許可を付与するポリシーの例を次に示します。現在の実装では、AWS Serverless Application Repository で一部の API アクションについて、AWS リソース ARN を使用した特定の AWS リソースの識別 (リソースレベルのアクセス許可とも呼ばれます) がサポートされていません。このような場合は、ワイルドカード文字 (*) を指定する必要があります。

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

すべての AWS Serverless Application Repository API アクションとそれらが適用される AWS リソースの表については、AWS Serverless Application RepositoryAPI のアクセス権限: アクションとリソースのリファレンスを参照してください。

条件キー

AWS Serverless Application Repository にはサービス固有の条件キーはありませんが、いくつかのグローバル条件キーの使用がサポートされています。すべてのAWSグローバル条件キーを確認するには、IAM ユーザーガイドの「AWS グローバル条件コンテキストキー」を参照してください。

例

AWS Serverless Application Repository アイデンティティベースのポリシーの例を表示するには、AWS Serverless Application Repository アイデンティティベースのポリシーの例を参照してください。

AWS Serverless Application Repositoryアプリケーションポリシー

アプリケーションポリシーは、指定されたプリンシパルまたはAWS Serverless Application Repository PrincipalOrgがアプリケーションで実行できるアクションを決定します。

AWS Serverless Application Repository アプリケーションに関連付けられるポリシーに権限を追加できます。AWS Serverless Application Repositoryアプリケーションに添付された権限ポリシーは、アプリケーションポリシーと呼ばれます。アプリケーションポリシーは IAM リソースベースのポリシーを拡張したものです。AWS Serverless Application Repository主なリソースはアプリケーションです。AWS Serverless Application Repository アプリケーションポリシーを使用して、アプリケーションのデプロイのアクセス権限を管理できます。

AWS Serverless Application Repository アプリケーションポリシーは、アプリケーションのデプロイと、これらのアプリケーションの検索や詳細の表示などの関連操作に必要なアクセス許可をコンシューマーに付与するためにパブリッシャーが主に使用します。パブリッシャーは、アプリケーションへのアクセス許可を次の 3 つのカテゴリに設定できます。

- 非公開 – 同じアカウントで作成され、他のアカウントと共有されていないアプリケーション。この AWS アカウントを使用して作成されたアプリケーションをデプロイするためのアクセス許可が付与されます。
- 非公開共有 - パブリッシャーが一連の特定の AWS アカウントまたは AWS Organizations と明示的に共有しているアプリケーション。これらの AWS アカウントまたは AWS Organization と共有されているアプリケーションをデプロイするためのアクセス許可が付与されます。
- 公開共有 – パブリッシャーがすべてのユーザーと共有しているアプリケーション。すべての公開共有アプリケーションをデプロイするためのアクセス許可が付与されます。

アクセス許可を付与するには、AWS CLI、AWS SDK、または AWS Management Console を使用します。

例

AWS Serverless Application Repositoryアプリケーションポリシーの管理の例については、を参照してくださいAWS Serverless Application Repositoryアプリケーションポリシーの例。

AWS Serverless Application Repository タグに基づいた承認

AWS Serverless Application Repository は、タグに基づくリソースやアクションへのアクセスコントロールをサポートしていません。

AWS Serverless Application Repository IAM ロール

IAM ロールは AWS アカウント内のエンティティで、特定の許可を持っています。

AWS Serverless Application Repository による一時的な認証情報の使用

一時的な認証情報を使用して、フェデレーションでサインイン、IAM ロールを引き受ける、またはクロスアカウントロールを引き受けることができます。一時的なセキュリティ認証情報は、AWS STSAssumeRoleまたはなどの API オペレーションを呼び出すことで取得できますGetFederationToken。

AWS Serverless Application Repository は一時的な認証情報の使用をサポートしています。

サービスリンクロール

AWS Serverless Application Repository はサービスにリンクされたロールをサポートしていません。

サービスロール

AWS Serverless Application Repository はサービスロールをサポートしていません。

AWS Serverless Application Repository アイデンティティベースのポリシーの例

デフォルトでは、IAM ユーザーおよびロールには、AWS Serverless Application Repository リソースを作成または変更するアクセス許可はありません。また、AWS Management Console や AWS CLI、AWS API を使用してタスクを実行することもできません。IAM 管理者は、ユーザーとロールに必要な、指定されたリソースで特定の API オペレーションを実行する許可をユーザーとロールに付与する IAM ポリシーを作成する必要があります。続いて、管理者はそれらの許可が必要な IAM ユーザーまたはグループにそのポリシーを添付します。

JSON ポリシードキュメントのこれらの例を使用して、IAM アイデンティティベースのポリシーを作成する方法については、IAM ユーザーガイドの JSON タブでのポリシーの作成を参照してください。

トピック

ポリシーのベストプラクティス

アイデンティティベースのポリシーは非常に強力です。アカウント内で、AWS Serverless Application Repository リソースを作成、アクセス、または削除できるかどうかを決定します。これらのアクションを実行すると、AWS アカウントに追加料金が発生する可能性があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください。

- 最小特権を付与する - カスタムポリシーを作成するときは、タスクの実行に必要な許可のみを付与します。最小限の許可からスタートし、必要に応じて追加の許可を付与します。この方法は、寛容過ぎる許可から始めて、後から厳しくしようとするよりも安全です。詳細については、「IAM ユーザーガイド」の「Grant Least Privilege」(最小権限を付与する) を参照してください。
- 機密性の高いオペレーションに MFA を有効にする - 追加セキュリティとして、機密性の高いリソースまたは API オペレーションにアクセスするために IAM ユーザーに対して、多要素認証 (MFA) の使用を要求します。詳細については、IAM ユーザーガイドの「AWS での多要素認証 (MFA) の使用」を参照してください。
- 追加のセキュリティとしてポリシー条件を使用する - 実行可能な範囲内で、アイデンティティベースのポリシーがリソースへのアクセスを許可する条件を定義します。例えば、あるリクエストの送信が許可される IP アドレスの範囲を指定するための条件を記述できます。指定された日付または時間範囲内でのみリクエストを許可する条件を書くことも、SSL や MFA の使用を要求することもできます。詳細については、「IAM ユーザーガイド」の「IAM JSON Policy Elements: Condition」(IAM JSON ポリシー要素: 条件) を参照してください。

AWS Serverless Application Repository コンソールの使用

AWS Serverless Application Repository コンソールは、AWS Serverless Application Repository アプリケーションの発見および管理のための統合された環境を提供します。コンソールには、AWS Serverless Application RepositoryAPI のアクセス権限: アクションとリソースのリファレンス に説明されている API 固有のアクセス権限に加えて、AWS Serverless Application Repository アプリケーションの管理権限を必要とすることが多い機能やワークフローが多数あります。

AWS Serverless Application Repository コンソールを使用するために必要なアクセス許可の詳細については、「お客様が管理するポリシーの例」を参照してください。

ユーザーが自分のアクセス許可を表示できるようにする

この例では、ユーザーアイデンティティに添付されたインラインおよびマネージドポリシーの表示を IAM ユーザーに許可するポリシーを作成する方法を示します。このポリシーには、コンソールで、または AWS CLI か AWS API を使用してプログラム的に、このアクションを完了するアクセス許可が含まれています。

```
{
    "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": "*"
        }
    ]
}
```

お客様が管理するポリシーの例

このセクションの例では、ユーザーにアタッチできるサンプルポリシーのグループが用意されています。ポリシーの作成が初めての場合は、まずアカウントで IAM ユーザーを作成し、次にこのユーザーにポリシーをアタッチします。また、以下の例を使用して、複数のアクションを実行するためのアクセス許可を含む 1 つのカスタマイズされたポリシーを作成し、次にそのポリシーをユーザーにアタッチすることもできます。

ポリシーをユーザーに追加する方法については、IAM ユーザーガイドのユーザーへのアクセス許可の追加を参照してください。

例

- 発行者の例 1: 発行者にアプリケーションのリストを許可する
- 発行者の例 2: 発行者にアプリケーションまたはアプリケーションのバージョンの詳細の表示を許可する
- 発行者の例 3: 発行者にアプリケーションまたはアプリケーションのバージョンの作成を許可する
- 発行者の例 4: 他のユーザーとアプリケーションを共有するため発行者にアプリケーションポリシーの作成を許可する
- コンシューマーの例 1: コンシューマーにアプリケーションを検索することを許可する
- コンシューマーの例 2: コンシューマーにアプリケーションの詳細を表示することを許可する
- コンシューマーの例 3: コンシューマーにアプリケーションのデプロイを許可する
- コンシューマーの例 4: デプロイアセットへのアクセスを拒否する
- コンシューマーの例 5: コンシューマーによる公開アプリケーションの検索とデプロイを禁止する

発行者の例 1: 発行者にアプリケーションのリストを許可する

アカウントの IAM ユーザーには、`serverlessrepo:ListApplications` オペレーションへのアクセス許可が必要です。アクセス許可がないと、コンソールには何も表示されません。これらのアクセス許可を付与すると、ユーザーが属する特定の AWS リージョンで作成された AWS アカウントの AWS Serverless Application Repository アプリケーションがコンソールに一覧表示されます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListExistingApplications",
```

```
            "Effect": "Allow",
            "Action": [
                "serverlessrepo:ListApplications"
            ],
            "Resource": "*"
        }
    ]
}
```

発行者の例 2: 発行者にアプリケーションまたはアプリケーションのバージョンの詳細の表示を許可する

ユーザーは AWS Serverless Application Repository アプリケーションを選択し、アプリケーションの詳細を表示します。このような詳細には、作成者、説明、バージョン、およびその他の設定情報が含まれます。これを行うには、ユーザーに AWS Serverless Application Repository の `serverlessrepo:GetApplication` API オペレーションと`serverlessrepo:ListApplicationVersions` API オペレーションへのアクセス許可が必要です。

次の例では、これらのアクセス許可は、Amazon リソースネーム (ARN) が `Resource` 値として指定されている特定のアプリケーションに付与されます。

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

発行者の例 3: 発行者にアプリケーションまたはアプリケーションのバージョンの作成を許可する

AWS Serverless Application Repository アプリケーションを作成するためのアクセス権限をユーザーに許可する場合は、次のポリシーに示すように serverlessrepo:CreateApplication オペレーションと serverlessrepo:CreateApplicationVersions オペレーションへのアクセス許可を付与する必要があります。

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

発行者の例 4: 他のユーザーとアプリケーションを共有するため発行者にアプリケーションポリシーの作成を許可する

ユーザーが他のユーザーとアプリケーションを共有するには、次のポリシーに示すように、アプリケーションポリシーを作成するためのアクセス許可をユーザーに付与する必要があります。

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

```
}
```

コンシューマーの例 1: コンシューマーにアプリケーションを検索することを許可する

コンシューマーがアプリケーションを検索するには、次のアクセス権限を付与する必要があります。

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

コンシューマーの例 2: コンシューマーにアプリケーションの詳細を表示することを許可する

ユーザーは、AWS Serverless Application Repository アプリケーションを選択して、作成者、説明、バージョン、その他設定情報などのアプリケーションの詳細を表示できます。これを行うには、ユーザーには次の AWS Serverless Application Repository オペレーションのアクセス権限が必要です。

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

```
}
```

コンシューマーの例 3: コンシューマーにアプリケーションのデプロイを許可する

顧客がアプリケーションをデプロイするには、多くのオペレーションを実行するためのアクセス許可を付与する必要があります。次のポリシーは、顧客に必要な権限を提供します。

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

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

> **Note**
>
> アプリケーションのデプロイには、追加の AWS リソースを使用するためのアクセス権限が必要になる場合があります。AWS Serverless Application Repository は AWS CloudFormation と同じ基盤となるデプロイメカニズムを使用するため、詳細については、AWS Identity and Access Management によるアクセスの制御を参照してください。アクセス許可に関連するデプロイの問題については、「トラブルシューティング: IAM アクセス権限の不足」も参照してください。

コンシューマーの例 4: デプロイアセットへのアクセスを拒否する

アプリケーションを AWS アカウントと非公開共有している場合、デフォルトでは、そのアカウントのすべてのユーザーが同じアカウント内の他のすべてのユーザーのデプロイアセットにアクセス

できます。次のポリシーでは、アカウントのユーザーが AWS Serverless Application Repository の Amazon S3 バケットに保存されているデプロイアセットにアクセスすることを禁止します。

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

コンシューマーの例 5: コンシューマーによる公開アプリケーションの検索とデプロイを禁止する

ユーザーがアプリケーションに対して特定のアクションを実行できないようにすることができます。

次のポリシーは、`serverlessrepo:applicationType` を `public` に指定することで、公開アプリケーションに適用されます。これは、`Effect` を `Deny` 指定することで、ユーザーが多くのアクションを実行することを防ぎます。AWS Serverless Application Repository で使用できる条件キーの詳細については、「AWS Serverless Application Repository のアクション、リソース、および条件キー」を参照してください。

```
{
    "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**
>
> このポリシーステートメントは、サービスコントロールポリシーとして使用し、AWS Organization に適用することもできます。サービスコントロールポリシーの詳細については、AWS Organizations ユーザーガイドのサービスコントロールポリシーを参照してください。

AWS Serverless Application Repositoryアプリケーションポリシーの例

AWS Serverless Application Repositoryアプリケーションに添付された権限ポリシーは、アプリケーションポリシーと呼ばれます。アプリケーションポリシーは、指定されたプリンシパルまたはAWS Serverless Application Repository PrincipalOrgがアプリケーションで実行できるアクションを決定します。

AWS Serverless Application Repository アプリケーションは、AWS Serverless Application Repository のプライマリ AWS リソースです。AWS Serverless Application Repository アプリケーションポリシーは、アプリケーションのデプロイと、これらのアプリケーションの検索や詳細の表示などの関連操作に必要なアクセス許可をコンシューマーに付与するためにパブリッシャーが主に使用します。

パブリッシャーは、アプリケーションへのアクセス許可を次の 3 つのカテゴリに設定できます。

- 非公開 – 同じアカウントで作成され、他のアカウントと共有されていないアプリケーション。非公開アプリケーションをデプロイするアクセス許可を持つのは、AWS アカウントを共有するコンシューマーのみです。
- 非公開共有 - パブリッシャーが一連の特定の AWS アカウントまたは AWS Organization の AWS アカウントと明示的に共有しているアプリケーション。コンシューマーは、これらの AWS アカウントまたは AWS Organization と共有されているアプリケーションをデプロイするためのアクセ

ス許可を付与されます。AWS Organization の詳細については、AWS Organizations ユーザーガイドを参照してください。

- 公開共有 – パブリッシャーがすべてのユーザーと共有しているアプリケーション。すべてのコンシューマーは、すべての公開共有アプリケーションをデプロイするためのアクセス許可を付与されます。

> **Note**
>
> 非公開共有アプリケーションの場合、AWS Serverless Application Repository は AWS アカウントのみをプリンシパルとしてサポートします。パブリッシャーは、AWS Serverless Application Repository アプリケーションに対して AWS アカウント内のすべてのユーザーを 1 つのグループとして許可または拒否できます。パブリッシャーは、AWS Serverless Application Repository アプリケーションに対して AWS アカウント内の個別のユーザーを許可または拒否することはできません。

AWS Management Console を使用してアプリケーションへのアクセス許可を設定する手順については、「アプリケーションの共有」を参照してください。

AWS CLI を使用してアプリケーションへのアクセス許可を設定する手順と例については、次のセクションを参照してください。

アプリケーションへのアクセス許可 (AWS CLI および AWS SDK)

AWS CLI または AWS SDK を使用して AWS Serverless Application Repository アプリケーションへのアクセス許可を設定する場合は、次のアクションを指定できます。

アクション	説明
GetApplication	アプリケーションに関する情報を表示するためのアクセス許可を付与します。
CreateCloudFormationChangeSet	アプリケーションをデプロイするためのアクセス許可を付与します。 注意: このアクションではデプロイ以外の他のアクセス許可は付与されません。

アクション	説明
CreateCloudFormationTemplate	アプリケーションの AWS CloudFormation テンプレートを作成するアクセス許可を付与します。
ListApplicationVersions	アプリケーションのバージョンを一覧表示するためのアクセス許可を付与します。
ListApplicationDependencies	上位アプリケーションにネストされているリストアプリケーションを一覧表示するためのアクセス許可を付与します。
SearchApplications	アプリケーションを検索するためのアクセス許可を付与します。
デプロイ	このアクションにより、以上のすべてのアクションが有効になります。つまり、アプリケーションの表示、デプロイ、バージョンの一覧表示、および検索のアクセス許可が付与されます。

アプリケーションポリシーの例

以下の例では、AWS CLI を使用してアクセス許可を付与する方法を示します。AWS Management Console を使用してアクセス許可を付与する方法については、「アプリケーションの共有」を参照してください。

このセクションのすべての例では、以下の AWS CLI コマンドを使用して AWS Serverless Application Repository アプリケーションに関連付けられたアクセス許可ポリシーを管理します。

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

トピック

- 例 1: 別の特定のアカウントとアプリケーションを共有する
- 例 2: アプリケーションを公開共有する
- 例 3: アプリケーションを非公開にする
- 例 4: 複数のアカウントおよびアクセス許可の指定
- 例 5: AWS Organization のすべてのアカウントでアプリケーションを共有する
- 例 6: AWS Organization の一部のアカウントとアプリケーションを共有する

例 1: 別の特定のアカウントとアプリケーションを共有する

別の特定のアカウントとアプリケーションを共有し、他のアカウントとは共有しない場合は、共有する AWS アカウントの ID をプリンシパルとして指定します。この設定は、アプリケーションの非公開共有とも呼ばれます。これを行うには、次の AWS CLI コマンドを使用します。

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

> **Note**
>
> 非公開共有アプリケーションは、アプリケーションを作成した同じ AWS リージョンでのみ使用できます。

例 2: アプリケーションを公開共有する

アプリケーションを公開するには、次の例のように「*」をプリンシパルとして指定し、すべてのユーザーとアプリケーションを共有します。公開共有したアプリケーションは、すべてのリージョンで利用できます。

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

> **Note**
>
> アプリケーションを公開共有するには、SemanticVersion プロパティ LicenseUrl とプロパティの両方が設定されている必要があります。

例 3: アプリケーションを非公開にする

アプリケーションを非公開にして、誰とも共有せず、それを所有する AWS アカウントによってのみデプロイできます。そのためには、ポリシーからプリンシパルとアクションをクリアします。これにより、AWS Organization 内の他のアカウントのアクセス許可もアプリケーションのデプロイから削除されます。

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

> **Note**
>
> 非公開アプリケーションは、アプリケーションを作成した同じ AWS リージョンでのみ使用できます。

例 4: 複数のアカウントおよびアクセス許可の指定

複数のアクセス許可を付与できます。また、複数のアクセス許可を一度に複数の AWS アカウントに付与できます。これを行うには、次の例に示すように、プリンシパルおよびアクションとしてリストを指定します。

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

例 5: AWS Organization のすべてのアカウントでアプリケーションを共有する

アクセス許可は、AWS Organization 内のすべてのユーザーに付与できます。これを行うには、次の例のように、組織 ID を指定します。

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

AWS Organization の詳細については、AWS Organizations ユーザーガイドを参照してください。

> **Note**
>
> 指定できるのは、AWS アカウントがメンバーである AWS Organization のみです。メンバーではない AWS Organization を指定しようとすると、エラーが発生します。
> アプリケーションを AWS Organization と共有するには、将来共有を取り消す必要がある場合に備えて、UnshareApplication アクションのアクセス許可を含める必要があります。

例 6: AWS Organization の一部のアカウントとアプリケーションを共有する

アクセス許可は、AWS Organization 内の特定のアカウントに付与できます。これを行うには、次の例のように AWS アカウントのリストをプリンシパルとして指定し、組織 ID を指定します。

```
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**
>
> 指定できるのは、AWS アカウントがメンバーである AWS Organization のみです。メンバーではない AWS Organization を指定しようとすると、エラーが発生します。
> アプリケーションを AWS Organization と共有するには、将来共有を取り消す必要がある場合に備えて、UnshareApplication アクションのアクセス許可を含める必要があります。

例 7: アプリケーションポリシーを取得する

現在共有されているかどうかを確認するなど、アプリケーションの現在のポリシーを表示するには、次の例のように、get-application-policy コマンドを使用します。

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

例 8: アプリケーションをネストすることを特定のアカウントに許可する

公開アプリケーションのネストはすべてのユーザーに許可されます。アプリケーションをネストすることを特定のアカウントにのみ許可する場合は、次の例に示すように、以下の最小限のアクセス許可を設定する必要があります。

```
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 RepositoryAPI のアクセス権限: アクションとリソースのリファレンス

アクセスコントロールを設定し、IAM アイデンティティにアタッチできるアクセス許可ポリシー (アイデンティティベースのポリシー) を作成するときは、以下の表をリファレンスとして使用できます。このには、各 AWS Serverless Application Repository API オペレーション、アクションを実行するためのアクセス権限を付与できる対応アクション、およびアクセス権限を付与できる AWS リソースが含まれています。ポリシーの `Action` フィールドでアクションを指定し、ポリシーの `Resource` フィールドでリソースの値を指定します。

アクションを指定するには、API オペレーション名 (`serverlessrepo:ListApplications`など) の前に `serverlessrepo:` プレフィックスを使用します。

オペレーション	URI	方法	AWS リソース (ARN)
オペレーション: ListApplications 必要なアクセス許可 :serverlessrepo:ListApplications	/applications	GET	*
オペレーション: CreateApplication	/applications	POST	*

オペレーション	URI	方法	AWS リソース (ARN)
必要なアクセス許可 :serverlessrepo:CreateApplication			
オペレーション: GetApplication 必要なアクセス許可 :serverlessrepo:GetApplication	/applications/*application-id*	GET	arn:aws:serverlessrepo:*region*:*account-id* :applications/*application-name*
オペレーション: DeleteApplication 必要なアクセス許可 :serverlessrepo:DeleteApplication	/applications/*application-id*	DELETE	arn:aws:serverlessrepo:*region*:*account-id* :applications/*application-name*
オペレーション: UpdateApplication 必要なアクセス許可 :serverlessrepo:UpdateApplication	/applications/*application-id*	PATCH	arn:aws:serverlessrepo:*region*:*account-id* :applications/*application-name*
オペレーション: CreateCloudFormationChange設定 必要なアクセス許可 :serverlessrepo:CreateCloudFormationChange設定	/applications/*application-id*/changesets	POST	arn:aws:serverlessrepo:*region*:*account-id* :applications/*application-name*

オペレーション	URI	方法	AWS リソース (ARN)
オペレーション: GetApplicationポリシー 必要なアクセス許可 :serverlessrepo:GetApplicationポリシー	/applications/*application-id*/policy	GET	arn:aws:serverlessrepo:*region*:*account-id* :applications/*application-name*
オペレーション: PutApplicationポリシー 必要なアクセス許可 :serverlessrepo:PutApplicationポリシー	/applications/*application-id*/policy	PUT	arn:aws:serverlessrepo:*region*:*account-id* :applications/*application-name*
オペレーション: ListApplicationバージョン 必要なアクセス許可 :serverlessrepo:ListApplicationバージョン	/applications/*application-id*/versions	GET	arn:aws:serverlessrepo:*region*:*account-id* :applications/*application-name*
オペレーション: CreateApplicationVersion 必要なアクセス許可 :serverlessrepo:CreateApplicationVersion	/applications/*application-id* /versions/*semantic-version*	PUT	arn:aws:serverlessrepo:*region*:*account-id* :applications/*application-name*

オペレーション	URI	方法	AWS リソース (ARN)
オペレーション: ListApplicationの依存関係 必要なアクセス許可 :serverlessrepo:ListApplicationの依存関係	/applications/*application-id*/dependencies	GET	arn:aws:serverlessrepo:*region*:*account-id* :applications/*application-name*
オペレーション: SearchApplications 必要なアクセス許可 :serverlessrepo:SearchApplications	該当なし	該当なし	*

AWS Serverless Application Repository Identity and Access のトラブルシューティング

次の情報は、 および IAM の使用時に発生する可能性がある一般的な問題の診断 AWS Serverless Application Repository と修正に役立ちます。

トピック

- AWS Serverless Application Repositoryでのアクションの実行が承認されていない
- iam を実行する権限がありません。PassRole
- 管理者として へのアクセスを他のユーザーに許可したい AWS Serverless Application Repository
- 自分の AWS アカウント以外のユーザーに自分の AWS Serverless Application Repository リソースへのアクセスを許可したい

AWS Serverless Application Repositoryでのアクションの実行が承認されていない

がアクションを実行する権限がないと AWS Management Console 通知した場合は、管理者に連絡してサポートを依頼する必要があります。担当の管理者はお客様のユーザー名とパスワードを発行した人です。

次の例のエラーは、mateojackson IAM ユーザーがコンソールを使用してアプリケーションの詳細を表示する際に serverlessrepo:*GetApplication* アクセス許可を持っていないときに発生します。

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

この場合、Mateo は管理者に依頼し、serverlessrepo:*GetApplication* オペレーションを使用して *my-example-application* リソースにアクセスできるようにポリシーを更新してもらいます。

iam を実行する権限がありません。PassRole

iam:PassRole アクションを実行する権限がないというエラーが表示された場合は、ポリシーを更新して AWS Serverless Application Repositoryにロールを渡すことができるようにする必要があります。

一部の AWS のサービス では、新しいサービスロールまたはサービスにリンクされたロールを作成する代わりに、そのサービスに既存のロールを渡すことができます。そのためには、サービスにロールを渡す権限が必要です。

以下の例のエラーは、marymajor という IAM ユーザーがコンソールを使用して AWS Serverless Application Repositoryでアクションを実行しようする場合に発生します。ただし、このアクションをサービスが実行するには、サービスロールから付与された権限が必要です。メアリーには、ロールをサービスに渡す許可がありません。

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

この場合、Mary のポリシーを更新してメアリーに iam:PassRole アクションの実行を許可する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン認証情報を提供した担当者が管理者です。

管理者として へのアクセスを他のユーザーに許可したい AWS Serverless Application Repository

他のユーザーが にアクセスできるようにするには AWS Serverless Application Repository、アクセスが必要なユーザーまたはアプリケーションの IAM エンティティ (ユーザーまたはロール) を作成す

る必要があります。ユーザーは、このエンティティの認証情報を使用して AWSにアクセスします。次に、 AWS Serverless Application Repositoryの適切なアクセス許可を付与するポリシーを、そのエンティティにアタッチする必要があります。

すぐに開始するには、IAM ユーザーガイドの「IAM が委任した最初のユーザーおよびグループの作成」を参照してください。

自分の AWS アカウント以外のユーザーに自分の AWS Serverless Application Repository リソースへのアクセスを許可したい

他のアカウントのユーザーや組織外の人が、リソースにアクセスするために使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまたはアクセスコントロールリスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください:

- がこれらの機能 AWS Serverless Application Repository をサポートしているかどうかを確認するには、「」を参照してくださいAWS Serverless Application Repository で IAM を使用する方法。
- 所有 AWS アカウント している のリソースへのアクセスを提供する方法については、IAM ユーザーガイドの「所有 AWS アカウント している別の の IAM ユーザーへのアクセスを提供する」を参照してください。
- リソースへのアクセスをサードパーティー に提供する方法については AWS アカウント、IAM ユーザーガイドの「サードパーティー AWS アカウント が所有する へのアクセスを提供する」を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、IAM ユーザーガイドの外部で認証されたユーザー (ID フェデレーション) へのアクセスの許可を参照してください。
- クロスアカウントアクセスでのロールとリソースベースのポリシーの使用の違いについては、IAM ユーザーガイドの「IAM でのクロスアカウントリソースアクセス」を参照してください。

AWS Serverless Application Repository でのログ記録とモニタリング

モニタリングは、AWS ソリューションの信頼性、可用性、パフォーマンスを維持するうえで重要な部分です。マルチポイント障害が発生した場合は、その障害をデバッグしやすくするために、AWS ソリューションのすべての部分からモニタリングデータを収集する必要があります。AWS

には、AWS Serverless Application Repository リソースをモニタリングし、潜在的なインシデントに対応するための以下のような複数のツールが用意されています。

AWS CloudTrail ログ

AWS Serverless Application Repository は、ユーザー、ロール、または AWS Serverless Application Repository の AWS サービスによって実行されたアクションを記録するサービスである AWS CloudTrail と統合されています。CloudTrailのすべての API コールをキャプチャします。AWS Serverless Application Repositoryイベントとして。

トピックス

AWS CloudTrail を使用した AWS Serverless Application Repository API コールのログ記録

AWS Serverless Application Repository は、ユーザー、ロール、または AWS Serverless Application Repository の AWS サービスによって実行されたアクションを記録するサービスである AWS CloudTrail と統合されています。CloudTrailのすべての API コールをキャプチャします。AWS Serverless Application Repositoryイベントとして。キャプチャされた呼び出しには、AWS Serverless Application Repository コンソールの呼び出しと、AWS Serverless Application Repository API オペレーションへのコード呼び出しが含まれます。

証跡を作成すると、の継続的デリバリーが可能になります。CloudTrailAmazon S3 バケットへのイベント (のイベントを含む) AWS Serverless Application Repository。証跡を設定しない場合でも、CloudTrail コンソールの [Event history (イベント履歴)] で最新のイベントを表示できます。

CloudTrail で収集された情報に基づいて、AWS Serverless Application Repository に対して行われたリクエストを確認できます。リクエスト元の IP アドレス、リクエスト者、リクエスト日時、および追加の詳細を確認することもできます。

CloudTrail の詳細については、「AWS CloudTrail ユーザーガイド」を参照してください。

CloudTrail 内の AWS Serverless Application Repository 情報

CloudTrail は、アカウント作成時に AWS アカウントで有効になります。AWS Serverless Application Repository でアクティビティが発生すると、そのアクティビティは CloudTrail イベントとして他の AWS のサービスのイベントとともに、[Event history] に記録されます。最近のイベント

は、AWS アカウントで表示、検索、ダウンロードできます。詳細については、「CloudTrail イベント履歴でのイベントの表示」を参照してください。

AWS アカウントのイベント (AWS Serverless Application Repository のイベントなど) を継続的に記録するには、証跡を作成します。あるトレイル可能にするCloudTrailAmazon S3 バケットにログファイルを渡します。デフォルトでは、コンソールで追跡を作成するときに、追跡がすべてのAWSリージョンに適用されます。追跡では、AWS パーティション内のすべての AWS リージョンからのイベントがログに記録され、指定した Amazon S3 バケットにログファイルが配信されます。さらに、より詳細な分析と CloudTrail ログで収集されたデータに基づいた行動のためにその他の AWS サービスを設定できます。詳細については、以下を参照してください:

- 追跡を作成するための概要
- CloudTrail でサポートされるサービスと統合
- CloudTrail の Amazon SNS 通知の設定
- 「複数のリージョンから CloudTrail ログファイルを受け取る」と「複数のアカウントから CloudTrail ログファイルを受け取る」

すべての AWS Serverless Application Repository アクションは CloudTrail によって記録されます。これらのアクションは、[AWS Serverless Application Repository リソース] ページに説明されています。たとえば、`CreateApplication`、`UpdateApplications`、`ListApplications` のオペレーションを呼び出すと、CloudTrail ログファイルにエントリが生成されます。

各イベントまたはログエントリには、リクエストの生成者に関する情報が含まれます。同一性情報は以下の判断に役立ちます:

- リクエストが、ルート認証情報と AWS Identity and Access Management (IAM) ユーザー認証情報のどちらを使用して送信されたか.
- リクエストが、ロールとフェデレーティッドユーザーの一時的なセキュリティ認証情報で行われたか。
- リクエストが、別の AWS サービスによって送信されたかどうか。

詳細については、「CloudTrail userIdentity 要素」を参照してください。

AWS Serverless Application Repository ログファイルエントリの概要

追跡は、指定したAmazon S3バケットにイベントをログファイルとして配信するように設定できるものです。CloudTrail ログファイルには、1 つ以上のログエントリが含まれます。イベントはあらゆ

るソースからの単一のリクエストを表し、リクエストされたアクション、アクションの日時、リクエストのパラメータなどの情報が含まれます。CloudTrail ログファイルは、パブリック API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

以下の例は、CreateApplication アクションの CloudTrail ログエントリを示しています。

```
{
    "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"
}
```

AWS Serverless Application Repository のコンプライアンス検証

サードパーティーの監査人は、さまざまな AWS コンプライアンスプログラムの一環として、AWS Serverless Application Repository のセキュリティとコンプライアンスを評価します。これらのプログラムには、SOC、PCI、FedRAMP などがあります。

特定のコンプライアンスプログラムの対象となる AWS サービスのリストについては、コンプライアンスプログラムによる AWS 対象範囲内のサービスを参照してください。一般的な情報については、AWS コンプライアンスプログラムを参照してください。

サードパーティーの監査レポートをダウンロードするには、AWS Artifact を使用します。詳細については、AWS Artifact のレポートのダウンロード を参照してください。

AWS Serverless Application Repository を使用する際のお客様のコンプライアンス責任は、データの機密性、企業のコンプライアンス目的、適用法規によって決まります。AWS ではコンプライアンスに役立つ以下のリソースを用意しています。

- セキュリティおよびコンプライアンスのクイックスタートガイド – これらのデプロイメントガイドでは、アーキテクチャ上の考慮事項について説明し、セキュリティとコンプライアンスに重点を置いたベースライン環境を AWS にデプロイするための手順を示します。
- AWS コンプライアンスのリソース - このワークブックとガイドのコレクションは、お客様の業界や所在地に適用される場合があります。
- AWS Config - この AWS サービスは、お客様のリソース構成が、社内の慣行、業界ガイドラインおよび規制にどの程度準拠しているかを評価するものです。
- AWS Security Hub - この AWS サービスは、セキュリティ業界の標準やベストプラクティスに準拠しているかどうかを確認するために、AWS 内のセキュリティ状態を包括的に表示するものです。

AWS Serverless Application Repository の耐障害性

AWS のグローバルインフラストラクチャは AWS リージョンとアベイラビリティーゾーンを中心に構築されます。AWSリージョンには、低レイテンシー、高いスループット、そして高度の冗長ネットワークで接続されている複数の物理的に独立・隔離されたアベイラビリティーゾーンがあります。アベイラビリティーゾーンでは、アベイラビリティーゾーン間で中断せずに、自動的にフェイルオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性、耐障害性、および拡張性が優れています。

AWS リージョンとアベイラビリティーゾーンの詳細については、[AWS Global Infrastructure] (グローバルインフラストラクチャ) を参照してください。

AWS Serverless Application Repository のインフラストラクチャセキュリティ

マネージドサービスである AWS Serverless Application Repository は AWS グローバルネットワークセキュリティで保護されています。AWSセキュリティサービスと AWS がインフラストラクチャを保護する方法については、「AWS クラウドセキュリティ」を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して AWS 環境を設計するには、「セキュリティの柱 - AWS Well-Architected Framework」の「インフラストラクチャ保護」を参照してください。

AWS の発行済み API コールを使用して、ネットワーク経由で AWS Serverless Application Repository にアクセスします。クライアントは以下をサポートする必要があります。

- Transport Layer Security (TLS) TLS 1.2 および TLS 1.3 をお勧めします。
- DHE (Ephemeral Diffie-Hellman) や ECDHE (Elliptic Curve Ephemeral Diffie-Hellman) などの Perfect Forward Secrecy (PFS) を使用した暗号スイートです。これらのモードは、Java 7 以降など、最近のほとんどのシステムでサポートされています。

また、リクエストは、アクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または、AWS Security Token Service (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

AWS Serverless Application Repository のクォータ

AWS Serverless Application Repository には、各 AWS リージョンで AWS アカウントが持つことができる公開アプリケーションの数に関するクォータがあります。このクォータはリージョンごとに適用され、引き上げをリクエストできます。引き上げをリクエストするには、サポートセンターコンソールを使用してください。

リソース	デフォルトのクォータ
パブリックアプリケーション (各 AWS リージョンの AWS アカウントあたり)	100

次のクォータは、コードパッケージとアプリケーションポリシーで使用できるストレージに適用されます。これらのクォータは変更できません。

リソース	クォータ
コードパッケージ用の無料の Amazon S3 ストレージ (各 AWS リージョンの AWS アカウントあたり)	5 GB
アプリケーションポリシーの長さ	6,144 文字

AWS Serverless Application Repositoryのトラブルシューティング

AWS Serverless Application Repository を使用して、アプリケーションを作成、更新、または削除する際に、問題が発生することがあります。このセクションを使用して、一般的な問題が生じた場合のトラブルシューティングをすることができます。また、AWS Serverless Application Repository フォーラムで回答を検索したり、質問を投稿したりすることもできます。

> **Note**
>
> AWS Serverless Application Repository のアプリケーションは AWS CloudFormation を使用してデプロイされます。AWS CloudFormation の問題のトラブルシューティングの詳細については、AWS CloudFormation トラブルシューティングガイドを参照してください。

トピック

アプリケーションを公開することはできません

アプリケーションを公開することができない場合は、Open Source Initiative (OSI) によって承認されたアプリケーションのライセンスファイルがない可能性があります。

アプリケーションを公開するには、OSI 公認のライセンスファイルと、そのバージョンのソースコード URL を使用して正常に発行されたアプリケーションのバージョンが必要です。アプリケーションを作成するとアプリケーションのライセンスを更新することはできません。

ライセンスファイルがないためにアプリケーションを公開できない場合は、アプリケーションを削除して、同じ名前の新しいファイルを作成します。Open Source Initiative (OSI) 組織が承認した 1 つ以上のオープンソースライセンスを使用して提供していることを確認してください。

クォータを超過しました

クォータを超えたことを示すエラーメッセージが表示された場合は、リソースのクォータに達したかどうかを確認します。AWS Serverless Application Repository のクォータについては、「AWS Serverless Application Repository のクォータ」を参照してください。

更新された Readme ファイルがすぐに表示されません

アプリケーションを公開すると、アプリケーションの内容が更新されるまでに最大 24 時間かかることがあります。24 時間以上の遅延が発生した場合は、AWS サポートにお問い合わせください。詳細については、以下を参照してください。

IAM アクセス権限の不足のためアプリケーションをデプロイできません

AWS Serverless Application Repository アプリケーションをデプロイするには、AWS Serverless Application Repository リソースおよび AWS CloudFormation スタックに対するアクセス権限が必要です。アプリケーションで説明されている基盤となるサービスを使用するためのアクセス許可が必要になる場合もあります。例えば、Amazon S3 バケットや Amazon DynamoDB テーブルを作成する場合には、Amazon S3 または DynamoDB に対するアクセス許可が必要となります。

このタイプの問題が発生した場合は、AWS Identity and Access Management (IAM) ポリシーを確認して、必要なアクセス権限があることを確認します。詳細については、AWS Identity and Access Management によるアクセスの制御をご参照ください。

同じアプリケーションを 2 回デプロイすることができません

指定したアプリケーション名は、AWS CloudFormation スタックの名前として使用されます。アプリケーションのデプロイに問題がある場合は、同じ名前の既存の AWS CloudFormation スタックがないことを確認します。その場合は、別のアプリケーション名を指定するか、既存のスタックを削除して、同じ名前のアプリケーションをデプロイします。

アプリケーションが公開されていない理由

アプリケーションは、デフォルトではプライベートです。アプリケーションを公開するには、こちらの手順に従います。

Support へのお問い合わせ

問題によっては、このセクションや AWS Serverless Application Repository のフォーラムでトラブルシューティングの解決策が見つからないことがあります。AWS Premium Support を契約している場合は、AWS サポートで技術サポートケースを作成できます。

AWS サポートに連絡する前に、質問があるアプリケーションの Amazon リソースネーム (ARN) を取得してください。アプリケーションの ARN は、AWS Serverless Application Repository コンソールで確認できます。

操作

AWS Serverless Application Repository REST API には、以下のオペレーションが含まれます。

- CreateApplication

  アプリケーションを作成します。オプションでAWSSAM ファイルを指定し、最初のアプリケーションバージョンを同じコールで作成します。
- CreateApplicationVersion

  アプリケーションバージョンを作成します。
- CreateCloudFormationChangeSet

  を作成します。AWS CloudFormation指定されたアプリケーションのセットを変更します。
- CreateCloudFormationTemplate

  を作成します。AWS CloudFormationテンプレート。
- DeleteApplication

  指定されたアプリケーションを削除します。
- GetApplication

  指定されたアプリケーションを入手します。
- GetApplicationPolicy

  アプリケーションのポリシーを取得します。
- GetCloudFormationTemplate

  指定された値を取得します。AWS CloudFormationテンプレート。
- ListApplicationDependencies

  包含するアプリケーションにネストされたアプリケーションのリストを取得します。
- ListApplications

  リクエスタが所有しているアプリケーションを一覧表示します。
- ListApplicationVersions

  指定されたアプリケーションのバージョンを一覧表示します。

- PutApplicationPolicy

  アプリケーションのアクセス許可ポリシーを設定します。このオペレーションでサポートされているアクションの詳細については、アプリケーションへのアクセス許可を参照してください。

- UnshareApplication

  AWS Organization からアプリケーションを共有解除します。

  このオペレーションは、組織の管理アカウントからのみ呼び出すことができます。

- UpdateApplication

  指定されたアプリケーションを更新します。

リソース

AWS Serverless Application Repository REST API には、以下のリソースが含まれています。

トピック

Applications

[URI]

`/applications`

HTTP メソッド

GET

オペレーション ID: `ListApplications`

リクエスタが所有しているアプリケーションを一覧表示します。

クエリパラメータ

名前	型	必須	説明
`maxItems`	文字列	False	返される項目の合計数。

名前	型	必須	説明
nextToken	文字列	False	ページ分割を始める場所を指定するトークン。

レスポンス

ステータスコード	レスポンスモデル	説明
200	ApplicationPage	成功
400	BadRequestException	リクエストに含まれているパラメータの 1 つが無効です。
403	ForbiddenException	クライアントは認証されていません。
404	NotFoundException	リクエストで指定されたリソース (例えば、アクセスポリシーステートメント) は存在しません。
500	InternalServerErrorException	AWS Serverless Application Repository サービスで内部エラーが発生しました。

POST

オペレーション ID: CreateApplication

アプリケーションを作成し、オプションで AWS SAM ファイルを含めて、同じ呼び出しで最初のアプリケーションバージョンを作成します。

レスポンス

ステータスコード	レスポンスモデル	説明
201	Application	成功

ステータスコード	レスポンスモデル	説明
400	BadRequestException	リクエストに含まれているパラメータの 1 つが無効です。
403	ForbiddenException	クライアントは認証されていません。
409	ConflictException	リソースは既に存在します。
429	TooManyRequestsException	クライアントが、単位時間あたりの許可されるリクエスト数よりも多くのリクエストを送信しています。
500	InternalServerErrorException	AWS Serverless Application Repository サービスで内部エラーが発生しました。

OPTIONS

レスポンス

ステータスコード	レスポンスモデル	説明
200	なし	200 レスポンス

スキーマ

リクエストボディ

POST スキーマ

```
{
  "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",
  "sourceCodeArchiveUrl": "string"
}
```

レスポンス本文

ApplicationPage スキーマ

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

Application スキーマ

```
{
  "applicationId": "string",
  "name": "string",
  "description": "string",
```

```
  "author": "string",
  "isVerifiedAuthor": boolean,
  "verifiedAuthorUrl": "string",
  "spdxLicenseId": "string",
  "licenseUrl": "string",
  "readmeUrl": "string",
  "labels": [
    "string"
  ],
  "creationTime": "string",
  "homePageUrl": "string",
  "version": {
    "applicationId": "string",
    "semanticVersion": "string",
    "sourceCodeUrl": "string",
    "sourceCodeArchiveUrl": "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"
        ]
      }
    ],
    "requiredCapabilities": [
      enum
    ],
    "resourcesSupported": boolean
  }
```

```
}
```

BadRequestException スキーマ

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

ForbiddenException スキーマ

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

NotFoundException スキーマ

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

ConflictException スキーマ

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

TooManyRequestsException スキーマ

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

InternalServerErrorException スキーマ

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

プロパティ

Application

アプリケーションに関する詳細。

applicationId

アプリケーションの Amazon リソースネーム (ARN)。

タイプ：文字列
必須: True

name

アプリケーションの名前。

最小長: 1 最大長 = 140。

パターン: "[a-zA-Z0-9\\-]+";

タイプ：文字列
必須: True

description

アプリケーションの説明。

最小長: 1 最大長 = 256。

タイプ：文字列
必須: True

author

アプリケーションを公開する作成者の名前。

最小長: 1 最大長 = 127。

パターン: "^[a-z0-9](([a-z0-9]|-(?!-))*[a-z0-9])?$";

タイプ：文字列
必須: True

isVerifiedAuthor

このアプリケーションの作成者が検証されているかどうかを指定します。つまり、 AWS は、合理的かつ慎重なサービスプロバイダーとして、リクエスタから提供された情報を善意で確認し、リクエスタのアイデンティティがクレーム済みであることを確認しています。

タイプ：ブール値
必須: False

verifiedAuthorUrl

検証済み作成者のパブリックプロファイルへの URL。この URL は作成者によって提出されます。

タイプ：文字列
必須: False

spdxLicenseId

https://spdx.org/licenses/ からの有効な識別子。

タイプ：文字列
必須: False

licenseUrl

アプリケーションの spdxLicenseID 値に一致するアプリケーションのライセンスファイルへのリンク。

最大サイズ: 5 MB。

タイプ：文字列
必須: False

readmeUrl

Markdown 言語の readme ファイルへのリンク。アプリケーションとその動作に関する詳細な説明が含まれます。

最大サイズ: 5 MB。

タイプ：文字列
必須: False

labels

検索結果でアプリケーションを発見しやすくするラベル。

最小長: 1 最大長 = 127。ラベルの最大数: 10。

パターン: "^[a-zA-Z0-9+\\-_:\\/@]+$";

タイプ: string タイプの配列
必須: False

creationTime

このリソースが作成された日時。

タイプ：文字列
必須: False

homePageUrl

アプリケーションの GitHub リポジトリの場所など、アプリケーションに関する詳細情報を含む URL。

タイプ: 文字列
必須: False

version

アプリケーションに関するバージョン情報。

タイプ: バージョン
必須: False

ApplicationPage

アプリケーションの詳細のリスト。

applications

アプリケーション概要の配列。

タイプ: ApplicationSummary タイプの配列
必須: True

nextToken

次の結果ページを要求するためのトークン。

タイプ：文字列
必須: False

ApplicationSummary

アプリケーションに関する詳細の概要。

applicationId

アプリケーションの Amazon リソースネーム (ARN)。

タイプ：文字列
必須: True

name

アプリケーションの名前。

最小長: 1 最大長 = 140。

パターン: "[a-zA-Z0-9\\-]+";

タイプ：文字列
必須: True

description

アプリケーションの説明。

最小長: 1 最大長 = 256。

タイプ：文字列
必須: True

author

アプリケーションを公開する作成者の名前。

最小長: 1 最大長 = 127。

パターン: "^[a-z0-9](([a-z0-9]|-(?!-))*[a-z0-9])?$";

タイプ：文字列
必須: True

spdxLicenseId

https://spdx.org/licenses/ からの有効な識別子。

タイプ：文字列
必須: False

labels

検索結果でアプリケーションを発見しやすくするラベル。

最小長: 1 最大長 = 127。ラベルの最大数: 10。

パターン: "^[a-zA-Z0-9+\\-_:\\/@]+$";

タイプ: string タイプの配列
必須: False

creationTime

このリソースが作成された日時。

タイプ：文字列
必須: False

homePageUrl

アプリケーションの GitHub リポジトリの場所など、アプリケーションに関する詳細情報を含む URL。

タイプ: 文字列
必須: False

BadRequestException

リクエストに含まれているパラメータの 1 つが無効です。

message

リクエストに含まれているパラメータの 1 つが無効です。

タイプ：文字列
必須: False

errorCode

400

タイプ：文字列
必須: False

Capability

一部のアプリケーションをデプロイするために指定する必要がある値。

CAPABILITY_IAM
CAPABILITY_NAMED_IAM
CAPABILITY_AUTO_EXPAND
CAPABILITY_RESOURCE_POLICY

ConflictException

リソースは既に存在します。

message

リソースは既に存在します。

タイプ：文字列
必須: False

errorCode

409

タイプ：文字列
必須: False

CreateApplicationInput

アプリケーションリクエストを作成します。

name

公開するアプリケーションの名前。

最小長: 1 最大長 = 140。

パターン: "[a-zA-Z0-9\\-]+";

タイプ：文字列
必須: True

description

アプリケーションの説明。

最小長: 1 最大長 = 256。

タイプ：文字列
必須: True

author

アプリケーションを公開する作成者の名前。

最小長: 1 最大長 = 127。

パターン: "^[a-z0-9](([a-z0-9]|-(?!-))*[a-z0-9])?$";

タイプ：文字列
必須: True

spdxLicenseId

https://spdx.org/licenses/ からの有効な識別子。

タイプ：文字列
必須: False

licenseBody

アプリケーションの spdxLicenseID 値に一致するアプリケーションのライセンスを含むローカルテキストファイル。ファイルの形式は `file://<path>/<filename>` です。

最大サイズ: 5 MB。

指定できるのは、`licenseBody` か `licenseUrl` のいずれかです。それ以外の場合は、エラーが発生します。

タイプ：文字列
必須: False

licenseUrl

アプリケーションの spdxLicenseID 値に一致するアプリケーションのライセンスを含む S3 オブジェクトへのリンク。

最大サイズ: 5 MB。

指定できるのは、`licenseBody` か `licenseUrl` のいずれかです。それ以外の場合は、エラーが発生します。

タイプ：文字列
必須: False

readmeBody

Markdown 言語のローカルテキスト readme ファイル。アプリケーションとその動作に関する詳細な説明が含まれます。ファイルの形式は `file://<path>/<filename>` です。

最大サイズ: 5 MB。

指定できるのは、`readmeBody` か `readmeUrl` のいずれかです。それ以外の場合は、エラーが発生します。

タイプ：文字列
必須: False

readmeUrl

Markdown 言語の S3 オブジェクトへのリンク。アプリケーションとその動作に関する詳細な説明が含まれます。

最大サイズ: 5 MB。

指定できるのは、`readmeBody` か `readmeUrl` のいずれかです。それ以外の場合は、エラーが発生します。

タイプ：文字列
必須: False

labels

検索結果でアプリケーションを発見しやすくするラベル。

最小長: 1 最大長 = 127。ラベルの最大数: 10。

パターン: "^[a-zA-Z0-9+\\-_:\\/@]+$";

タイプ: string タイプの配列
必須: False

homePageUrl

アプリケーションの GitHub リポジトリの場所など、アプリケーションに関する詳細情報を含む URL。

タイプ: 文字列
必須: False

semanticVersion

アプリケーションのセマンティックバージョン：

https://semver.org/

タイプ：文字列
必須: False

templateBody

アプリケーションのローカル raw パッケージ AWS SAM テンプレートファイル。ファイルの形式は `file://<path>/<filename>` です。

指定できるのは、`templateBody` か `templateUrl` のいずれかです。それ以外の場合は、エラーが発生します。

タイプ：文字列
必須: False

templateUrl

アプリケーションのパッケージ化された AWS SAM テンプレートを含む S3 オブジェクトへのリンク。

指定できるのは、`templateBody` か `templateUrl` のいずれかです。それ以外の場合は、エラーが発生します。

タイプ：文字列
必須: False

sourceCodeUrl

特定の GitHub コミットの URL など、アプリケーションのソースコードのパブリックリポジトリへのリンク。

タイプ: 文字列
必須: False

sourceCodeArchiveUrl

アプリケーションのこのバージョンのソースコードの ZIP アーカイブを含む S3 オブジェクトへのリンク。

最大サイズ: 50 MB。

タイプ：文字列
必須: False

ForbiddenException

クライアントは認証されていません。

message

クライアントは認証されていません。

タイプ：文字列
必須: False

errorCode

403

タイプ：文字列
必須: False

InternalServerErrorException

AWS Serverless Application Repository サービスで内部エラーが発生しました。

message

AWS Serverless Application Repository サービスで内部エラーが発生しました。

タイプ: 文字列
必須: False

errorCode

500

タイプ：文字列
必須: False

NotFoundException

リクエストで指定されたリソース (例えば、アクセスポリシーステートメント) は存在しません。

message

リクエストで指定されたリソース (例えば、アクセスポリシーステートメント) は存在しません。

タイプ：文字列
必須: False

errorCode

404

タイプ：文字列

必須: False

ParameterDefinition

アプリケーションでサポートされるパラメータ。

name

パラメータの名前。

タイプ：文字列
必須: True

defaultValue

スタックの作成時に値を指定しなかった場合に、テンプレートで使用される適切な型の値。パラメーターの制約を定義する場合は、これらの制約に従う値を指定する必要があります。

タイプ：文字列
必須: False

description

パラメータについて説明する最大 4000 文字の文字列。

タイプ：文字列
必須: False

type

パラメータのタイプ。

有効な値: `String | Number | List<Number> | CommaDelimitedList`

`String`: リテラル文字列。

例えば、`"MyUserName"` と指定することができます。

`Number`: 整数または float. AWS CloudFormation valid は、パラメータ値を数値として検証します。ただし、テンプレート内の他の場所で使用した場合には (`Ref` 組み込み関数を使用した場合など) 文字列として扱います。

例えば、"8888" のように指定することがができます。

List<Number>: カンマで区切られた整数または浮動小数値の配列。 AWS CloudFormation はパラメータ値を数値として検証します。ただし、テンプレート内の他の場所で使用した場合には (Ref 組み込み関数を使用した場合など) 文字列のリストとして扱います。

例えば、「80,20」を指定すると、Ref は ["80","20"] になります。

CommaDelimitedList: カンマで区切られたリテラル文字列の配列。文字列の合計数は、カンマの合計数よりも 1 つ多いはずです。また、各メンバー文字列の前後の空白は削除されます。

例えば、「test,dev,prod」を指定すると、Ref は ["test","dev","prod"] になります。

タイプ：文字列
必須: False

noEcho

スタックの詳細を取得する呼び出しが他のユーザーによって作成された場合に、必ずパラメータ値をマスクするかどうか。値を true に設定すると、パラメータ値はアスタリスク (*****) でマスクされます。

タイプ：ブール値
必須: False

allowedPattern

String 型に使用できるパターンを表す正規表現。

タイプ：文字列
必須: False

constraintDescription

制約が違反された場合に、制約について説明する文字列。たとえば、制約の説明を指定しないとき、許容されているパターンが [A-Za-z0-9]+ であるパラメーターの場合、ユーザーが無効な値を指定すると次のエラーメッセージが表示されます。

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

「must contain only uppercase and lowercase letters and numbers」などの制約の説明を追加することによって、次のようにカスタマイズされたエラーメッセージを表示することができます。

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

タイプ：文字列
必須: False

minValue

`Number` タイプに使用できる数値の最小値を決定する数値。

タイプ: 整数
必須: False

maxValue

`Number` タイプに使用できる数値の最大値を決定する数値。

タイプ: 整数
必須: False

minLength

`String` タイプに使用できる最小文字数を決定する整数値。

タイプ: 整数
必須: False

maxLength

`String` タイプに使用できる最大文字数を決定する整数値。

タイプ: 整数
必須: False

allowedValues

パラメーターに許容される一連の値を含む配列。

タイプ: string タイプの配列
必須: False

referencedByResources

このパラメータを使用する AWS SAM リソースのリスト。

タイプ: string タイプの配列
必須: True

TooManyRequestsException

クライアントが、単位時間あたりの許可されるリクエスト数よりも多くのリクエストを送信しています。

message

クライアントが、単位時間あたりの許可されるリクエスト数よりも多くのリクエストを送信しています。

タイプ：文字列
必須: False

errorCode

429

タイプ：文字列
必須: False

Version

アプリケーションのバージョンの詳細。

applicationId

アプリケーションの Amazon リソースネーム (ARN)。

タイプ：文字列
必須: True

semanticVersion

アプリケーションのセマンティックバージョン：

https://semver.org/

タイプ：文字列
必須: True

sourceCodeUrl

特定の GitHub コミットの URL など、アプリケーションのソースコードのパブリックリポジトリへのリンク。

タイプ: 文字列
必須: False

sourceCodeArchiveUrl

アプリケーションのこのバージョンのソースコードの ZIP アーカイブを含む S3 オブジェクトへのリンク。

最大サイズ: 50 MB。

タイプ：文字列
必須: False

templateUrl

アプリケーションのパッケージ化された AWS SAM テンプレートへのリンク。

タイプ: 文字列

必須: True

creationTime

このリソースが作成された日時。

タイプ：文字列
必須: True

parameterDefinitions

アプリケーションでサポートされるパラメータタイプの配列。

タイプ: ParameterDefinition タイプの配列
必須: True

requiredCapabilities

特定のアプリケーションをデプロイする前に指定する必要がある値のリスト。一部のアプリケーションには、新しい AWS Identity and Access Management （ IAM) ユーザーを作成するなど、 AWS アカウントのアクセス許可に影響を与える可能性のあるリソースが含まれている場合があります。このようなアプリケーションの場合は、このパラメータを指定して、それらの機能を明示的に認識する必要があります。

有効な値は、CAPABILITY_IAM、CAPABILITY_NAMED_IAM、CAPABILITY_RESOURCE_POLICY、および CAPABILITY_AUTO_EXPAND のみです。

次のリソースでは、 CAPABILITY_IAMまたは を指定する必要がありますCAPABILITY_NAMED_IAM: AWS::IAM::Group、AWS::IAM::InstanceProfile、AWS::IAM::Policy、および AWS::IAM::Role。アプリケーションに IAM リソースがある場合、CAPABILITY_IAM または CAPABILITY_NAMED_IAM のいずれかを指定できます。アプリケーションにカスタム名を持つ IAM リソースがある場合は、CAPABILITY_NAMED_IAM を指定する必要があります。

次のリソースではCAPABILITY_RESOURCE_POLICYAWS::Lambda::Permission、、AWS::IAM:Policy、、AWS::ApplicationAを指定する必要がありますAWS::SQS::QueuePolicyAWS::SNS::TopicPolicy。

1 つまたは複数のネストされたアプリケーションが含まれているアプリケーションでは、CAPABILITY_AUTO_EXPAND を指定する必要があります。

アプリケーションテンプレートに前述のリソースが含まれている場合、デプロイする前にアプリケーションに関連付けられたすべてのアクセス許可を確認することをお勧めします。機能を必要とするアプリケーションにこのパラメータを指定しないと、呼び出しは失敗します。

タイプ: Capability タイプの配列
必須: True

resourcesSupported

このアプリケーションに含まれるすべての AWS リソースが、取得するリージョンでサポートされているかどうか。

タイプ：ブール値
必須: True

以下も参照してください。

言語固有の AWS SDKs とリファレンスのいずれかでこの API を使用する方法の詳細については、以下を参照してください。

ListApplications

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby v3

CreateApplication

- AWS コマンドラインインターフェイス
- AWS SDK for .NET

- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby v3

Applications applicationId

[URI]

`/applications/applicationId`

HTTP メソッド

GET

オペレーション ID: `GetApplication`

指定されたアプリケーションを入手します。

パスパラメータ

名前	型	必須	説明
applicationId	文字列	True	アプリケーションの Amazon リソースネーム (ARN) です。

クエリパラメータ

名前	型	必須	説明
semanticVersion	文字列	False	取得するアプリケーションのセマンティックバージョン。

レスポンス

ステータスコード	レスポンスモデル	説明
200	Application	成功
400	BadRequestException	リクエストに含まれているパラメータの 1 つが無効です。
403	ForbiddenException	クライアントは認証されていません。
404	NotFoundException	リクエストで指定されたリソース (例えば、アクセスポリシーステートメント) は存在しません。
429	TooManyRequestsException	クライアントが、単位時間あたりの許可されるリクエスト数よりも多くのリクエストを送信しています。
500	InternalServerErrorException	AWS Serverless Application Repository サービスで内部エラーが発生しました。

DELETE

オペレーション ID: `DeleteApplication`

指定されたアプリケーションを削除します。

パスパラメータ

名前	型	必須	説明
applicationId	文字列	True	アプリケーションの Amazon リソースネーム (ARN) です。

レスポンス

ステータスコード	レスポンスモデル	説明
204	なし	成功
400	BadRequestException	リクエストに含まれているパラメータの 1 つが無効です。
403	ForbiddenException	クライアントは認証されていません。
404	NotFoundException	リクエストで指定されたリソース (例えば、アクセスポリシーステートメント) は存在しません。
409	ConflictException	リソースは既に存在します。
429	TooManyRequestsException	クライアントが、単位時間あたりの許可されるリクエスト数よりも多くのリクエストを送信しています。
500	InternalServerErrorException	AWS Serverless Application Repository サービスで内部エラーが発生しました。

OPTIONS

パスパラメータ

名前	型	必須	説明
applicationId	文字列	True	アプリケーションの Amazon リソースネーム (ARN) です。

レスポンス

ステータスコード	レスポンスモデル	説明
200	なし	200 レスポンス

PATCH

オペレーション ID: UpdateApplication

指定されたアプリケーションを更新します。

パスパラメータ

名前	型	必須	説明
applicationId	文字列	True	アプリケーションの Amazon リソースネーム (ARN) です。

レスポンス

ステータスコード	レスポンスモデル	説明
200	Application	成功
400	BadRequestException	リクエストに含まれているパラメータの 1 つが無効です。
403	ForbiddenException	クライアントは認証されていません。
404	NotFoundException	リクエストで指定されたリソース (例えば、アクセスポリシーステートメント) は存在しません。
409	ConflictException	リソースは既に存在します。

ステータスコード	レスポンスモデル	説明
429	TooManyRequestsException	クライアントが、単位時間あたりの許可されるリクエスト数よりも多くのリクエストを送信しています。
500	InternalServerErrorException	AWS Serverless Application Repository サービスで内部エラーが発生しました。

スキーマ

リクエストボディ

PATCH スキーマ

```
{
  "description": "string",
  "author": "string",
  "readmeBody": "string",
  "readmeUrl": "string",
  "labels": [
    "string"
  ],
  "homePageUrl": "string"
}
```

レスポンス本文

Application スキーマ

```
{
  "applicationId": "string",
  "name": "string",
  "description": "string",
  "author": "string",
  "isVerifiedAuthor": boolean,
  "verifiedAuthorUrl": "string",
```

```
  "spdxLicenseId": "string",
  "licenseUrl": "string",
  "readmeUrl": "string",
  "labels": [
    "string"
  ],
  "creationTime": "string",
  "homePageUrl": "string",
  "version": {
    "applicationId": "string",
    "semanticVersion": "string",
    "sourceCodeUrl": "string",
    "sourceCodeArchiveUrl": "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"
        ]
      }
    ],
    "requiredCapabilities": [
      enum
    ],
    "resourcesSupported": boolean
  }
}
```

BadRequestException スキーマ

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

ForbiddenException スキーマ

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

NotFoundException スキーマ

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

ConflictException スキーマ

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

TooManyRequestsException スキーマ

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

InternalServerErrorException スキーマ

```
{
```

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

プロパティ

Application

アプリケーションに関する詳細。

applicationId

アプリケーションの Amazon リソースネーム (ARN)。

タイプ：文字列
必須: True

name

アプリケーションの名前。

最小長: 1 最大長 = 140。

パターン: "[a-zA-Z0-9\\-]+";

タイプ：文字列
必須: True

description

アプリケーションの説明。

最小長: 1 最大長 = 256。

タイプ：文字列
必須: True

author

アプリケーションを公開する作成者の名前。

最小長: 1 最大長 = 127。

パターン: "^[a-z0-9](([a-z0-9]|-(?!-))*[a-z0-9])?$";

タイプ：文字列
必須: True

isVerifiedAuthor

このアプリケーションの作成者が検証されているかどうかを指定します。つまり、 AWS は、合理的かつ慎重なサービスプロバイダーとして、リクエスタから提供された情報を善意で確認し、リクエスタのアイデンティティがクレーム済みであることを確認しています。

タイプ：ブール値
必須: False

verifiedAuthorUrl

検証済み作成者のパブリックプロファイルへの URL。この URL は作成者によって提出されます。

タイプ：文字列
必須: False

spdxLicenseId

https://spdx.org/licenses/ からの有効な識別子。

タイプ：文字列
必須: False

licenseUrl

アプリケーションの spdxLicenseID 値に一致するアプリケーションのライセンスファイルへのリンク。

最大サイズ: 5 MB。

タイプ：文字列
必須: False

readmeUrl

Markdown 言語の readme ファイルへのリンク。アプリケーションとその動作に関する詳細な説明が含まれます。

最大サイズ: 5 MB。

タイプ：文字列
必須: False

labels

検索結果でアプリケーションを発見しやすくするラベル。

最小長: 1 最大長 = 127。ラベルの最大数: 10。

パターン: "^[a-zA-Z0-9+\\-_:\\/@]+$";

タイプ: string タイプの配列
必須: False

creationTime

このリソースが作成された日時。

タイプ：文字列
必須: False

homePageUrl

アプリケーションの GitHub リポジトリの場所など、アプリケーションに関する詳細情報を含む URL。

タイプ: 文字列
必須: False

version

アプリケーションに関するバージョン情報。

タイプ: バージョン
必須: False

BadRequestException

リクエストに含まれているパラメータの 1 つが無効です。

message

リクエストに含まれているパラメータの 1 つが無効です。

タイプ：文字列
必須: False

errorCode

400

タイプ：文字列
必須: False

Capability

一部のアプリケーションをデプロイするために指定する必要がある値。

```
CAPABILITY_IAM
CAPABILITY_NAMED_IAM
CAPABILITY_AUTO_EXPAND
CAPABILITY_RESOURCE_POLICY
```

ConflictException

リソースは既に存在します。

message

リソースは既に存在します。

タイプ：文字列
必須: False

errorCode

409

タイプ：文字列
必須: False

ForbiddenException

クライアントは認証されていません。

message

クライアントは認証されていません。

タイプ：文字列
必須: False

errorCode

403

タイプ：文字列
必須: False

InternalServerErrorException

AWS Serverless Application Repository サービスで内部エラーが発生しました。

message

AWS Serverless Application Repository サービスで内部エラーが発生しました。

タイプ: 文字列
必須: False

errorCode

500

タイプ：文字列
必須: False

NotFoundException

リクエストで指定されたリソース (例えば、アクセスポリシーステートメント) は存在しません。

message

リクエストで指定されたリソース (例えば、アクセスポリシーステートメント) は存在しません。

タイプ：文字列
必須: False

errorCode

404

タイプ：文字列
必須: False

ParameterDefinition

アプリケーションでサポートされるパラメータ。

name

パラメータの名前。

タイプ：文字列
必須: True

defaultValue

スタックの作成時に値を指定しなかった場合に、テンプレートで使用される適切な型の値。パラメーターの制約を定義する場合は、これらの制約に従う値を指定する必要があります。

タイプ：文字列
必須: False

description

パラメータについて説明する最大 4000 文字の文字列。

タイプ：文字列
必須: False

type

パラメータのタイプ。

有効な値: `String | Number | List<Number> | CommaDelimitedList`

`String`: リテラル文字列。

例えば、`"MyUserName"` と指定することができます。

`Number`: 整数または float. AWS CloudFormation valid は、パラメータ値を数値として検証します。ただし、テンプレート内の他の場所で使用した場合には (`Ref` 組み込み関数を使用した場合など) 文字列として扱います。

例えば、`"8888"` のように指定することがができます。

`List<Number>`: カンマで区切られた整数または浮動小数値の配列。 AWS CloudFormation はパラメータ値を数値として検証します。ただし、テンプレート内の他の場所で使用した場合には (`Ref` 組み込み関数を使用した場合など) 文字列のリストとして扱います。

例えば、「80,20」を指定すると、`Ref` は `["80","20"]` になります。

`CommaDelimitedList`: カンマで区切られたリテラル文字列の配列。文字列の合計数は、カンマの合計数よりも 1 つ多いはずです。また、各メンバー文字列の前後の空白は削除されます。

例えば、「test,dev,prod」を指定すると、`Ref` は `["test","dev","prod"]` になります。

タイプ：文字列
必須: False

noEcho

スタックの詳細を取得する呼び出しが他のユーザーによって作成された場合に、必ずパラメータ値をマスクするかどうか。値を true に設定すると、パラメータ値はアスタリスク (*****) でマスクされます。

タイプ：ブール値
必須: False

allowedPattern

`String` 型に使用できるパターンを表す正規表現。

タイプ：文字列
必須: False

constraintDescription

制約が違反された場合に、制約について説明する文字列。たとえば、制約の説明を指定しないとき、許容されているパターンが `[A-Za-z0-9]+` であるパラメーターの場合、ユーザーが無効な値を指定すると次のエラーメッセージが表示されます。

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

「must contain only uppercase and lowercase letters and numbers」などの制約の説明を追加することによって、次のようにカスタマイズされたエラーメッセージを表示することができます。

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

タイプ：文字列
必須: False

minValue

`Number` タイプに使用できる数値の最小値を決定する数値。

タイプ: 整数
必須: False

maxValue

`Number` タイプに使用できる数値の最大値を決定する数値。

タイプ: 整数
必須: False

minLength

String タイプに使用できる最小文字数を決定する整数値。

タイプ: 整数
必須: False

maxLength

String タイプに使用できる最大文字数を決定する整数値。

タイプ: 整数
必須: False

allowedValues

パラメーターに許容される一連の値を含む配列。

タイプ: string タイプの配列
必須: False

referencedByResources

このパラメータを使用する AWS SAM リソースのリスト。

タイプ: string タイプの配列
必須: True

TooManyRequestsException

クライアントが、単位時間あたりの許可されるリクエスト数よりも多くのリクエストを送信しています。

message

クライアントが、単位時間あたりの許可されるリクエスト数よりも多くのリクエストを送信しています。

タイプ：文字列
必須: False

errorCode

429

タイプ：文字列
必須: False

UpdateApplicationInput

アプリケーションリクエストを更新します。

description

アプリケーションの説明。

最小長: 1 最大長 = 256。

タイプ：文字列
必須: False

author

アプリケーションを公開する作成者の名前。

最小長: 1 最大長 = 127。

パターン: "^[a-z0-9](([a-z0-9]|-(?!-))*[a-z0-9])?$";

タイプ：文字列
必須: False

readmeBody

Markdown 言語のテキスト readme ファイル。アプリケーションとその動作に関する詳細な説明が含まれます。

最大サイズ: 5 MB。

タイプ：文字列
必須: False

readmeUrl

Markdown 言語の readme ファイルへのリンク。アプリケーションとその動作に関する詳細な説明が含まれます。

最大サイズ: 5 MB。

タイプ：文字列
必須: False

labels

検索結果でアプリケーションを発見しやすくするラベル。

最小長: 1 最大長 = 127。ラベルの最大数: 10。

パターン: "^[a-zA-Z0-9+\\-_:\\/@]+$";

タイプ: string タイプの配列
必須: False

homePageUrl

アプリケーションの GitHub リポジトリの場所など、アプリケーションに関する詳細情報を含む URL。

タイプ: 文字列
必須: False

Version

アプリケーションのバージョンの詳細。

applicationId

アプリケーションの Amazon リソースネーム (ARN)。

タイプ：文字列
必須: True

semanticVersion

アプリケーションのセマンティックバージョン：

https://semver.org/

タイプ：文字列
必須: True

sourceCodeUrl

特定の GitHub コミットの URL など、アプリケーションのソースコードのパブリックリポジトリへのリンク。

タイプ: 文字列
必須: False

sourceCodeArchiveUrl

アプリケーションのこのバージョンのソースコードの ZIP アーカイブを含む S3 オブジェクトへのリンク。

最大サイズ: 50 MB。

タイプ：文字列
必須: False

templateUrl

アプリケーションのパッケージ化された AWS SAM テンプレートへのリンク。

タイプ: 文字列
必須: True

creationTime

このリソースが作成された日時。

タイプ：文字列

必須: True

parameterDefinitions

アプリケーションでサポートされるパラメータタイプの配列。

タイプ: ParameterDefinition タイプの配列
必須: True

requiredCapabilities

特定のアプリケーションをデプロイする前に指定する必要がある値のリスト。一部のアプリケーションには、新しい AWS Identity and Access Management（ IAM) ユーザーを作成するなど、 AWS アカウントのアクセス許可に影響を与える可能性のあるリソースが含まれている場合があります。このようなアプリケーションの場合は、このパラメータを指定して、それらの機能を明示的に認識する必要があります。

有効な値は、CAPABILITY_IAM、CAPABILITY_NAMED_IAM、CAPABILITY_RESOURCE_POLICY、および CAPABILITY_AUTO_EXPAND のみです。

次のリソースでは、 CAPABILITY_IAMまたは を指定する必要がありますCAPABILITY_NAMED_IAM: AWS::IAM::Group、AWS::IAM::InstanceProfile、AWS::IAM::Policy、および AWS::IAM::Role。アプリケーションに IAM リソースがある場合、CAPABILITY_IAM または CAPABILITY_NAMED_IAM のいずれかを指定できます。アプリケーションにカスタム名を持つ IAM リソースがある場合は、CAPABILITY_NAMED_IAM を指定する必要があります。

次のリソースではCAPABILITY_RESOURCE_POLICYAWS::Lambda::Permission、、AWS::IAM:Policy、、AWS::Application/よび を指定する必要がありますAWS::SNS::TopicPolicy。

1 つまたは複数のネストされたアプリケーションが含まれているアプリケーションでは、CAPABILITY_AUTO_EXPAND を指定する必要があります。

アプリケーションテンプレートに前述のリソースが含まれている場合、デプロイする前にアプリケーションに関連付けられたすべてのアクセス許可を確認することをお勧めします。機能を必要とするアプリケーションにこのパラメータを指定しないと、呼び出しは失敗します。

タイプ: Capability タイプの配列
必須: True

resourcesSupported

このアプリケーションに含まれるすべての AWS リソースが、取得するリージョンでサポートされているかどうか。

タイプ：ブール値
必須: True

以下も参照してください。

言語固有の AWS SDKs とリファレンスのいずれかでこの API を使用する方法の詳細については、以下を参照してください。

GetApplication

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby v3

DeleteApplication

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3

- AWS SDK for Python
- AWS SDK for Ruby v3

UpdateApplication

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby v3

Applications applicationId Changesets

[URI]

`/applications/applicationId/changesets`

HTTP メソッド

POST

オペレーション ID: `CreateCloudFormationChangeSet`

指定されたアプリケーション AWS CloudFormation の変更セットを作成します。

パスパラメータ

名前	型	必須	説明
applicationId	文字列	True	アプリケーションの Amazon リソースネーム (ARN) です。

レスポンス

ステータスコード	レスポンスモデル	説明
201	ChangeSetDetails	成功
400	BadRequestException	リクエストに含まれているパラメータの 1 つが無効です。
403	ForbiddenException	クライアントは認証されていません。
429	TooManyRequestsException	クライアントが、単位時間あたりの許可されるリクエスト数よりも多くのリクエストを送信しています。
500	InternalServerErrorException	AWS Serverless Application Repository サービスで内部エラーが発生しました。

OPTIONS

パスパラメータ

名前	型	必須	説明
applicationId	文字列	True	アプリケーションの Amazon リソースネーム (ARN) です。

レスポンス

ステータスコード	レスポンスモデル	説明
200	なし	200 レスポンス

スキーマ

リクエストボディ

POST スキーマ

```
{
  "stackName": "string",
  "semanticVersion": "string",
  "templateId": "string",
  "parameterOverrides": [
    {
      "name": "string",
      "value": "string"
    }
  ],
  "capabilities": [
    "string"
  ],
  "changeSetName": "string",
  "clientToken": "string",
  "description": "string",
  "notificationArns": [
    "string"
  ],
  "resourceTypes": [
    "string"
  ],
  "rollbackConfiguration": {
    "rollbackTriggers": [
      {
        "arn": "string",
        "type": "string"
      }
    ],
    "monitoringTimeInMinutes": integer
  },
  "tags": [
    {
      "key": "string",
      "value": "string"
    }
  ]
```

```
}
```

レスポンス本文

ChangeSetDetails スキーマ

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

BadRequestException スキーマ

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

ForbiddenException スキーマ

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

TooManyRequestsException スキーマ

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

InternalServerErrorException スキーマ

```
{
  "message": "string",
```

```
  "errorCode": "string"
}
```

プロパティ

BadRequestException

リクエストに含まれているパラメータの 1 つが無効です。

message

リクエストに含まれているパラメータの 1 つが無効です。

タイプ：文字列
必須: False

errorCode

400

タイプ：文字列
必須: False

ChangeSetDetails

変更セットの詳細。

applicationId

アプリケーションの Amazon リソースネーム (ARN)。

タイプ：文字列
必須: True

semanticVersion

アプリケーションのセマンティックバージョン：

https://semver.org/

タイプ：文字列

必須: True

changeSetId

変更セットの Amazon リソースネーム (ARN)。

長さの制限: 最小長は 1 です。

パターン: ARN:[-a-zA-Z0-9:/]*

タイプ：文字列
必須: True

stackId

スタックの一意の ID。

タイプ：文字列
必須: True

CreateCloudFormationChangeSetInput

アプリケーション変更セットリクエストを作成します。

stackName

このプロパティは、 AWS CloudFormation CreateChangeSet API の同じ名前の パラメータに対応します。

タイプ: 文字列
必須: True

semanticVersion

アプリケーションのセマンティックバージョン：

https://semver.org/

タイプ：文字列
必須: False

templateId

によって返される UUID CreateCloudFormationTemplate。

パターン: [0-9a-fA-F]{8}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{12}

タイプ：文字列
必須: False

parameterOverrides

アプリケーションのパラメータのパラメータ値のリスト。

タイプ: ParameterValue タイプの配列
必須: False

capabilities

特定のアプリケーションをデプロイする前に指定する必要がある値のリスト。一部のアプリケーションには、新しい AWS Identity and Access Management （ IAM) ユーザーを作成するなど、 AWS アカウントのアクセス許可に影響を与える可能性のあるリソースが含まれている場合があります。このようなアプリケーションの場合は、このパラメータを指定して、それらの機能を明示的に認識する必要があります。

有効な値は、CAPABILITY_IAM、CAPABILITY_NAMED_IAM、CAPABILITY_RESOURCE_POLICY、および CAPABILITY_AUTO_EXPAND のみです。

次のリソースでは、 CAPABILITY_IAMまたは を指定する必要がありますCAPABILITY_NAMED_IAM: AWS::IAM::Group、AWS::IAM::InstanceProfile、AWS::IAM::Policy、および AWS::IAM::Role。アプリケーションに IAM リソースがある場合、CAPABILITY_IAM または CAPABILITY_NAMED_IAM のいずれかを指定できます。アプリケーションにカスタム名を持つ IAM リソースがある場合は、CAPABILITY_NAMED_IAM を指定する必要があります。

次のリソースではCAPABILITY_RESOURCE_POLICYAWS::Lambda::Permission、、AWS::IAM:Policy、、AWS::ApplicationAおよび AWS::SNS:TopicPolicy を指定する必要があります。

1 つまたは複数のネストされたアプリケーションが含まれているアプリケーションでは、CAPABILITY_AUTO_EXPAND を指定する必要があります。

アプリケーションテンプレートに前述のリソースが含まれている場合、デプロイする前にアプリケーションに関連付けられたすべてのアクセス許可を確認することをお勧めします。機能を必要とするアプリケーションにこのパラメータを指定しないと、呼び出しは失敗します。

タイプ: string タイプの配列
必須: False

changeSetName

このプロパティは、 AWS CloudFormation CreateChangeSet API の同じ名前の パラメータに対応します。

タイプ: 文字列
必須: False

clientToken

このプロパティは、 AWS CloudFormation CreateChangeSet API の同じ名前の パラメータに対応します。

タイプ: 文字列
必須: False

description

このプロパティは、AWS CloudFormation CreateChangeSetAPI の同じ名前の パラメータに対応します。

タイプ: 文字列
必須: False

notificationArns

このプロパティは、 AWS CloudFormation CreateChangeSet API の同じ名前の パラメータに対応します。

タイプ: string タイプの配列
必須: False

resourceTypes

このプロパティは、 AWS CloudFormation CreateChangeSet API の同じ名前の パラメータに対応します。

タイプ: string タイプの配列
必須: False

rollbackConfiguration

このプロパティは、 AWS CloudFormation CreateChangeSet API の同じ名前の パラメータに対応します。

タイプ： RollbackConfiguration
必須: False

tags

このプロパティは、 AWS CloudFormation CreateChangeSet API の同じ名前の パラメータに対応します。

タイプ: Tag タイプの配列
必須: False

ForbiddenException

クライアントは認証されていません。

message

クライアントは認証されていません。

タイプ：文字列
必須: False

errorCode

403

タイプ：文字列
必須: False

InternalServerErrorException

AWS Serverless Application Repository サービスで内部エラーが発生しました。

message

AWS Serverless Application Repository サービスで内部エラーが発生しました。

タイプ: 文字列
必須: False

errorCode

500

タイプ：文字列
必須: False

ParameterValue

アプリケーションのパラメータ値。

name

パラメータに関連付けられたキー。特定のパラメータにキーと値が指定されていない場合、 AWS CloudFormation はテンプレートに指定されているデフォルト値を使用します。

タイプ：文字列
必須: True

value

パラメータに関連付けられた入力値。

タイプ：文字列

必須: True

RollbackConfiguration

このプロパティは、 AWS CloudFormation RollbackConfiguration データ型に対応します。

rollbackTriggers

このプロパティは、AWS CloudFormation RollbackConfigurationデータ型の同じ名前の内容に対応します。

タイプ: RollbackTrigger タイプの配列
必須: False

monitoringTimeInMinutes

このプロパティは、AWS CloudFormation RollbackConfigurationデータ型の同じ名前の内容に対応します。

タイプ: 整数
必須: False

RollbackTrigger

このプロパティは AWS CloudFormation RollbackTrigger データ型に対応します。

arn

このプロパティは、AWS CloudFormation RollbackTriggerデータ型の同じ名前の内容に対応します。

タイプ: 文字列
必須: True

type

このプロパティは、AWS CloudFormation RollbackTriggerデータ型の同じ名前の内容に対応します。

タイプ: 文字列

必須: True

Tag

このプロパティは AWS CloudFormation Tag データタイプに対応します。

key

このプロパティは、AWS CloudFormation Tag データタイプの同じ名前のコンテンツに対応します。

タイプ：文字列
必須: True

value

このプロパティは、AWS CloudFormation Tag データタイプの同じ名前のコンテンツに対応します。

タイプ：文字列
必須: True

TooManyRequestsException

クライアントが、単位時間あたりの許可されるリクエスト数よりも多くのリクエストを送信しています。

message

クライアントが、単位時間あたりの許可されるリクエスト数よりも多くのリクエストを送信しています。

タイプ：文字列
必須: False

errorCode

429

タイプ：文字列
必須: False

以下も参照してください。

言語固有の AWS SDKs とリファレンスのいずれかでこの API を使用する方法の詳細については、以下を参照してください。

CreateCloudFormationChangeSet

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby v3

Applications applicationId Dependencies

[URI]

`/applications/applicationId/dependencies`

HTTP メソッド

GET

オペレーション ID: `ListApplicationDependencies`

包含するアプリケーションにネストされたアプリケーションのリストを取得します

パスパラメータ

名前	型	必須	説明
applicationId	文字列	True	アプリケーションの Amazon リソースネーム (ARN) です。

クエリパラメータ

名前	型	必須	説明
nextToken	文字列	False	ページ分割を始める場所を指定するトークン。
maxItems	文字列	False	返される項目の合計数。
semanticVersion	文字列	False	取得するアプリケーションのセマンティックバージョン。

レスポンス

ステータスコード	レスポンスモデル	説明
200	ApplicationDependencyPage	成功
400	BadRequestException	リクエストに含まれているパラメータの 1 つが無効です。
403	ForbiddenException	クライアントは認証されていません。
404	NotFoundException	リクエストで指定されたリソース (例えば、アクセスポリシーステートメント) は存在しません。
429	TooManyRequestsException	クライアントが、単位時間あたりの許可されるリクエスト数よりも多くのリクエストを送信しています。

ステータスコード	レスポンスモデル	説明
500	InternalServerErrorException	AWS Serverless Application Repository サービスで内部エラーが発生しました。

OPTIONS

パスパラメータ

名前	型	必須	説明
applicationId	文字列	True	アプリケーションの Amazon リソースネーム (ARN) です。

レスポンス

ステータスコード	レスポンスモデル	説明
200	なし	200 レスポンス

スキーマ

レスポンス本文

ApplicationDependencyPage スキーマ

```
{
  "dependencies": [
    {
      "applicationId": "string",
      "semanticVersion": "string"
    }
  ],
  "nextToken": "string"
}
```

BadRequestException スキーマ

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

ForbiddenException スキーマ

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

NotFoundException スキーマ

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

TooManyRequestsException スキーマ

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

InternalServerErrorException スキーマ

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

プロパティ

ApplicationDependencyPage

アプリケーションにネストされたアプリケーション概要のリスト。

dependencies

アプリケーションにネストされたアプリケーション概要の配列。

タイプ: ApplicationDependencySummary タイプの配列
必須: True

nextToken

次の結果ページを要求するためのトークン。

タイプ：文字列
必須: False

ApplicationDependencySummary

ネストされたアプリケーション概要。

applicationId

ネストされたアプリケーションの Amazon リソースネーム (ARN)。

タイプ：文字列
必須: True

semanticVersion

ネストされたアプリケーションのセマンティックバージョンです。

タイプ：文字列
必須: True

BadRequestException

リクエストに含まれているパラメータの 1 つが無効です。

message

リクエストに含まれているパラメータの 1 つが無効です。

タイプ：文字列
必須: False

errorCode

400

タイプ：文字列
必須: False

ForbiddenException

クライアントは認証されていません。

message

クライアントは認証されていません。

タイプ：文字列
必須: False

errorCode

403

タイプ：文字列
必須: False

InternalServerErrorException

AWS Serverless Application Repository サービスで内部エラーが発生しました。

message

AWS Serverless Application Repository サービスで内部エラーが発生しました。

タイプ: 文字列
必須: False

errorCode

500

タイプ：文字列
必須: False

NotFoundException

リクエストで指定されたリソース (例えば、アクセスポリシーステートメント) は存在しません。

message

リクエストで指定されたリソース (例えば、アクセスポリシーステートメント) は存在しません。

タイプ：文字列
必須: False

errorCode

404

タイプ：文字列
必須: False

TooManyRequestsException

クライアントが、単位時間あたりの許可されるリクエスト数よりも多くのリクエストを送信しています。

message

クライアントが、単位時間あたりの許可されるリクエスト数よりも多くのリクエストを送信しています。

タイプ：文字列

必須: False

errorCode

429

タイプ：文字列

必須: False

以下も参照してください。

言語固有の AWS SDKs とリファレンスのいずれかでこの API を使用する方法の詳細については、以下を参照してください。

ListApplicationDependencies

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby v3

Applications applicationId Policy

[URI]

`/applications/applicationId/policy`

HTTP メソッド

GET

オペレーション ID: `GetApplicationPolicy`

アプリケーションのポリシーを取得します。

パスパラメータ

名前	型	必須	説明
applicationId	文字列	True	アプリケーションの Amazon リソースネーム (ARN) です。

レスポンス

ステータスコード	レスポンスモデル	説明
200	ApplicationPolicy	成功
400	BadRequestException	リクエストに含まれているパラメータの 1 つが無効です。
403	ForbiddenException	クライアントは認証されていません。
404	NotFoundException	リクエストで指定されたリソース (例えば、アクセスポリシーステートメント) は存在しません。

ステータスコード	レスポンスモデル	説明
429	TooManyRequestsException	クライアントが、単位時間あたりの許可されるリクエスト数よりも多くのリクエストを送信しています。
500	InternalServerErrorException	AWS Serverless Application Repository サービスで内部エラーが発生しました。

PUT

オペレーション ID: PutApplicationPolicy

アプリケーションのアクセス許可ポリシーを設定します。このオペレーションでサポートされているアクションの詳細については、アプリケーションへのアクセス許可を参照してください。

パスパラメータ

名前	型	必須	説明
applicationId	文字列	True	アプリケーションの Amazon リソースネーム (ARN) です。

レスポンス

ステータスコード	レスポンスモデル	説明
200	ApplicationPolicy	成功
400	BadRequestException	リクエストに含まれているパラメータの 1 つが無効です。
403	ForbiddenException	クライアントは認証されていません。

ステータスコード	レスポンスモデル	説明
404	NotFoundException	リクエストで指定されたリソース (例えば、アクセスポリシーステートメント) は存在しません。
429	TooManyRequestsException	クライアントが、単位時間あたりの許可されるリクエスト数よりも多くのリクエストを送信しています。
500	InternalServerErrorException	AWS Serverless Application Repository サービスで内部エラーが発生しました。

OPTIONS

パスパラメータ

名前	型	必須	説明
applicationId	文字列	True	アプリケーションの Amazon リソースネーム (ARN) です。

レスポンス

ステータスコード	レスポンスモデル	説明
200	なし	200 レスポンス

スキーマ

リクエストボディ

PUT スキーマ

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

レスポンス本文

ApplicationPolicy スキーマ

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

BadRequestException スキーマ

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

ForbiddenException スキーマ

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

NotFoundException スキーマ

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

TooManyRequestsException スキーマ

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

InternalServerErrorException スキーマ

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

プロパティ

ApplicationPolicy

アプリケーションに適用されるポリシーステートメント。

statements

アプリケーションに適用されるポリシーステートメントの配列。

タイプ: ApplicationPolicyStatement タイプの配列
必須: True

ApplicationPolicyStatement

アプリケーションに適用されるポリシーステートメント。

statementId

ステートメントの一意の ID。

タイプ：文字列
必須: False

principals

アプリケーションを共有する AWS アカウント IDs の配列、またはアプリケーションを公開するための *。

タイプ: string タイプの配列
必須: True

actions

このオペレーションでサポートされているアクションの詳細については、アプリケーションへのアクセス許可を参照してください。

タイプ: string タイプの配列
必須: True

principalOrgIDs

アプリケーションを共有する AWS Organizations ID。

タイプ: string タイプの配列
必須: False

BadRequestException

リクエストに含まれているパラメータの 1 つが無効です。

message

リクエストに含まれているパラメータの 1 つが無効です。

タイプ：文字列
必須: False

errorCode

400

タイプ：文字列
必須: False

ForbiddenException

クライアントは認証されていません。

message

クライアントは認証されていません。

タイプ：文字列
必須: False

errorCode

403

タイプ：文字列
必須: False

InternalServerErrorException

AWS Serverless Application Repository サービスで内部エラーが発生しました。

message

AWS Serverless Application Repository サービスで内部エラーが発生しました。

タイプ: 文字列
必須: False

errorCode

500

タイプ：文字列
必須: False

NotFoundException

リクエストで指定されたリソース (例えば、アクセスポリシーステートメント) は存在しません。

message

リクエストで指定されたリソース (例えば、アクセスポリシーステートメント) は存在しません。

タイプ：文字列
必須: False

errorCode

404

タイプ：文字列
必須: False

TooManyRequestsException

クライアントが、単位時間あたりの許可されるリクエスト数よりも多くのリクエストを送信しています。

message

クライアントが、単位時間あたりの許可されるリクエスト数よりも多くのリクエストを送信しています。

タイプ：文字列
必須: False

errorCode

429

タイプ：文字列
必須: False

以下も参照してください。

言語固有の AWS SDKs とリファレンスのいずれかでこの API を使用する方法の詳細については、以下を参照してください。

GetApplicationPolicy

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby v3

PutApplicationPolicy

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby v3

Applications applicationId Templates

[URI]

`/applications/applicationId/templates`

HTTP メソッド

POST

オペレーション ID: `CreateCloudFormationTemplate`

AWS CloudFormation テンプレートを作成します。

パスパラメータ

名前	型	必須	説明
applicationId	文字列	True	アプリケーションの Amazon リソースネーム (ARN) です。

レスポンス

ステータスコード	レスポンスモデル	説明
201	TemplateDetails	成功
400	BadRequestException	リクエストに含まれているパラメータの 1 つが無効です。
403	ForbiddenException	クライアントは認証されていません。
404	NotFoundException	リクエストで指定されたリソース (例えば、アクセスポリシーステートメント) は存在しません。
429	TooManyRequestsException	クライアントが、単位時間あたりの許可されるリクエスト数よりも多くのリクエストを送信しています。
500	InternalServerErrorException	AWS Serverless Application Repository サービスで内部エラーが発生しました。

OPTIONS

パスパラメータ

名前	型	必須	説明
applicationId	文字列	True	アプリケーションの Amazon リソースネーム (ARN) です。

レスポンス

ステータスコード	レスポンスモデル	説明
200	なし	200 レスポンス

スキーマ

リクエストボディ

POST スキーマ

```
{
  "semanticVersion": "string"
}
```

レスポンス本文

TemplateDetails スキーマ

```
{
  "templateId": "string",
  "templateUrl": "string",
  "applicationId": "string",
  "semanticVersion": "string",
  "status": enum,
  "creationTime": "string",
  "expirationTime": "string"
}
```

BadRequestException スキーマ

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

ForbiddenException スキーマ

```
{
```

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

NotFoundException スキーマ

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

TooManyRequestsException スキーマ

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

InternalServerErrorException スキーマ

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

プロパティ

BadRequestException

リクエストに含まれているパラメータの 1 つが無効です。

message

リクエストに含まれているパラメータの 1 つが無効です。

タイプ：文字列
必須: False

errorCode

400

タイプ：文字列
必須: False

CreateCloudFormationTemplateInput

テンプレートリクエストを作成します。

semanticVersion

アプリケーションのセマンティックバージョン：

https://semver.org/

タイプ：文字列
必須: False

ForbiddenException

クライアントは認証されていません。

message

クライアントは認証されていません。

タイプ：文字列
必須: False

errorCode

403

タイプ：文字列
必須: False

InternalServerErrorException

AWS Serverless Application Repository サービスで内部エラーが発生しました。

message

AWS Serverless Application Repository サービスで内部エラーが発生しました。

タイプ: 文字列
必須: False

errorCode

500

タイプ：文字列
必須: False

NotFoundException

リクエストで指定されたリソース (例えば、アクセスポリシーステートメント) は存在しません。

message

リクエストで指定されたリソース (例えば、アクセスポリシーステートメント) は存在しません。

タイプ：文字列
必須: False

errorCode

404

タイプ：文字列
必須: False

TemplateDetails

テンプレートの詳細。

templateId

によって返される UUID CreateCloudFormationTemplate。

パターン: [0-9a-fA-F]{8}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{12}

タイプ：文字列
必須: True

templateUrl

を使用してアプリケーションをデプロイするために使用できるテンプレートへのリンク AWS CloudFormation。

タイプ: 文字列
必須: True

applicationId

アプリケーションの Amazon リソースネーム (ARN)。

タイプ：文字列
必須: True

semanticVersion

アプリケーションのセマンティックバージョン：

https://semver.org/

タイプ：文字列
必須: True

status

テンプレート作成ワークフローのステータス。

使用できる値: `PREPARING | ACTIVE | EXPIRED`

タイプ：文字列
必須: True
値： `PREPARING | ACTIVE | EXPIRED`

creationTime

このリソースが作成された日時。

タイプ：文字列
必須: True

expirationTime

このテンプレートの有効期限が切れる日時。テンプレートは作成から 1 時間後に期限切れになります。

タイプ：文字列
必須: True

TooManyRequestsException

クライアントが、単位時間あたりの許可されるリクエスト数よりも多くのリクエストを送信しています。

message

クライアントが、単位時間あたりの許可されるリクエスト数よりも多くのリクエストを送信しています。

タイプ：文字列
必須: False

errorCode

429

タイプ：文字列
必須: False

以下も参照してください。

言語固有の AWS SDKs とリファレンスのいずれかでこの API を使用する方法の詳細については、以下を参照してください。

CreateCloudFormationTemplate

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby v3

Applications applicationId Templates templateId

[URI]

`/applications/applicationId/templates/templateId`

HTTP メソッド

GET

オペレーション ID: `GetCloudFormationTemplate`

指定された AWS CloudFormation テンプレートを取得します。

パスパラメータ

名前	型	必須	説明
applicationId	文字列	True	アプリケーションの Amazon リソースネーム (ARN) です。
templateId	文字列	True	によって返される UUID CreateClo

名前	型	必須	説明
			udFormationTemplate。 パターン: [0-9a-fA-F]{8}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{12}

レスポンス

ステータスコード	レスポンスモデル	説明
200	TemplateDetails	成功
400	BadRequestException	リクエストに含まれているパラメータの 1 つが無効です。
403	ForbiddenException	クライアントは認証されていません。
404	NotFoundException	リクエストで指定されたリソース (例えば、アクセスポリシーステートメント) は存在しません。
429	TooManyRequestsException	クライアントが、単位時間あたりの許可されるリクエスト数よりも多くのリクエストを送信しています。
500	InternalServerErrorException	AWS Serverless Application Repository サービスで内部エラーが発生しました。

OPTIONS

パスパラメータ

名前	型	必須	説明
applicationId	文字列	True	アプリケーションの Amazon リソースネーム (ARN) です。
templateId	文字列	True	によって返される UUID CreateCloudFormationTemplate。 パターン: [0-9a-fA-F]{8}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{12}

レスポンス

ステータスコード	レスポンスモデル	説明
200	なし	200 レスポンス

スキーマ

レスポンス本文

TemplateDetails スキーマ

```
{
  "templateId": "string",
  "templateUrl": "string",
  "applicationId": "string",
  "semanticVersion": "string",
```

```
  "status": enum,
  "creationTime": "string",
  "expirationTime": "string"
}
```

BadRequestException スキーマ

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

ForbiddenException スキーマ

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

NotFoundException スキーマ

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

TooManyRequestsException スキーマ

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

InternalServerErrorException スキーマ

```
{
```

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

プロパティ

BadRequestException

リクエストに含まれているパラメータの 1 つが無効です。

message

リクエストに含まれているパラメータの 1 つが無効です。

タイプ：文字列
必須: False

errorCode

400

タイプ：文字列
必須: False

ForbiddenException

クライアントは認証されていません。

message

クライアントは認証されていません。

タイプ：文字列
必須: False

errorCode

403

タイプ：文字列
必須: False

InternalServerErrorException

AWS Serverless Application Repository サービスで内部エラーが発生しました。

message

AWS Serverless Application Repository サービスで内部エラーが発生しました。

タイプ: 文字列
必須: False

errorCode

500

タイプ：文字列
必須: False

NotFoundException

リクエストで指定されたリソース (例えば、アクセスポリシーステートメント) は存在しません。

message

リクエストで指定されたリソース (例えば、アクセスポリシーステートメント) は存在しません。

タイプ：文字列
必須: False

errorCode

404

タイプ：文字列
必須: False

TemplateDetails

テンプレートの詳細。

templateId

によって返される UUID CreateCloudFormationTemplate。

パターン: [0-9a-fA-F]{8}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{12}

タイプ：文字列
必須: True

templateUrl

を使用してアプリケーションをデプロイするために使用できるテンプレートへのリンク AWS CloudFormation。

タイプ: 文字列
必須: True

applicationId

アプリケーションの Amazon リソースネーム (ARN)。

タイプ：文字列
必須: True

semanticVersion

アプリケーションのセマンティックバージョン：

https://semver.org/

タイプ：文字列
必須: True

status

テンプレート作成ワークフローのステータス。

使用できる値: PREPARING | ACTIVE | EXPIRED

タイプ：文字列
必須: True
値： PREPARING | ACTIVE | EXPIRED

creationTime

このリソースが作成された日時。

タイプ：文字列
必須: True

expirationTime

このテンプレートの有効期限が切れる日時。テンプレートは作成から 1 時間後に期限切れになります。

タイプ：文字列
必須: True

TooManyRequestsException

クライアントが、単位時間あたりの許可されるリクエスト数よりも多くのリクエストを送信しています。

message

クライアントが、単位時間あたりの許可されるリクエスト数よりも多くのリクエストを送信しています。

タイプ：文字列
必須: False

errorCode

429

タイプ：文字列

必須: False

以下も参照してください。

言語固有の AWS SDKs とリファレンスのいずれかでこの API を使用する方法の詳細については、以下を参照してください。

GetCloudFormationTemplate

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby v3

Applications applicationId Unshare

[URI]

`/applications/applicationId/unshare`

HTTP メソッド

POST

オペレーション ID: `UnshareApplication`

Organization からアプリケーションの共有を解除します AWS 。

このオペレーションは、組織の管理アカウントからのみ呼び出すことができます。

パスパラメータ

名前	型	必須	説明
applicationId	文字列	True	アプリケーションの Amazon リソースネーム (ARN) です。

レスポンス

ステータスコード	レスポンスモデル	説明
204	なし	成功
400	BadRequestException	リクエストに含まれているパラメータの 1 つが無効です。
403	ForbiddenException	クライアントは認証されていません。
404	NotFoundException	リクエストで指定されたリソース (例えば、アクセスポリシーステートメント) は存在しません。
429	TooManyRequestsException	クライアントが、単位時間あたりの許可されるリクエスト数よりも多くのリクエストを送信しています。
500	InternalServerErrorException	AWS Serverless Application Repository サービスで内部エラーが発生しました。

OPTIONS

パスパラメータ

名前	型	必須	説明
applicationId	文字列	True	アプリケーションの Amazon リソースネーム (ARN) です。

レスポンス

ステータスコード	レスポンスモデル	説明
200	なし	200 レスポンス

スキーマ

リクエストボディ

POST スキーマ

```
{
  "organizationId": "string"
}
```

レスポンス本文

BadRequestException スキーマ

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

ForbiddenException スキーマ

```
{
```

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

NotFoundException スキーマ

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

TooManyRequestsException スキーマ

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

InternalServerErrorException スキーマ

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

プロパティ

BadRequestException

リクエストに含まれているパラメータの 1 つが無効です。

message

リクエストに含まれているパラメータの 1 つが無効です。

タイプ：文字列
必須: False

errorCode

400

タイプ：文字列
必須: False

ForbiddenException

クライアントは認証されていません。

message

クライアントは認証されていません。

タイプ：文字列
必須: False

errorCode

403

タイプ：文字列
必須: False

InternalServerErrorException

AWS Serverless Application Repository サービスで内部エラーが発生しました。

message

AWS Serverless Application Repository サービスで内部エラーが発生しました。

タイプ: 文字列
必須: False

errorCode

500

タイプ：文字列
必須: False

NotFoundException

リクエストで指定されたリソース (例えば、アクセスポリシーステートメント) は存在しません。

message

リクエストで指定されたリソース (例えば、アクセスポリシーステートメント) は存在しません。

タイプ：文字列
必須: False

errorCode

404

タイプ：文字列
必須: False

TooManyRequestsException

クライアントが、単位時間あたりの許可されるリクエスト数よりも多くのリクエストを送信しています。

message

クライアントが、単位時間あたりの許可されるリクエスト数よりも多くのリクエストを送信しています。

タイプ：文字列
必須: False

errorCode

429

タイプ：文字列

必須: False

UnshareApplicationInput

アプリケーションリクエストの共有を解除します。

organizationId

アプリケーションの共有を解除する AWS Organizations ID。

タイプ: 文字列
必須: True

以下も参照してください。

言語固有の AWS SDKs とリファレンスのいずれかでこの API を使用する方法の詳細については、以下を参照してください。

UnshareApplication

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby v3

Applications applicationId Versions

[URI]

`/applications/applicationId/versions`

HTTP メソッド

GET

オペレーション ID: `ListApplicationVersions`

指定されたアプリケーションのバージョンを一覧表示します。

パスパラメータ

名前	型	必須	説明
applicationId	文字列	True	アプリケーションの Amazon リソースネーム (ARN) です。

クエリパラメータ

名前	型	必須	説明
maxItems	文字列	False	返される項目の合計数。
nextToken	文字列	False	ページ分割を始める場所を指定するトークン。

レスポンス

ステータスコード	レスポンスモデル	説明
200	ApplicationVersionPage	成功
400	BadRequestException	リクエストに含まれているパラメータの 1 つが無効です。
403	ForbiddenException	クライアントは認証されていません。

ステータスコード	レスポンスモデル	説明
404	NotFoundException	リクエストで指定されたリソース (例えば、アクセスポリシーステートメント) は存在しません。
429	TooManyRequestsException	クライアントが、単位時間あたりの許可されるリクエスト数よりも多くのリクエストを送信しています。
500	InternalServerErrorException	AWS Serverless Application Repository サービスで内部エラーが発生しました。

OPTIONS

パスパラメータ

名前	型	必須	説明
applicationId	文字列	True	アプリケーションの Amazon リソースネーム (ARN) です。

レスポンス

ステータスコード	レスポンスモデル	説明
200	なし	200 レスポンス

スキーマ

レスポンス本文

ApplicationVersionPage スキーマ

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

BadRequestException スキーマ

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

ForbiddenException スキーマ

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

NotFoundException スキーマ

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

TooManyRequestsException スキーマ

```
{
  "message": "string",
```

```
  "errorCode": "string"
}
```

InternalServerErrorException スキーマ

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

プロパティ

ApplicationVersionPage

アプリケーションのバージョン概要のリスト。

versions

アプリケーションのバージョン概要の配列。

タイプ: VersionSummary タイプの配列
必須: True

nextToken

次の結果ページを要求するためのトークン。

タイプ：文字列
必須: False

BadRequestException

リクエストに含まれているパラメータの 1 つが無効です。

message

リクエストに含まれているパラメータの 1 つが無効です。

タイプ：文字列
必須: False

errorCode

400

タイプ：文字列
必須: False

ForbiddenException

クライアントは認証されていません。

message

クライアントは認証されていません。

タイプ：文字列
必須: False

errorCode

403

タイプ：文字列
必須: False

InternalServerErrorException

AWS Serverless Application Repository サービスで内部エラーが発生しました。

message

AWS Serverless Application Repository サービスで内部エラーが発生しました。

タイプ: 文字列
必須: False

errorCode

500

タイプ：文字列
必須: False

NotFoundException

リクエストで指定されたリソース (例えば、アクセスポリシーステートメント) は存在しません。

message

リクエストで指定されたリソース (例えば、アクセスポリシーステートメント) は存在しません。

タイプ：文字列
必須: False

errorCode

404

タイプ：文字列
必須: False

TooManyRequestsException

クライアントが、単位時間あたりの許可されるリクエスト数よりも多くのリクエストを送信しています。

message

クライアントが、単位時間あたりの許可されるリクエスト数よりも多くのリクエストを送信しています。

タイプ：文字列
必須: False

errorCode

429

タイプ：文字列
必須: False

VersionSummary

アプリケーションのバージョン概要。

applicationId

アプリケーションの Amazon リソースネーム (ARN)。

タイプ：文字列
必須: True

semanticVersion

アプリケーションのセマンティックバージョン：

https://semver.org/

タイプ：文字列
必須: True

sourceCodeUrl

特定の GitHub コミットの URL など、アプリケーションのソースコードのパブリックリポジトリへのリンク。

タイプ: 文字列
必須: False

creationTime

このリソースが作成された日時。

タイプ：文字列
必須: True

以下も参照してください。

言語固有の AWS SDKs とリファレンスのいずれかでこの API を使用する方法の詳細については、以下を参照してください。

ListApplicationVersions

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby v3

Applications applicationId Versions semanticVersion

[URI]

`/applications/applicationId/versions/semanticVersion`

HTTP メソッド

PUT

オペレーション ID: `CreateApplicationVersion`

アプリケーションバージョンを作成します。

パスパラメータ

名前	型	必須	説明
applicationId	文字列	True	アプリケーションの Amazon リソースネーム (ARN) です。
semanticVersion	文字列	True	新しいバージョンのセマンティックバージョン。

レスポンス

ステータスコード	レスポンスモデル	説明
201	Version	成功
400	BadRequestException	リクエストに含まれているパラメータの 1 つが無効です。
403	ForbiddenException	クライアントは認証されていません。
409	ConflictException	リソースは既に存在します。
429	TooManyRequestsException	クライアントが、単位時間あたりの許可されるリクエスト数よりも多くのリクエストを送信しています。
500	InternalServerErrorException	AWS Serverless Application Repository サービスで内部エラーが発生しました。

OPTIONS

パスパラメータ

名前	型	必須	説明
applicationId	文字列	True	アプリケーションの Amazon リソースネーム (ARN) です。
semanticVersion	文字列	True	新しいバージョンのセマンティックバージョン。

レスポンス

ステータスコード	レスポンスモデル	説明
200	なし	200 レスポンス

スキーマ

リクエストボディ

PUT スキーマ

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

レスポンス本文

Version スキーマ

```
{
  "applicationId": "string",
  "semanticVersion": "string",
  "sourceCodeUrl": "string",
  "sourceCodeArchiveUrl": "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"
      ]
    }
  ],
  "requiredCapabilities": [
    enum
  ],
  "resourcesSupported": boolean
}
```

BadRequestException スキーマ

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

ForbiddenException スキーマ

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

ConflictException スキーマ

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

TooManyRequestsException スキーマ

```
{
```

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

InternalServerErrorException スキーマ

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

プロパティ

BadRequestException

リクエストに含まれているパラメータの 1 つが無効です。

message

リクエストに含まれているパラメータの 1 つが無効です。

タイプ：文字列
必須: False

errorCode

400

タイプ：文字列
必須: False

Capability

一部のアプリケーションをデプロイするために指定する必要がある値。

CAPABILITY_IAM
CAPABILITY_NAMED_IAM
CAPABILITY_AUTO_EXPAND
CAPABILITY_RESOURCE_POLICY

ConflictException

リソースは既に存在します。

message

リソースは既に存在します。

タイプ：文字列
必須: False

errorCode

409

タイプ：文字列
必須: False

CreateApplicationVersionInput

バージョンリクエストを作成します。

templateBody

アプリケーションの未加工のパッケージ化された AWS SAM テンプレート。

タイプ: 文字列
必須: False

templateUrl

アプリケーションのパッケージ化された AWS SAM テンプレートへのリンク。

タイプ: 文字列
必須: False

sourceCodeUrl

特定の GitHub コミットの URL など、アプリケーションのソースコードのパブリックリポジトリへのリンク。

タイプ: 文字列
必須: False

sourceCodeArchiveUrl

アプリケーションのこのバージョンのソースコードの ZIP アーカイブを含む S3 オブジェクトへのリンク。

最大サイズ: 50 MB。

タイプ：文字列
必須: False

ForbiddenException

クライアントは認証されていません。

message

クライアントは認証されていません。

タイプ：文字列
必須: False

errorCode

403

タイプ：文字列
必須: False

InternalServerErrorException

AWS Serverless Application Repository サービスで内部エラーが発生しました。

message

AWS Serverless Application Repository サービスで内部エラーが発生しました。

タイプ: 文字列

必須: False

errorCode

500

タイプ：文字列
必須: False

ParameterDefinition

アプリケーションでサポートされるパラメータ。

name

パラメータの名前。

タイプ：文字列
必須: True

defaultValue

スタックの作成時に値を指定しなかった場合に、テンプレートで使用される適切な型の値。パラメーターの制約を定義する場合は、これらの制約に従う値を指定する必要があります。

タイプ：文字列
必須: False

description

パラメータについて説明する最大 4000 文字の文字列。

タイプ：文字列
必須: False

type

パラメータのタイプ。

有効な値: `String | Number | List<Number> | CommaDelimitedList`

`String`: リテラル文字列。

例えば、`"MyUserName"` と指定することができます。

`Number`: 整数または float. AWS CloudFormation valid は、パラメータ値を数値として検証します。ただし、テンプレート内の他の場所で使用した場合には (`Ref` 組み込み関数を使用した場合など) 文字列として扱います。

例えば、`"8888"` のようにに指定することがができます。

`List<Number>`: カンマで区切られた整数または浮動小数値の配列。 AWS CloudFormation はパラメータ値を数値として検証します。ただし、テンプレート内の他の場所で使用した場合には (`Ref` 組み込み関数を使用した場合など) 文字列のリストとして扱います。

例えば、「80,20」を指定すると、`Ref` は `["80","20"]` になります。

`CommaDelimitedList`: カンマで区切られたリテラル文字列の配列。文字列の合計数は、カンマの合計数よりも 1 つ多いはずです。また、各メンバー文字列の前後の空白は削除されます。

例えば、「test,dev,prod」を指定すると、`Ref` は `["test","dev","prod"]` になります。

タイプ：文字列
必須: False

noEcho

スタックの詳細を取得する呼び出しが他のユーザーによって作成された場合に、必ずパラメータ値をマスクするかどうか。値を true に設定すると、パラメータ値はアスタリスク (*****) でマスクされます。

タイプ：ブール値
必須: False

allowedPattern

`String` 型に使用できるパターンを表す正規表現。

タイプ：文字列

必須: False

constraintDescription

制約が違反された場合に、制約について説明する文字列。たとえば、制約の説明を指定しないとき、許容されているパターンが [A-Za-z0-9]+ であるパラメーターの場合、ユーザーが無効な値を指定すると次のエラーメッセージが表示されます。

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

「must contain only uppercase and lowercase letters and numbers」などの制約の説明を追加することによって、次のようにカスタマイズされたエラーメッセージを表示することができます。

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

タイプ：文字列
必須: False

minValue

Number タイプに使用できる数値の最小値を決定する数値。

タイプ: 整数
必須: False

maxValue

Number タイプに使用できる数値の最大値を決定する数値。

タイプ: 整数
必須: False

minLength

String タイプに使用できる最小文字数を決定する整数値。

タイプ: 整数

必須: False

maxLength

String タイプに使用できる最大文字数を決定する整数値。

タイプ: 整数
必須: False

allowedValues

パラメーターに許容される一連の値を含む配列。

タイプ: string タイプの配列
必須: False

referencedByResources

このパラメータを使用する AWS SAM リソースのリスト。

タイプ: string タイプの配列
必須: True

TooManyRequestsException

クライアントが、単位時間あたりの許可されるリクエスト数よりも多くのリクエストを送信しています。

message

クライアントが、単位時間あたりの許可されるリクエスト数よりも多くのリクエストを送信しています。

タイプ：文字列
必須: False

errorCode

429

タイプ：文字列
必須: False

Version

アプリケーションのバージョンの詳細。

applicationId

アプリケーションの Amazon リソースネーム (ARN)。

タイプ：文字列
必須: True

semanticVersion

アプリケーションのセマンティックバージョン：

https://semver.org/

タイプ：文字列
必須: True

sourceCodeUrl

特定の GitHub コミットの URL など、アプリケーションのソースコードのパブリックリポジトリへのリンク。

タイプ: 文字列
必須: False

sourceCodeArchiveUrl

アプリケーションのこのバージョンのソースコードの ZIP アーカイブを含む S3 オブジェクトへのリンク。

最大サイズ: 50 MB。

タイプ：文字列

必須: False

templateUrl

アプリケーションのパッケージ化された AWS SAM テンプレートへのリンク。

タイプ: 文字列
必須: True

creationTime

このリソースが作成された日時。

タイプ：文字列
必須: True

parameterDefinitions

アプリケーションでサポートされるパラメータタイプの配列。

タイプ: ParameterDefinition タイプの配列
必須: True

requiredCapabilities

特定のアプリケーションをデプロイする前に指定する必要がある値のリスト。一部のアプリケーションには、新しい AWS Identity and Access Management（IAM) ユーザーを作成するなど、 AWS アカウントのアクセス許可に影響を与える可能性のあるリソースが含まれている場合があります。このようなアプリケーションの場合は、このパラメータを指定して、それらの機能を明示的に認識する必要があります。

有効な値は、CAPABILITY_IAM、CAPABILITY_NAMED_IAM、CAPABILITY_RESOURCE_POLICY、および CAPABILITY_AUTO_EXPAND のみです。

次のリソースでは、 CAPABILITY_IAMまたは を指定する必要がありますCAPABILITY_NAMED_IAM: AWS::IAM::Group、AWS::IAM::InstanceProfile、AWS::IAM::Policy、および AWS::IAM::Role。アプリケーションに IAM リソースがある場合、CAPABILITY_IAM または CAPABILITY_NAMED_IAM のいずれかを指定できます。アプリケーションにカスタム名を持つ IAM リソースがある場合は、CAPABILITY_NAMED_IAM を指定する必要があります。

次のリソースで
はCAPABILITY_RESOURCE_POLICYAWS::Lambda::Permission、、AWS::IAM:Policy、、AWS::ApplicationA
よび を指定する必要がありますAWS::SNS::TopicPolicy。

1 つまたは複数のネストされたアプリケーションが含まれているアプリケーションで
は、CAPABILITY_AUTO_EXPAND を指定する必要があります。

アプリケーションテンプレートに前述のリソースが含まれている場合、デプロイする前にアプリケーションに関連付けられたすべてのアクセス許可を確認することをお勧めします。機能を必要とするアプリケーションにこのパラメータを指定しないと、呼び出しは失敗します。

タイプ: Capability タイプの配列
必須: True

resourcesSupported

このアプリケーションに含まれるすべての AWS リソースが、取得するリージョンでサポートされているかどうかを示します。

タイプ：ブール値
必須: True

以下も参照してください。

言語固有の AWS SDKs とリファレンスのいずれかでこの API を使用する方法の詳細については、以下を参照してください。

CreateApplicationVersion

- AWS コマンドラインインターフェイス
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python

- AWS SDK for Ruby v3

ドキュメント履歴

- API バージョン: 最新
- ドキュメントの最終更新日: 2020 年 3 月 10 日

以下の表には、AWS Serverless Application Repository デベロッパーガイドの各リリースにおける重要な変更点が説明されています。このドキュメントの更新に関する通知については、RSS フィードでサブスクライブできます。

変更	説明	日付
アプリケーションの共有とアクセス制限の更新	AWS Organization のアカウントへのアプリケーションの共有、および AWS アカウントおよび AWS Organizations の公開アプリケーションへのアクセスを制限するサポートが追加されました。組織内のユーザーとアプリケーションを共有するその他の例については、「AWS Serverless Application Repositoryアプリケーションポリシーの例」を参照してください。公開アプリケーションへのアクセスを制限する例については、「AWS Serverless Application Repository アイデンティティベースのポリシーの例」を参照してください。	2020 年 3 月 10 日
新しいサポート対象リソース	サポートされるリソースが増えました。サポートされるリソースの詳細なリストについては、サポートされる AWS	2020 年 1 月 17 日

	リソースのリストを参照してください。	
中国地域	AWS Serverless Application Repository が中国リージョンの北京および寧夏で利用可能になりました。AWS Serverless Application Repository のリージョンおよびエンドポイントの詳細については、https://docs.aws.amazon.com/general/latest/gr/rande.htmlのAWS 全般のリファレンスリージョンとエンドポイントを参照してください。	2020 年 1 月 15 日
他の AWS のサービスとの整合性のためにセキュリティセクションを更新しました。	詳細については、「セキュリティ」を参照してください。	2020 年 1 月 2 日

アプリケーションの公開プロセスの簡略化	AWS SAM CLI の新しい `sam publish` コマンドは、AWS Serverless Application Repository でサーバレスアプリケーションを公開するプロセスをシンプル化します。end-to-endサンプルアプリケーションのダウンロードと公開に関するチュートリアルについては、「クイックスタート:アプリケーションの公開」を参照してください。AWS クラウドで開発済みおよびテスト済みのアプリケーションを発行する手順については、AWS SAM CLI を使用したアプリケーションの発行を参照してください。	2018 年 12 月 21 日
ネストアプリケーションとレイヤーのサポート	ネストされたアプリケーションとレイヤーのサポートを追加しました。これには、サポート対象の AWS サービスとアプリケーション機能の承認の更新が含まれます。	2018 年 11 月 29 日

カスタム IAM ロールとリソースポリシーによるアプリケーションの公開	カスタム IAM ロールとリソースポリシーを使用してアプリケーションを発行するサポートを追加しました。これには、AWS Serverless Application Repository デベロッパーガイドのアプリケーションの消費ワークフローおよびアプリケーションの発行ワークフローの更新と、サポート対象の AWS リソースおよび API リファレンスの更新が含まれます。	2018 年 11 月 16 日
ポリシーテンプレートの更新	AWS Serverless Application Repository デベロッパーガイドのサポートされるポリシーテンプレートの更新	2018 年 9 月 26 日
ドキュメンテーションの更新	認証とアクセスコントロールトピックが AWS Serverless Application Repository デベロッパーガイドに追加されました。	2018 年 7 月 2 日

公開リリース	14 の AWS リージョンで利用可能になった、AWS Serverless Application Repository のパブリックリリース。AWSAWS Serverless Application RepositoryAWS Serverless Application Repositoryが利用できるリージョンとエンドポイントの詳細については、の「リージョンとエンドポイント」を参照してくださいAWS 全般のリファレンス。	2018 年 2 月 20 日
新しいガイド	これは AWS Serverless Application Repository デベロッパーガイドの初回のプレビューリリースです。	2017 年 11 月 30 日

AWS 用語集

AWS の最新の用語については、「AWS の用語集リファレンス」の「AWS 用語集」を参照してください。

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。