デベロッパーガイド

AWS Serverless Application Repository

AWS Serverless Application Repository: デベロッパーガイド

Table of Contents

AWS Serverless Application Repositoryとは

AWS Serverless Application Repository を使用すると、開発者や企業は 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検索、デプロイ、公開できます。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 マネジメントコンソール、、 AWS SAM コマンドラインインターフェイス (AWS SAM CLI)、または AWS SDKs を使用してコードをアップロードします。コードとともに、 AWS Serverless Application Model (AWS SAM) テンプレートとも呼ばれるシンプルなマニフェストファイルをアップロードします。詳細については AWS SAM、「 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

- サンプルアプリケーションを に発行するチュートリアルについては AWS Serverless Application Repository、「」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。
- からアプリケーションをデプロイする手順については AWS Serverless Application Repository、「」を参照してくださいアプリケーションをデプロイする方法。

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 AWS Serverless Application Repository を使用してサンプルサーバーレスアプリケーションをダウンロード、構築、テストし、 に公開する手順について説明します。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 関数がローカルで呼び出されても、 AWS クラウド内の AWS リソースに対して読み取りと書き込みを行うことに注意してください。
3. パッケージ化する。Lambda 関数に満足したら、 を使用して Lambda 関数、 AWS SAM テンプレート、および依存関係を CloudFormation デプロイパッケージにバンドルしますsam package。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。このステップを完了する AWS Serverless Application Repository と、 でアプリケーションを表示し、 を使用して AWS クラウドにデプロイできます AWS Serverless Application Repository。

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、 は 関数 AWS Lambda を実行し、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 2 つの AWS リソースを定義する AWS SAM テンプレート。GET オペレーションを使用する API Gateway サービスと 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 ユーザーの 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。
- CLI (コマンドラインインターフェイス) AWS SAM がインストールされ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 がインストールされ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次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は Python で記述され、 `init` コマンドでランタイムを指定します。 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 AWS Lambda 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

ローカルマシンに AWS SAM アプリケーションが置かれたので、以下の手順に従って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、 CLI AWS SAM 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. 必要なアプリケーション情報を提供する`Metadata`セクションを AWS SAM テンプレートファイルに追加します。 AWS SAM テンプレート`Metadata`の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 に発行するには AWS Serverless Application Repository

- 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。他の AWS アカウントがアプリケーションを表示およびデプロイできるようにするには、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。

- でアプリケーションを表示する AWS Serverless Application Repository – `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への AWS Serverless Application Repository 直接リ

ンクが含まれます。 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するか、アカウントの特定のリストにアクセス許可を付与する必要があります AWS 。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 CLI 「」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。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Metadataのセクションsam packageと CLI AWS SAM の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

サーバーレスアプリケーションを に公開すると AWS Serverless Application Repository、他のユーザーがそれを見つけてデプロイできるようになります。

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。機能の使用 AWS SAM と確認の詳細については、「」を参照してくださいAWS SAM で を使用する AWS Serverless Application Repository。

サーバーレスアプリケーションを発行するには、、 AWS SAM コマンドラインインターフェイス (AWS SAM CLI) AWS マネジメントコンソール、または AWS SDK を使用します。にアプリケーションを公開する手順の詳細については AWS Serverless Application Repository、「」を参照してくださいアプリケーションを発行する方法。

アプリケーションを公開すると、最初はプライベートに設定されます。つまり、アプリケーションを作成した 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、プライベート共有 (特定の AWS アカウントセットとのみ共有) またはパブリック共有 (全員と共有) に設定する必要があります。

アプリケーションを発行してパブリックに設定する AWS Serverless Application Repository と、このサービスはすべてのリージョンのコンシューマーがアプリケーションを利用できるようにします。コンシューマーがアプリケーションを最初に公開したリージョン以外のリージョンにパブリックアプリケーションをデプロイすると、 はアプリケーションのデプロイアーティファクトを送信先リージョンの Amazon S3 バケット AWS Serverless Application Repository 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を更新して、代わりに送信先リージョンの Amazon S3 バケット内のファイルを参照します。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**
>
> プライベートアプリケーションとプライベート共有アプリケーションは、作成された AWS リージョンでのみ使用できます。パブリック共有アプリケーションは、すべての AWS リージョンで利用できます。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 で を使用する AWS Serverless Application Repository

AWS Serverless Application Model (AWS SAM) は、 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フレームワークです AWS。 AWS SAM を使用してサーバーレスアプリケーションを構築する方法の詳細については、 AWS Serverless Application Model デベロッパーガイドを参照してください。

に公開されるアプリケーションを構築するときは AWS Serverless Application Repository、サポートされている一連の AWS リソースとポリシーテンプレートの使用を検討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。

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

は、多くの AWS SAM および CloudFormation リソースで構成されるサーバーレスアプリケーション AWS Serverless Application Repository をサポートしています。でサポートされている AWS リソースの完全なリストを確認するには AWS Serverless Application Repository、「」を参照してください サポートされている AWS リソースのリスト。

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::InstanceProfile、AWS::IAM::Policy、および AWS::IAM::Role。
- リソースポリシー: AWS::Lambda::LayerVersionPermission、AWS::Lambda::Permission、AWS::Events::EventBusPolic 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、Lambda 関数のアクセス許可をアプリケーションで使用されるリソースにスコープする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 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。

サポートされている AWS リソースのリスト

これは、 でサポートされている AWS リソースの完全なリストです AWS Serverless Application Repository。

- `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、 CLI または AWS Serverless Application Repository を使用してサーバーレスアプリケーションを AWS SAM に発行する手順について説明します AWS マネジメントコンソール。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 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 AWS Serverless Application Repository 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「 CLI AWS SAM を使用したアプリケーションの公開」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、 を使用して AWS マネジメントコンソール に新しいアプリケーションを発行する方法について説明します AWS Serverless Application Repository。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

にアプリケーションを公開する前に AWS Serverless Application Repository、以下が必要です。

- 有効な AWS アカウント。
- 使用される AWS リソースを定義する valid 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 「テンプレートの基本」を参照してください。
- の `package` コマンドを使用して AWS CloudFormation 作成したアプリケーションのパッケージ AWS CLI。このコマンドは、 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、 CloudFormation ドキュメントの「 パッケージ」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. Amazon S3 コンソール (https://console.aws.amazon.com/s3/) 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. [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。必ず `Resource`要素でバケット名を、 `Condition`要素で AWS アカウント ID を置き換えてくだ

さい。Condition 要素の 式は、指定された AWS アカウントからアプリケーションにアクセスするアクセス許可 AWS Serverless Application Repository のみを持つようにします。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。

JSON

```
{
    "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

次の手順を使用して AWS Serverless Application Repository 、 で新しいアプリケーションを作成します。

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

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])?$";
ホームページ	FALSE	アプリケーションに関する詳細情報が含まれた URL。例えば、アプリケーションの GitHub リポジトリの場所などです。
説明	正	アプリケーションの説明。 最小長: 1 最大長 = 256。
ラベル	FALSE	検索結果でアプリケーションを見つけやすくするためのラベル。 最小長 = 1。最大長 = 127。ラベルの最大数:10。 パターン: "^[a-zA-Z0-9+\\-_:\\/@]+$";

プロパティ	必要	説明
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。注意: ドロップダウンでライセンスを選択すると、[ライセンス] テキストボックスの内容が置き換えられ、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	使用される AWS リソースを定義する valid 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。 AWS アカウントを共有するコンシューマーのみが、プライベートアプリケーションをデプロイするアクセス許可を持っています。
- プライベート共有 – パブリッシャーが特定の AWS アカウントのセットまたは AWS 組織内の AWS アカウントと明示的に共有したアプリケーション。コンシューマーには、自分の 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アクセス許可があります。詳細については AWS Organizations、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 に公開すると AWS Serverless Application Repository、デフォルトでプライベートに設定されます。このセクションでは、アプリケーションを特定の AWS アカウントまたは AWS 組織とプライベートに共有する方法、または全員とパブリックに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、2 つのオプションがあります。1) 特定の AWS アカウントまたは AWS 組織内の AWS アカウントと共有するか、2) 全員とパブリックに共有します。詳細については AWS Organizations、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 AWS 組織内の特定の AWS アカウント (複数可) またはアカウントと共有するには

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. [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**
>
> 組織と共有している場合は、 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。自分がメンバーではない AWS 組織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、今後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確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。

を介した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 AWS CLI 、 `put-application-policy` コマンドを使用して、プリンシパルとして共有する AWS アカウント (複数可) を指定するアクセス許可を付与します。

CLI を使用してアプリケーションを共有する方法の詳細については、 AWS 「」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。

アプリケーションの共有解除

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AWS 。

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 組織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で共有解除アプリケーションオペレーションを実行できます。

   > **Note**
   >
   > アプリケーションが「アプリケーションの共有解除」オペレーションで AWS Organization から共有解除された場合、そのアプリケーションを AWS Organization と再度共有することはできません。

   詳細については AWS Organizations、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

を通じてアプリケーションの共有を解除するには AWS マネジメントコンソール、他の AWS アカウントと共有してい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. [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。

を介したアクセス許可の削除 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 AWS CLI、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プライベートにしたり、別の AWS アカウントのセットと共有したりできます。

CLI を使用したアクセス許可の変更の詳細については、 AWS 「」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。

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管理アカウントが AWS Organization からアプリケーションを共有解除する

を通じて AWS Organization からアプリケーションの共有を解除するには AWS マネジメントコンソール、管理アカウントのユーザーが以下を実行できます。

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。

を通じて AWS 組織からアプリケーションを共有解除する管理アカウント AWS CLI

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が `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の共有を解除します。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、*`organization-id`* は Organization ID です AWS 。

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

アプリケーションの削除

AWS マネジメントコンソール または AWS SAM CLI AWS Serverless Application Repository を使用して、 からアプリケーションを削除できます。

アプリケーションの削除 (コンソール)

を使用して公開されたアプリケーションを削除するには AWS マネジメントコンソール、次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. [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)

を使用して公開されたアプリケーションを削除するには AWS CLI、 `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、 CLI または AWS Serverless Application Repository 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 SAM に発行する方法について説明します AWS マネジメントコンソール。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「 CLI AWS SAM を使用したアプリケーションの公開」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. [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	使用される AWS リソースを定義する valid 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

の検証済み作成者 AWS Serverless Application Repository 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 AWS 誠実にレビューし、リクエスタの ID が要求どおりであることを確認した作成者です。

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

serverlessrepo-verified-author@amazon.com に E メールを送信 AWS Serverless Application Repository することで、 で検証済み作成者としての承認をリクエストできます。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 (GitHub または LinkedIn プロファイルなど)

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 アカウントには複数の作成者がいる可能性があるため、異なる作成者名を持つアプリケーションではバッジは表示されません。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 Lambda レイヤーを簡単に共有できます。

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。これにより、レイヤーのコピーをユーザーの AWS アカウントで作成できます。

1. レイヤーをリソースとして含む AWS SAM テンプレート、つまり `AWS::Serverless::LayerVersion`または `AWS::Lambda::LayerVersion`リソース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 に公開し AWS Serverless Application Repository、 (パブリックまたはプライベートに) 共有します。
3. 顧客はアプリケーションをデプロイし、自分の AWS アカウントに Layer のコピーを作成します。顧客は、クライアントアプリケーションの AWS アカウントで Layer の Amazon リソースネーム (ARN) を参照できるようになりました。

例

以下は、共有する Lambda レイヤーを含むアプリケーションの AWS SAM テンプレートの例です。

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

```
    Value: !Ref SharedLayer
```

顧客が からアプリケーションをデプロイすると AWS Serverless Application Repository、 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。パブリックサイトにアクセスして、 AWS アカウントなしで公開されているアプリケーションを参照できます。または、 AWS Lambda コンソール内からアプリケーションを参照することもできます。

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。作成者は、 AWS がリクエスタから提供された情報を合理的かつ慎重なサービスプロバイダーとして誠実にレビューし、リクエスタの ID が要求どおりであることを確認した場合、検証済み作成者と見なされます。

からアプリケーションをデプロイする前に AWS Serverless Application Repository、以下のトピックを参照して、アプリケーションのデプロイのアクセス許可とアプリケーション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

にアプリケーションをデプロイするには AWS Serverless Application Repository、そ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。
- プライベート共有 – パブリッシャーが特定の AWS アカウントセットと明示的に共有したアプリケーション。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 アカウントとプライベートに共有されたアプリケーション、パブリックに共有された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、アプリケーションがパブリックに共有されているが、親アプリケーションを作成した AWS アカウントとのみプライベートに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、 AWS アカウントに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ない場合、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、 はアプリケーションのテンプレートで IAM ロール、 AWS リソースポリシー、およびテンプレートが作成すると指定するネストされたアプリケーション AWS Serverless Application Repository をチェックします。フルアクセスの IAM ロールなどの IAM リソースは、 AWS アカウント内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。これを確実に行うには、 がユーザーに代わってアプリケーションを AWS Serverless Application Repository デプロイする前に、アプリケーションに 機能が含まれ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。

`CAPABILITY_IAM` または `CAPABILITY_NAMED_IAM`
は、AWS::IAM::Group、AWS::IAM::InstanceProfile、AWS::IAM::Policy、および AWS::IAM::Role の各リソースに対して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。

CAPABILITY_RESOURCE_POLICY
は、AWS::Lambda::LayerVersionPermission、AWS::Lambda::Permission、AWS::Events::EventBusPolicy、および AWS::SNS::TopicPolicy の各リソースに対して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)

利用可能なアプリケーションは、AWS Serverless Application Repository ウェブサイト AWS Serverless Application Repository の または Lambda コンソール (タブの関数の作成ページ AWS Serverless Application Repository) 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」セクションのチェックボックスをオンにします。

アプリケーションの機能の表示 (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 AWS CLI、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 マネジメントコンソール または AWS Serverless Application Repository を使用して からサーバーレスアプリケーションをデプロイする手順について説明します AWS CLI。

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 Serverless Application Repository を使用して から新しいアプリケーションをデプロイする方法について説明します AWS マネジメントコンソール。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

次の手順を使用して、 でアプリケーションを検索、設定 AWS Serverless Application Repository 、デプロイします。

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

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すると、そのアクセス許可、機能、 AWS 顧客によってデプロイされた回数などの詳細が表示されます。

アプリケーションのデプロイ先の 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とアプリケーションリソースを表示します。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. [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されたリソースを確認および管理できます。

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 Serverless Application Repository を使用して から新しいアプリケーションをデプロイする方法について説明します AWS CLI。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)

を使用してアプリケーションの機能を確認するには AWS CLI、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ます。 次の AWS CLI 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。 AWS SDKs で 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. 変更セットを作成します。 CloudFormation 変更セットを作成するときは、必要な機能のセットを指定する必要があります。たとえば、次の AWS CLI コマンドを使用して、その機能を確認してアプリケーションをデプロイします。

   ```
   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。 AWS SDKs 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、次の AWS CLI コマンドは、カスタム名と 1 つ以上のネストされたアプリケーションを持つ AWS::IAM::Role リソース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. 変更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
   ```

   AWS SDKs 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

を使用して以前にデプロイしたアプリケーションを削除するには AWS Serverless Application Repository、 CloudFormation スタックの削除と同じ手順に従います。

- AWS マネジメントコンソール: を使用してアプリケーションを削除するには AWS マネジメントコンソール、「 ユーザーガイド」の「 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 CloudFormation
- 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

からアプリケーションをデプロイしたら AWS Serverless Application Repository、更新できます。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 マネジメントコンソール または を使用してアプリケーションの新しいバージョンをデプロイする方法について説明します AWS CLI。

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。特に、 はアプリケーション名の AWS Serverless Application Repository 先頭`serverlessrepo-`に付加されます。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。特に、 はスタック名の AWS Serverless Application Repository 先頭serverlessrepo-に 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。

のセキュリティ AWS Serverless Application Repository

のクラウドセキュリティが最優先事項 AWS です。お客様は AWS 、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を活用できます。

セキュリティは、 AWS お客様とお客様の間の責任共有です。責任共有モデルでは、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 – クラウドで AWS 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 AWS 責任があります。 AWS また、 では、安全に使用できるサービスも提供しています。「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。
- クラウドのセキュリティ – お客様の責任は、使用する AWS サービスによって決まります。また、ユーザーは、データの機密性、会社の要件、適用される法律や規制など、その他の要因についても責任を負います。

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 AWS Serverless Application Repository ように を設定する方法について説明します。また、 AWS Serverless Application Repository リソースのモニタリングや保護に役立つ他の AWS サービスの使用方法についても説明します。

トピック

のデータ保護 AWS Serverless Application Repository

責任 AWS 共有モデル、 でのデータ保護に適用されます AWS Serverless Application Repository。このモデルで説明されているように、 AWS はすべての 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があります AWS クラウド。ユーザーは、このインフラストラクチャでホストされ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もユーザーの責任となります。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に関する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。

データ保護の目的で、認証情報を保護し AWS アカウント 、 AWS IAM アイデンティティセンターまたは AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定することをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 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 は必須ですが、TLS 1.3 を推奨します。
- で API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。CloudTrail 証跡を使用して AWS アクティビティをキャプチャする方法については、「 AWS CloudTrail ユーザーガイド」のCloudTrail 証跡の使用」を参照してください。
- AWS 暗号化ソリューションと、その中のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。
- Amazon Macie などの高度な管理された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 AWS を介して にアクセスするときに FIPS 140-3 検証済み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) 140-3」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を、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含めないことを強くお勧めします。これは、コンソール AWS Serverless Application Repository 、API、または SDK を使用して AWS CLIまたは他の AWS のサービス を操作する場合も同様です。 AWS SDKs タグ、または名前に使用される自由記述のテキストフィールドに入力したデータは、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 を提供する場合は、そのサーバーへのリクエストを有効に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 エンドポイントは、HTTPS 経由の安全な接続のみをサポートします。、 AWS SDK AWS マネジメントコンソール、または AWS Serverless Application Repository API を使用してリソースを管理する AWS Serverless Application Repository 場合、すべての通信は Transport Layer Security (TLS) で暗号化されます。

API エンドポイントの完全なリストについては、AWS 『』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。

保管時の暗号化

は AWS Serverless Application Repository 、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、 にアップロードするファイルを暗号化します。

の Identity and Access Management AWS Serverless Application Repository

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 リソースの使用を許可する (アクセス許可を付与する) かを制御します。IAM は、追加料金なしで使用できる AWS のサービス 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

- オーディエンス
- 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
- と AWS Serverless Application Repository IAM の連携方法
- AWS Serverless Application Repository ID 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 Repository API アクセス許可: 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

オーディエンス

AWS Identity and Access Management (IAM) の使用方法は、ロールによって異なります。

- サービスユーザー - 機能にアクセスできない場合は、管理者にアクセス許可をリクエストします (「AWS Serverless Application Repository Identity and Access のトラブルシューティング」を参照)。
- サービス管理者 - ユーザーアクセスを決定し、アクセス許可リクエストを送信します (「と AWS Serverless Application Repository IAM の連携方法」を参照)
- IAM 管理者 - アクセスを管理するためのポリシーを作成します (「AWS Serverless Application Repository ID ベースのポリシーの例」を参照)

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。、IAM ユーザー AWS アカウントのルートユーザー、または IAM ロールを引き受けることで認証される必要があります。

AWS IAM アイデンティティセンター (IAM Identity Center)、シングルサインオン認証、Google/Facebook 認証情報などの ID ソースからの認証情報を使用して、フェデレーティッド ID としてサインインできます。サインインの詳細については、「AWS サインイン ユーザーガイド」の「AWS アカウントにサインインする方法」を参照してください。

プログラムによるアクセスの場合、 は SDK と CLI AWS を提供してリクエストを暗号化して署名します。詳細については、「IAM ユーザーガイド」の「API リクエストに対するAWS 署名バージョン 4」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、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 AWS アカウント ルートユーザーと呼ばれる 1 つのサインインアイデンティティから始めます。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。ルートユーザー認証情報を必要とするタスクについては、「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、特定の個人やアプリケーションに対する特定のアクセス許可を持つアイデンティティです。長期認証情報を持つ IAM ユーザーの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。詳細については、IAM ユーザーガイドの「ID プロバイダーとのフェデレーションを使用して にアクセスする必要がある AWS」を参照してください。

IAM グループは、IAM ユーザーの集合を指定し、大量のユーザーに対するアクセス許可の管理を容易にします。詳細については、「IAM ユーザーガイド」の「IAM ユーザーに関するユースケース」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を持つアイデンティであり、一時的な認証情報を提供します。ユーザーから IAM ロール (コンソール) に切り替えるか、 または API オペレーションを呼び出すことで、ロールを引き受けることができます。 AWS CLI AWS 詳細については、「IAM ユーザーガイド」の「ロールを引き受けるための各種方法」を参照してください。

IAM ロールは、フェデレーションユーザーアクセス、一時的な IAM ユーザーのアクセス許可、クロスアカウントアクセス、クロスサービスアクセス、および Amazon EC2 で実行するアプリケーションに役立ちます。詳細については、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。

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、ID AWS またはリソースにアタッチします。ポリシーは、ID またはリソースに関連付けられたときにアクセス許可を定義します。 は、プリンシパル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。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、ポリシーを使用して、どのプリンシパルがどのリソースに対して、どのような条件でアクションを実行できるかを定義することで、誰が何にアクセスできるかを指定します。

デフォルトでは、ユーザーやロールにアクセス許可はありません。IAM 管理者は IAM ポリシーを作成してロールに追加し、このロールをユーザーが引き受けられるようにします。IAM ポリシーは、オペレーションの実行方法を問わず、アクセス許可を定義します。

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のポリシーは、アイデンティティ (ユーザー、グループ、またはロール) にアタッチできる JSON アクセス許可ポリシードキュメントです。これらのポリシーは、アイデンティティがどのリソースに対してどのような条件下でどのようなアクションを実行できるかを制御します。アイデンティティベースポリシーの作成方法については、IAM ユーザーガイド の カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する 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、インラインポリシー (単一の ID に直接埋め込む) または管理ポリシー (複数の ID にアタッチされたスタンドアロンポリシー) にすることができます。管理ポリシーと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「管理ポリシーとインラインポリシーのいずれかを選択する」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。例としては、IAM ロール信頼ポリシーや Amazon S3 バケットポリシーなどがあります。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。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。

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アクセス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、および Amazon VPC は AWS WAF、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、より一般的なポリシータイプによって付与されるアクセス許可の上限を設定できる追加のポリシータイプをサポートしています。

- アクセス許可の境界 – アイデンティティベースのポリシーで IAM エンティティに付与することのできるアクセス許可の数の上限を設定します。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 (SCP) - AWS Organizations内の組織または組織単位の最大のアクセス許可を指定します。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。
- リソースコントロールポリシー (RCP) – は、アカウント内のリソースで利用できる最大数のアクセス許可を定義します。詳細については、「AWS Organizations ユーザーガイド」の「リソースコントロールポリシー (RCP)」を参照してください。
- セッションポリシー – ロールまたはフェデレーションユーザーの一時セッションを作成する際にパラメータとして渡される高度なポリシーです。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アクセス許可を理解するのがさらに難しくなります。が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。

と AWS Serverless Application Repository IAM の連携方法

IAM を使用して へのアクセスを管理する前に AWS Serverless Application Repository、 で使用できる IAM 機能を理解しておく必要があります AWS Serverless Application Repository。

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AWS 「IAM と連携する のサービス」を参照してください。

トピック

- AWS Serverless Application Repository ID 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 タグに基づいた認可
- AWS Serverless Application Repository IAM ロール

AWS Serverless Application Repository ID 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。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "CreateApplication",
            "Effect": "Allow",
            "Action": [
                "serverlessrepo:CreateApplication"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CreateApplicationVersion",
            "Effect": "Allow",
            "Action": [
                "serverlessrepo:CreateApplicationVersion"
            ],
            "Resource": "arn:aws:serverlessrepo:us-
east-1:111122223333: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、ワイルドカード文字 (*) をResource値として指定して、すべての AWS Serverless Application Repository リソースserverlessrepo:CreateApplicationに対する AWS Serverless Application Repository アクションのアクセス許可を付与します。
- 2 番目のステートメントは、 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serverlessrepo:CreateApplicationVersionに対する AWS Serverless Application

Repository アクションの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 リソースを示す表については、「」を参照してくださいAWS Serverless Application Repository API アクセス許可: アクションとリソースのリファレンス。

アクション

管理者は JSON AWS 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。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アクセス許可を付与するポリシーで使用されます。

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 AWS Serverless Application Repository を使用しますserverlessrepo:。たとえば、 SearchApplications API オペレーションで AWS Serverless Application Repository AWS Serverless Application Repository インスタンスを実行するアクセス許可を付与するには、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。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 AWS Serverless Application Repository 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コンマで区切ります。

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

ワイルドカード (*) を使用して複数アクションを指定できます。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確認するには、IAM ユーザーガイドの「 で定義されるアクション AWS Serverless Application Repository」を参照してください。

リソース

管理者は JSON AWS 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 ポリシー要素はアクションが適用されるオブジェクトを指定します。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。リソースレベルのアクセス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": "*"
```

では AWS Serverless Application Repository、プライマリ AWS リソースは application AWS Serverless Application Repository . AWS Serverless Application Repository applications です。次の表に示すように、アプリケーションには一意の Amazon リソースネーム (ARNs) 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 (ARNs AWS 「サービス名前空間」を参照してください。

以下は、すべての AWS リソースに対する `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です。現在の実装では、 AWS Serverless Application Repository は一部の API アクションで AWS リソース ARNs (リソースレベルのアクセス許可とも呼ばれます) を使用した特定の AWS リソースの識別をサポートしていません。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。

JSON

```
{
    "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 リソースを示す表については、「」を参照してくださいAWS Serverless Application Repository API アクセス許可: アクションとリソースのリファレンス。

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、一部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、「」を参照してくださいAWS Serverless Application Repository ID 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

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 principalOrg が AWS Serverless Application Repository 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たポリシーにアクセス許可を追加できます。 AWS Serverless Application Repository アプリケーションにアタッチされたアクセス許可ポリシーは、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の拡張機能です。プライマリリソースはアプリケーションです AWS

Serverless Application Repository 。 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許可を管理できます。

AWS Serverless Application Repository アプリケーションポリシーは、主にパブリッシャーが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コンシューマーに付与するために使用されます。また、これらのアプリケーションの詳細を検索して表示するための などの関連オペレーションも使用します。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。
- プライベート共有 – パブリッシャーが特定の AWS アカウントまたは AWS Organizations のセットと明示的に共有したアプリケーション。アカウント AWS 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アクセス許可があります。
- 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 SDKs、または を使用します AWS マネジメントコンソール。

例

AWS Serverless Application Repository アプリケーションポリシーの管理例については、「」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 タグに基づいた認可

AWS Serverless Application Repository は、タグに基づくリソースまたはアクションへのアクセスの制御をサポートしていません。

AWS Serverless Application Repository IAM ロール

IAM ロールは、特定のアクセス許可を持つ AWS アカウント内のエンティティです。

での一時的な認証情報の使用 AWS Serverless Application Repository

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を取得するには、AssumeRole や GetFederationToken などの AWS STS API オペレーションを呼び出します。

では、一時的な認証情報の使用 AWS Serverless Application Repository がサポートされています。

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 ID 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 マネジメントコンソール、 AWS CLI、または AWS API を使用してタスクを実行することはできません。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権限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、管理者はそれらの権限が必要な IAM ユーザーまたはグループにそのポリシーをアタッチする必要があります。

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
- AWS Serverless Application Repository コンソールの使用
- 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
- お客様が管理するポリシーの例

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ポリシーは非常に強力です。アカウント内の 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では、 AWS アカウントのコスト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 コンソールには、 AWS Serverless Application Repository アプリケーションを検出および管理するための統合環境が用意されています。コンソールには、「」で説明されている API 固有のアクセス許可に加えて、 AWS Serverless Application Repository アプリケーションを管理するためのアクセス許可が必要になることが多い機能とワークフローが用意されていますAWS Serverless Application Repository API アクセス許可: アクションとリソースのリファレンス。

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、「」を参照してくださいお客様が管理するポリシーの例。

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アタッチ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の作成方法を示します。このポリシーには、コンソールで、または AWS CLI または AWS API を使用してプログラムでこのアクションを実行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、コンソールには、ユーザーが属する特定の AWS リージョンで作成された AWS アカウント内の AWS Serverless Application Repository アプリケーションのリストが表示されます。

JSON

```
{
    "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できます。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。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ViewApplication",
            "Effect": "Allow",
            "Action": [
                "serverlessrepo:GetApplication",
                "serverlessrepo:ListApplicationVersions"
            ],
            "Resource": "arn:aws:serverlessrepo:us-
east-1:111122223333: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

ユーザーに AWS Serverless Application Repository アプリケーションを作成するアクセス許可を付与する場合は、次のポリシーに示すように、 `serverlessrepo:CreateApplication` および `serverlessrepo:CreateApplicationVersions`オペレーションにアクセス許可を付与する必要があります。

JSON

```
{
    "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。

JSON

```
{
    "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。

JSON

```
{
    "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、作成者、説明、バージョン、その他の設定情報など、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、ユーザーは次の AWS Serverless Application Repository オペレーションに対するアクセス許可を持っている必要があります。

JSON

```
{
    "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。

JSON

```
{
    "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。は と同じ基盤となるデプロイメカニズム AWS Serverless Application Repository を使用するため CloudFormation、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

アプリケーションが AWS アカウントとプライベートに共有されている場合、デフォルトでは、そのアカウントのすべてのユーザーは、同じアカウント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。

JSON

```
{
    "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。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 AWS Serverless Application Repository」を参照してください。

JSON

```
{
    "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 組織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アタッチされたアクセス許可ポリシーは、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 principalOrg が AWS Serverless Application Repository 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、 の主要な AWS リソースです AWS Serverless Application Repository。 AWS Serverless Application Repository アプリケーションポリシーは、主にパブリッシャーがアプリケーションをデプロイするアクセス許可をコンシューマーに付与するために使用されます。また、これらのアプリケーションを検索して詳細を表示するなどの関連オペレーションも行います。

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。AWS アカウントを共有するコンシューマーのみが、プライベートアプリケーションをデプロイするアクセス許可を持っています。
- プライベート共有 – パブリッシャーが特定の AWS アカウントのセットまたは AWS AWS 組織内のアカウントと明示的に共有したアプリケーション。コンシューマーには、自分の AWS アカウントまたは AWS 組織と共有されているアプリケーションをデプロイするアクセス許可があります。AWS 組織の詳細については、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**
>
> プライベート共有アプリケーションの場合、 はプリンシパルとしてAWS アカウント AWS Serverless Application Repository のみをサポートします。パブリッシャーは、 AWS アカウント内のすべてのユーザーを単一のグループとして AWS Serverless Application Repository アプリケーションに付与または拒否できます。パブリッシャーは、 AWS アカウント内の個々のユーザーをアプリケーションに AWS Serverless Application Repository 付与または拒否することはできません。

を使用してアプリケーションのアクセス許可を設定する手順については AWS マネジメントコンソール、「」を参照してくださいアプリケーションの共有。

AWS CLI および の例を使用してアプリケーションのアクセス許可を設定する手順については、以下のセクションを参照してください。

アプリケーションのアクセス許可 (AWS CLI および AWS SDKs)

AWS CLI または AWS SDKs を使用してアプリケーションのアクセス許可 AWS Serverless Application Repository 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。
CreateCloudFormationTemplate	アプリケーションの 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。を使用してアクセス許可を付与する方法については AWS マネジメントコンソール、「」を参照してくださいアプリケーションの共有。

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

アプリケーションを別の特定のアカウントと共有し、他のユーザーと共有しないようにするには、プリンシパルとして共有する AWS アカウント ID を指定します。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**
>
> プライベート共有アプリケーションは、アプリケーションが作成されたのと同じ 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プライベートにできるため、誰とも共有されず、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 組織内の他のアカウントからのアクセス許可がアプリケーションのデプロイからも削除されます。

```
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 組織内のすべて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 組織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 組織の詳細については、AWS Organizations 「 ユーザーガイド」を参照してください。

> **Note**
>
> AWS アカウントがメンバーである AWS 組織のみを指定できます。自分がメンバーではない AWS 組織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 組織と共有するには、今後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 組織内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 組織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**
>
> AWS アカウントがメンバーである AWS 組織のみを指定できます。自分がメンバーではない AWS 組織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 組織と共有するには、今後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 Repository API アクセス許可: 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。、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許可を付与できる対応するアクション、およびアクセス許可を付与できる 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。

運用	[URI]	Method	AWS リソース (ARNs)
オペレーション: ListApplications 必要なアクセス権限: serverlessrepo:ListApplications	/applications	GET	*
オペレーション: CreateApplication 必要なアクセス権限: serverlessrepo:CreateApplication	/applications	POST	*
オペレーション: GetApplication 必要なアクセス権限: serverlessrepo:GetApplication	/applications/*application-id*	GET	arn:aws:serverlessrepo:*region*:*account-id* :applications/*application-name*
オペレーション: DeleteApplication 必要なアクセス権限: serverlessrepo:DeleteApplication	/applications/*application-id*	DELETE	arn:aws:serverlessrepo:*region*:*account-id* :applications/*application-name*
オペレーション: UpdateApplication	/applications/*application-id*	PATCH	arn:aws:serverlessrepo:*region*:*account-id* :applicat

運用	[URI]	Method	AWS リソース (ARNs)
必要なアクセス権限 : serverlessrepo:UpdateApplication			ions/*application-name*
オペレーション: CreateCloudFormationChangeSet 必要なアクセス権限 : serverlessrepo:CreateCloudFormationChangeSet	/applications/*application-id*/changesets	POST	arn:aws:serverlessrepo:*region*:*account-id* :applications/*application-name*
オペレーション: GetApplicationPolicy 必要なアクセス権限 : serverlessrepo:GetApplicationPolicy	/applications/*application-id*/policy	GET	arn:aws:serverlessrepo:*region*:*account-id* :applications/*application-name*
オペレーション: PutApplicationPolicy 必要なアクセス権限 : serverlessrepo:PutApplicationPolicy	/applications/*application-id*/policy	PUT	arn:aws:serverlessrepo:*region*:*account-id* :applications/*application-name*
オペレーション: ListApplicationVersions 必要なアクセス権限 : serverlessrepo:ListApplicationVersions	/applications/*application-id*/versions	GET	arn:aws:serverlessrepo:*region*:*account-id* :applications/*application-name*

運用	[URI]	Method	AWS リソース (ARNs)
オペレーション: CreateApplicationVersion 必要なアクセス権限: 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: ListApplicationDependencies 必要なアクセス許可: serverlessrepo:ListApplicationDependencies	/applications/*application-id*/dependencies	GET	arn:aws:serverlessrepo:*region*:*account-id* :applications/*application-name*
オペレーション: SearchApplications 必要なアクセス権限: 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、 と 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

- でアクションを実行する権限がありません AWS Serverless Application Repository
- iam:PassRole を実行する権限がない

- 管理者として、他のユーザーに へのアクセスを許可したい AWS Serverless Application Repository
- AWS 自分の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

でアクションを実行する権限がありません AWS Serverless Application Repository

にアクションを実行する権限がないと AWS マネジメントコンソール 通知された場合は、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:PassRole を実行する権限がない

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、既存のロールをそのサービスに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と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。Mary には、ロールをサービスに渡すアクセス許可がありません。

```
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。サインイン資格情報を提供した担当者が管理者です。

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

他のユーザーにアクセスを許可するには AWS Serverless Application Repository、アクセスを必要とするユーザーまたはアプリケーションにアクセス許可を付与する必要があります。 AWS IAM アイデンティティセンター を使用してユーザーとアプリケーションを管理する場合は、アクセスレベルを定義するアクセス許可セットをユーザーまたはグループに割り当てます。アクセス許可セットは、ユーザーまたはアプリケーションに関連付けられている IAM ロールに自動的に IAM ポリシーを作成して割り当てます。詳細については、「AWS IAM アイデンティティセンター 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。

IAM アイデンティティセンターを使用していない場合は、アクセスを必要としている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。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。アクセス許可が付与されたら、ユーザーまたはアプリケーション開発者に認証情報を提供します。これらの認証情報を使用して AWSにアクセスします。IAM ユーザー、グループ、ポリシー、アクセス許可の作成の詳細については、「IAM ユーザーガイド」の「IAM アイデンティティ」と「IAM のポリシーとアクセス許可」を参照してください。

AWS 自分の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と AWS Serverless Application Repository IAM の連携方法。

- 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におけるロールとリソースベースのポリシーの使用方法の違いについては、「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。

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

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上で重要な部分です。マルチポイント障害が発生した場合は、その障害をより簡単にデバッグできるように、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。 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ツールがいくつか AWS 用意されています。次に例を示します。

AWS CloudTrail ログ

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

トピック

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

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

AWS Serverless Application Repository は と統合されています。これは AWS CloudTrail、 のユーザー、ロール、または のサービスによって実行されたアクションを記録する AWS サービスです AWS Serverless Application Repository。CloudTrail は、 のすべての API コールをイベント AWS Serverless Application Repository としてキャプチャします。キャプチャされた呼び出しには、AWS Serverless Application Repository コンソールからの呼び出しと 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。

証跡を作成する場合は、 AWS Serverless Application Repositoryのイベントを含めた CloudTrail イベントの Amazon S3 バケットへの継続的な配信を有効にすることができます。証跡を設定しない場合でも、CloudTrail コンソールの [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。

AWS Serverless Application Repository CloudTrail の情報

CloudTrail は、 AWS アカウントの作成時にアカウントで有効になります。でアクティビティが発生すると AWS Serverless Application Repository、そのアクティビティはイベント履歴の他の AWS サービスイベントとともに CloudTrail イベントに記録されます。 AWS アカウントで最近のイベントを表示、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。

のイベントなど、 AWS アカウントのイベントの継続的な記録については AWS Serverless Application Repository、証跡を作成します。証跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。デフォルトでは、コンソールで証跡を作成すると、証跡はすべての AWS リージョンに適用されます。証跡は、 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をログに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、CloudTrail ログで収集されたイベントデータをさらに分析して処理するように、他の AWS サービスを設定できます。詳細については、次を参照してください:

- 証跡の作成のための概要
- CloudTrail がサポートするサービスと統合
- CloudTrail 用 Amazon SNS 通知の構成

- 複数のリージョンから CloudTrail ログファイルを受け取るおよび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 によってログに記録され、AWS Serverless Application Repository リソースページに記録されます。たとえば CreateApplication、UpdateApplications、および ListApplications の各オペレーションへのコールは、CloudTrail ログファイル内にエントリを生成します。

各イベントまたはログエントリには、誰がリクエストを生成したかという情報が含まれます。アイデンティティ情報は、以下を判別するのに役立ちます。

- リクエストがルートまたは AWS Identity and Access Management (IAM) ユーザー認証情報を使用して行われたかどうか。
- リクエストがロールまたはフェデレーションユーザーのテンポラリなセキュリティ認証情報を使用して行われたかどうか。
- リクエストが別の AWS サービスによって行われたかどうか。

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について

「トレイル」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ます。CloudTrail のログファイルは、単一か複数のログエントリを含みます。イベントは、任意の出典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、リクエストパラメータなどに関する情報が含まれます。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 アクションを示す CloudTrail ログエントリです。

```
{
    "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"
}
```

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

サードパーティーの監査者は、複数の コンプライアンスプログラム AWS Serverless Application Repository の一環として のセキュリティと AWS 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、AWS「コンプライアンスプログラムによる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。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、AWS「アーティファクトでのレポートのダウンロード」を参照してください。

を使用する際のお客様のコンプライアンス責任 AWS Serverless Application Repository は、お客様のデータの機密性、貴社のコンプライアンス目的、適用される法律および規制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提供しています。

- セキュリティと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デプロイする手順について説明します AWS。
- AWS コンプライアンスリソース – このワークブックとガイドのコレクションは、お客様の業界や地域に適用される場合があります。
- AWS Config – この AWS サービスは、リソース設定が内部プラクティス、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します。
- AWS Security Hub CSPM – この AWS サービスは、 内のセキュリティ状態を包括的に把握 AWS し、セキュリティ業界標準とベストプラクティスへの準拠を確認するのに役立ちます。

の耐障害性 AWS Serverless Application Repository

AWS 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ています。 AWS リージョンは、低レイテンシー、高スループット、高度に冗長なネットワークで接続された複数の物理的に分離されたアベイラビリティーゾーンを提供します。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、およびスケーラビリティが優れています。

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 「 グローバルインフラストラクチャ」を参照してください。

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

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 がインフラストラクチャ AWS 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環境を AWS 設計するには、「Security Pillar AWS Well-Architected Framework」の「Infrastructure Protection」を参照してください。

AWS が発行した API コールを使用して、ネットワーク AWS Serverless Application Repository 経由で 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 が必須で、TLS 1.3 をお勧めします。

- DHE (楕円ディフィー・ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。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。

インターフェイスエンドポイント (AWS PrivateLink) AWS Serverless Application Repository を使用した へのアクセス

を使用して AWS PrivateLink 、VPC と の間にプライベート接続を作成できます AWS Serverless Application Repository。インターネットゲートウェイ、NAT デバイス、VPN 接続、または Direct Connect 接続を使用せずに、VPC 内にある AWS Serverless Application Repository かのように にアクセスできます。VPC 内のインスタンスは AWS Serverless Application Repositoryにアクセスするためにパブリック IP アドレスを必要としません。

このプライベート接続を確立するには、 AWS PrivateLinkを利用したインターフェイスエンドポイントを作成します。インターフェイスエンドポイントに対して有効にする各サブネットにエンドポイントネットワークインターフェイスを作成します。これらは、 AWS Serverless Application Repository 宛てのトラフィックのエントリポイントとして機能するリクエスタ管理型ネットワークインターフェイスです。

詳細については、「AWS PrivateLink ガイド」の「AWS PrivateLinkから AWS のサービス にアクセスする」を参照してください。

に関する考慮事項 AWS Serverless Application Repository

のインターフェイスエンドポイントを設定する前に AWS Serverless Application Repository、「AWS PrivateLink ガイド」の「考慮事項」を参照してください。

AWS Serverless Application Repository は、インターフェイスエンドポイントを介したすべての API アクションの呼び出しをサポートしています。

のインターフェイスエンドポイントを作成する AWS Serverless Application Repository

Amazon VPC コンソールまたは AWS Command Line Interface () AWS Serverless Application Repository を使用して、 のインターフェイスエンドポイントを作成できますAWS CLI。詳細については、「AWS PrivateLink ガイド」の「インターフェイスエンドポイントを作成」を参照してください。

次のサービス名 AWS Serverless Application Repository を使用して のインターフェイスエンドポイントを作成します。

```
com.amazonaws.region.serverlessrepo
```

インターフェイスエンドポイントのプライベート DNS を有効にすると、リージョンのデフォルト DNS 名を使用して、 AWS Serverless Application Repository への API リクエストを実行できます。例えば、`serverlessrepo.us-east-1.amazonaws.com`。

インターフェイスエンドポイントのエンドポイントポリシーを作成する

エンドポイントポリシーは、インターフェイスエンドポイントにアタッチできる IAM リソースです。デフォルトのエンドポイントポリシーでは、インターフェイスエンドポイント AWS Serverless Application Repository を介した へのフルアクセスが許可されます。VPC AWS Serverless Application Repository から に許可されるアクセスを制御するには、カスタムエンドポイントポリシーをインターフェイスエンドポイントにアタッチします。

エンドポイントポリシーは以下の情報を指定します。

- アクションを実行できるプリンシパル (AWS アカウント、IAM ユーザー、IAM ロール)。
- 実行可能なアクション。
- このアクションを実行できるリソース。

詳細については、AWS PrivateLink ガイドのControl access to services using endpoint policies (エンドポイントポリシーを使用してサービスへのアクセスをコントロールする)を参照してください。

例: AWS Serverless Application Repository アクションの VPC エンドポイントポリシー

以下は、カスタムエンドポイントポリシーの例です。このポリシーをインターフェイスエンドポイントにアタッチすると、すべてのリソースのすべてのプリンシパルに対して、リストされた AWS Serverless Application Repository アクションへのアクセスが許可されます。次の例では、VPC エンドポイントを介してアプリケーションを作成するアクセス許可をすべてのユーザーに付与します。

```
{
   "Statement": [
      {
         "Principal": "*",
         "Effect": "Allow",
         "Action": [
```

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

AWS Serverless Application Repository クォータ

AWS Serverless Application Repository には、 AWS アカウントが各 AWS リージョンに持つことができるパブリックアプリケーションの数に対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 (AWS アカウントごと、 AWS リージョンごと)	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
コードパッケージの無料の Amazon S3 ストレージ (AWS アカウントごと、 AWS リージョンごと)	5 GB
アプリケーションポリシーの長さ	6,144 文字

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

を使用すると AWS Serverless Application Repository、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可能性があります。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**
>
> のアプリケーション AWS Serverless Application Repository は、 を使用してデプロイされます CloudFormation。 CloudFormation 問題のトラブルシューティングの詳細については、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 クォータについては、「」を参照してくださいAWS Serverless Application Repository クォータ。

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 リソースと CloudFormation スタックへのアクセス許可が必要です。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。

このような問題が発生した場合は、 AWS Identity and Access Management (IAM) ポリシーを確認し、必要なアクセス許可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」を参照してください。

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、 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 CloudFormation スタックがないことを確認してください。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 プレミアムサポートをお持ちの場合は、 サポートでテクニカルサポートケースを作成できますAWS。

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。

オペレーション

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、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。
- CreateApplicationVersion

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

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

  AWS CloudFormation テンプレートを作成します。
- 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 AWS CloudFormation テンプレートを取得します。
- 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

  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。

- 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`	String	False	返される項目の合計数。

名前	型	必須	説明
nextToken	String	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	Success
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	Success

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	None	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誠実に確認し、リクエスタの ID が要求どおり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

アプリケーションに関する詳細情報が含まれた URL。例えば、アプリケーションの GitHub リポジトリの場所などです。

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

アプリケーションに関する詳細情報が含まれた URL。例えば、アプリケーションの GitHub リポジトリの場所などです。

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

アプリケーションに関する詳細情報が含まれた URL。例えば、アプリケーションの GitHub リポジトリの場所などです。

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. CloudFormation validates the parameter value as a number。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。 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。

CAPABILITY_IAM または CAPABILITY_NAMED_IAM は、AWS::IAM::Group、AWS::IAM::InstanceProfile、AWS::IAM::Policy、および AWS::IAM::Role の各リソースに対して指定する必要があります。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。

以下のリソースでは、CAPABILITY_RESOURCE_POLICY: AWS::Lambda::Permission、AWS::IAM:Policy、AWS::ApplicationAutoScaling::ScalingPolicy、AWS::S3::Bu 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: 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

関連情報

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 V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- 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 V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- 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	String	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	String	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	Success
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。

ステータスコード	レスポンスモデル	説明
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	String	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	None	Success

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。
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	String	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	None	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	String	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	Success
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誠実に確認し、リクエスタの ID が要求どおり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

アプリケーションに関する詳細情報が含まれた URL。例えば、アプリケーションの GitHub リポジトリの場所などです。

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. CloudFormation validates the parameter value as a number。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。 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

アプリケーションに関する詳細情報が含まれた URL。例えば、アプリケーションの GitHub リポジトリの場所などです。

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。

CAPABILITY_IAM または CAPABILITY_NAMED_IAM は、AWS::IAM::Group、AWS::IAM::InstanceProfile、AWS::IAM::Policy、および AWS::IAM::Role の各リソースに対して指定する必要があります。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。

以下のリソースでは、CAPABILITY_RESOURCE_POLICY: AWS::Lambda::Permission、AWS::IAM:Policy、AWS::ApplicationAutoScaling::ScalingPolicy、AWS::S3::Buc 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: 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

関連情報

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 V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- 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 V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2

- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- 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 V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- 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の AWS CloudFormation 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	String	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	Success
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	String	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	None	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、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

CreateCloudFormationTemplate によって返される UUID。

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。

CAPABILITY_IAM または CAPABILITY_NAMED_IAM は、AWS::IAM::Group、AWS::IAM::InstanceProfile、AWS::IAM::Policy、および AWS::IAM::Role の各リソースに対して指定する必要があります。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。

以下のリソースでは、CAPABILITY_RESOURCE_POLICY: AWS::Lambda::Permission、AWS::IAM:Policy、AWS::ApplicationAutoScaling::ScalingPolicy、AWS::S3::Bu
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 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

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、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 CloudFormation RollbackConfiguration データータイプに対応します。

rollbackTriggers

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

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 データータイプの同じ名前のコンテンツに対応します。

タイプ: 整数
必須: False

RollbackTrigger

このプロパティは CloudFormation RollbackTrigger データータイプに対応します。

arn

このプロパティは、AWS CloudFormation RollbackTrigger データー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

type

このプロパティは、AWS CloudFormation RollbackTrigger データー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

Tag

このプロパティは 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

関連情報

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 V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- 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	String	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	String	False	ページ分割を始める場所を指定するトークン。
maxItems	String	False	返される項目の合計数。
semanticVersion	String	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	Success
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

ステータスコード	レスポンスモデル	説明
		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	String	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	None	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

関連情報

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 V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- 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	String	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	Success
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	String	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	Success
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	String	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	None	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

関連情報

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 V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- 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 V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- 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	String	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	Success
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	String	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	None	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

CreateCloudFormationTemplate によって返される UUID。

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

関連情報

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 V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2

- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- 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	String	True	アプリケーションの Amazon リソースネーム (ARN) です。
templateId	String	True	CreateCloudFormationTemplate によって返される UUID。 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	Success
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	String	True	アプリケーションの Amazon リソースネーム (ARN) です。
templateId	String	True	CreateCloudFormationTemplate によって返される UUID。

名前	型	必須	説明
			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	None	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

CreateCloudFormationTemplate によって返される UUID。

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

関連情報

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 V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- 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	String	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	None	Success
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

ステータスコード	レスポンスモデル	説明
		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	String	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	None	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

関連情報

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 V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2

- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- 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	String	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	String	False	返される項目の合計数。
nextToken	String	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	Success
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	String	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	None	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

関連情報

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 V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- 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	String	True	アプリケーションの Amazon リソースネーム (ARN) です。
semanticVersion	String	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	Success
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	String	True	アプリケーションの Amazon リソースネーム (ARN) です。
semanticVersion	String	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	None	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 raw パッケージ 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. CloudFormation validates the parameter value as a number。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。 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。

CAPABILITY_IAM または CAPABILITY_NAMED_IAM は、AWS::IAM::Group、AWS::IAM::InstanceProfile、AWS::IAM::Policy、および AWS::IAM::Role

の各リソースに対して指定する必要があります。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。

以下のリソースでは、CAPABILITY_RESOURCE_POLICY: AWS::Lambda::Permission、AWS::IAM:Policy、AWS::ApplicationAutoScaling::ScalingPolicy、AWS::S3::Buc 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: 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

関連情報

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 V2
- AWS SDK for .NET V4
- AWS SDK for C++
- AWS SDK for Go v2

- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- 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を受け取るには、RSS フィードにサブスクライブできます。

変更	説明	日付
アプリケーションへのアクセスの共有と制限の更新	AWS Organization のアカウントとアプリケーションを共有し、 AWS アカウント AWS と Organizations のパブリック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
新しくサポートされたリソース	サポートされるリソースが増えました。サポートされているリソースの完全なリストについては、「サポートされ	2020 年 1 月 17 日

	ている AWS リソースのリスト」を参照してください。	
中国リージョン	AWS Serverless Application Repository が中国リージョン、北京、寧夏で利用可能になりました。 AWS Serverless Application Repository 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。	2020 年 1 月 15 日
他の AWS サービスとの一貫性を保つため、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素化	AWS SAM CLI の新しいsam publishコマンドは、 でサーバーレスアプリケーションを公開するプロセスを簡素化します AWS Serverless Application Repository。サンプルアプリケーションのダウンロードと発行に関する詳細な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 クラウドで既に開発およびテスト済みのアプリケーションを公開する手順については、「 CLI AWS SAM を使用したアプリケーションの公開」を参照してください。	2018 年 12 月 21 日

ネストされた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、サポートされている AWS リソースの更新とアプリケーション機能の確認が含まれます。	2018 年 11 月 29 日
カスタム IAM ロールとリソースポリシーを使用した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、「デベロッパーガイド」の「アプリケーションの消費と公開」ワークフローの更新と、サポートされている AWS リソースと API リファレンスの更新が含まれます。AWS Serverless Application Repository	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
ドキュメントの更新	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

パブリックリリース	のパブリックリリース AWS Serverless Application Repository。14 AWS リージョンで利用可能になりました。AWS Serverless Application Repository が利用可能な AWS リージョンと AWS Serverless Application Repository 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

最新の AWS 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。