aws

Developer Guide

AWS Serverless Application Repository

AWS Serverless Application Repository: Developer Guide

Table of Contents

What Is the AWS Serverless Application Repository?

The AWS Serverless Application Repository makes it easy for developers and enterprises to quickly find, deploy, and publish serverless applications in the AWS Cloud. For more information about serverless applications, see Serverless Computing and Applications on the AWS website.

You can easily publish applications, sharing them publicly with the community at large, or privately within your team or across your organization. To publish a serverless application (or app), you can use the AWS Management Console, the AWS SAM command line interface (AWS SAM CLI), or AWS SDKs to upload your code. Along with your code, you upload a simple manifest file, also known as an AWS Serverless Application Model (AWS SAM) template. For more information about AWS SAM, see the *AWS Serverless Application Model Developer Guide*.

The AWS Serverless Application Repository is deeply integrated with the AWS Lambda console. This integration means that developers of all levels can get started with serverless computing without needing to learn anything new. You can use category keywords to browse for applications such as web and mobile backends, data processing applications, or chatbots. You can also search for applications by name, publisher, or event source. To use an application, you simply choose it, configure any required fields, and deploy it with a few clicks.

In this guide, you can learn about the two ways to work with the AWS Serverless Application Repository:

- Publishing Applications – Configure and upload applications to make them available to other developers, and publish new versions of applications.
- Deploying Applications – Browse for applications and view information about them, including source code and readme files. Also install, configure, and deploy applications of your choosing.

Next Steps

- For a tutorial about publishing a sample application to the AWS Serverless Application Repository, see Quick Start: Publishing Applications.
- For instructions about deploying applications from the AWS Serverless Application Repository, see How to Deploy Applications.

Quick Start: Publishing Applications

This guide walks you through the steps to download, build, test and publish an example serverless application to the AWS Serverless Application Repository using AWS SAM CLI. You can use this example application as a starting point for developing and publishing your own serverless application.

Overview

The following steps outline how to download, build and publish a sample serverless application:

1. **Initialize**. Download a sample application from template using `sam init`.
2. **Test Locally**. Test the application locally using `sam local invoke` and/or `sam local start-api`. Note that with these commands, even though your Lambda function is invoked locally, it reads from and writes to AWS resources in the AWS Cloud.
3. **Package**. When you're satisfied with your Lambda function, bundle the Lambda function, AWS SAM template, and any dependencies into an AWS CloudFormation deployment package using `sam package`. In this step you will also include information about the application that will be uploaded to AWS Serverless Application Repository.
4. **Publish**. Publish the application to the AWS Serverless Application Repository using `sam publish`. At the conclusion of this step, you're able to view your application in AWS Serverless Application Repository and deploy it to the AWS Cloud using AWS Serverless Application Repository.

The example Hello World Application in the next section walks you through these steps in building and publishing a serverless application.

Hello World Application

In this exercise, you download and test a Hello World serverless application that represents a simple API backend. It has an Amazon API Gateway endpoint that supports a GET operation and a Lambda function. When a GET request is sent to the endpoint, API Gateway invokes the Lambda function. Then, AWS Lambda executes the function, which simply returns a `hello world` message.

The application has the following components:

- An AWS SAM template that defines two AWS resources for the Hello World application: an API Gateway service with a GET operation, and a Lambda function. The template also defines the mapping between the API Gateway GET operation and the Lambda function.
- Application code that's written in Python.

Before You Begin

Make sure that you have the required setup for this exercise:

- You must have an AWS account with an IAM user that has administrator permissions. See Set Up an AWS Account.
- You must have the AWS SAM CLI (command line interface) installed. See Installing the AWS SAM CLI.
- You must have version 1.16.77 or later of the AWS CLI installed. See Installing the AWS Command Line Interface.

Step 1: Initialize the Application

In this section, you download the sample application, which consists of an AWS SAM template and application code.

To initialize the application

1. Run the following command at an AWS SAM CLI command prompt.

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

2. Review the contents of the directory that the command created (`sam-app/`):

   - `template.yaml` – Defines two AWS resources that the Hello World application needs: a Lambda function and an API Gateway endpoint that supports a GET operation. The template also defines mapping between the two resources.
   - Content related to the Hello World application code:
     - `hello_world/` directory – Contains the application code, which returns `hello world` when you run it.

> **Note**
>
> For this exercise, the application code is written in Python, and you specify the runtime in the `init` command. AWS Lambda supports additional languages for creating application code. If you specify another supported runtime, the `init` command provides the Hello World code in the specified language, and a `README.md` file that you can follow along for that language. For information about supported runtimes, see Lambda Execution Environment and Available Libraries.

Step 2: Test the Application Locally

Now that you have the AWS SAM application on your local machine, follow the steps below to test it locally.

To test the application locally

1. Start the API Gateway endpoint locally. You must run the following command from the directory that contains the `template.yaml` file.

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

   The command returns an API Gateway endpoint, which you can send requests to for local testing.

2. Test the application. Copy the API Gateway endpoint URL, paste it in the browser, and choose **Enter**. An example API Gateway endpoint URL is `http://127.0.0.1:3000/hello`.

   API Gateway locally invokes the Lambda function that the endpoint is mapped to. The Lambda function executes in the local Docker container and returns `hello world`. API Gateway returns a response to the browser that contains the text.

Exercise: Change the message string

After successfully testing the sample application, you can experiment with making a simple modification: change the message string that's returned.

1. Edit the `/hello_world/app.py` file to change the message string from `'hello world'` to `'Hello World!'`.
2. Reload the test URL in your browser and observe the new string.

You will notice that your new code is loaded dynamically, without your having restart the `sam local` process.

Step 3: Package the Application

After testing your application locally, you use the AWS SAM CLI to create a deployment package and a packaged AWS SAM template.

> **Note**
>
> In the following steps, you create a .zip file for the contents of the `hello_world/` directory, which contains the application code. This .zip file is the **deployment package** for your serverless application. For more information, see Creating a Deployment Package (Python) in the *AWS Lambda Developer Guide*.

To create a Lambda deployment package

1. Add a `Metadata` section to your AWS SAM template file providing the required application information. For more information about the `Metadata` section of AWS SAM templates, see AWS SAM Template Metdata Section Properties in *AWS Serverless Application Model Developer Guide*.

   Here is an example `Metadata` section:

   ```
   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
```

The `LicenseUrl` and `ReadmeUrl` properties can either be references to local files (as in the example above), or they can be links to Amazon S3 buckets that already host these artifacts.

2. Create an S3 bucket in the location where you want to save the packaged code. If you want to use an existing S3 bucket, skip this step.

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

3. Create the Lambda function deployment package by running the following `package` AWS SAM CLI command.

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

The command does the following:

- Zips the contents of the `aws-sam/hello_world/` directory and uploads it to Amazon S3.
- Uploads the deployment package, README file, and LICENSE file to the Amazon S3 bucket specified by the `--s3-bucket` option.
- Outputs a new template file, called `packaged.yaml`, which you use in the next step to publish the application to AWS Serverless Application Repository. The `packaged.yaml` template file is similar to the original template file (`template.yaml`), but has a key difference—the `CodeUri`, `LicenseUrl`, and `ReadmeUrl` properties point to the Amazon S3 bucket and objects that contains the respective artifacts. The following snippet from an example `packaged.yaml` template file shows the `CodeUri` property:

```
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
```

```
...
```

Step 4: Publish the Application

Now that you've created the deployment package, you use it to publish the application to AWS Serverless Application Repository.

To publish the serverless application to the AWS Serverless Application Repository

- Execute the following command to publish the new application in AWS Serverless Application Repository with the first version created as 0.0.1.

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

> **Note**
>
> The application will be created as private by default. You must share the application before other AWS accounts will be allowed to view and deploy your application. See **Next Steps** below for more details about sharing your application.

Next Steps

Now that you have published your sample application, following are a few things you might want to do with it.

- **View Your Application in AWS Serverless Application Repository** – The output of the `sam publish` command will include a link to the AWS Serverless Application Repository directly to the detail page of your application. You can also go to the AWS Serverless Application Repository landing page and search for your application.
- **Share Your Application** – Because your application is set to private by default, it is not visible to other AWS Accounts. In order to share your application with others, you must either make it public or grant permission to a specific list of AWS Accounts. For information on sharing your application using the AWS CLI see AWS Serverless Application Repository Application

Policy Examples. For information on sharing your application using the console see Sharing an Application.

More Information

For more information about the `Metadata` section of AWS SAM templates, `sam package` and `sam publish` commands of AWS SAM CLI, see Publishing Applications Using AWS SAM CLI in the *AWS Serverless Application Model Developer Guide*.

Publishing Applications

When you publish a serverless application to the AWS Serverless Application Repository, you make it available for others to find and deploy.

You first define your application with an *AWS Serverless Application Model (AWS SAM) template.* When you define your application, you must consider whether consumers of your application will be required to acknowledge the application's capabilities. For more information about using AWS SAM and acknowledging capabilities, see Using AWS SAM with the AWS Serverless Application Repository.

You can publish serverless applications by using the AWS Management Console, the AWS SAM command line interface (AWS SAM CLI), or an AWS SDK. To learn more about the procedures for publishing applications to the AWS Serverless Application Repository, see How to Publish Applications.

When you publish your application, it's initially set to *private*, which means that it's only available to the AWS account that created it. To share your application with others, you must either set it to *privately shared* (shared only with a specific set of AWS accounts), or *publicly shared* (shared with everyone).

When you publish an application to the AWS Serverless Application Repository and set it to public, the service makes the application available to consumers in all Regions. When a consumer deploys a public application to a Region other than the Region in which the application was first published, the AWS Serverless Application Repository copies the application's deployment artifacts to an Amazon S3 bucket in the destination Region. It updates any resources in the AWS SAM template that use those artifacts to instead reference the files in the Amazon S3 bucket for the destination Region. Deployment artifacts can include Lambda function code, API definition files, and so on.

> **Note**
>
> *Private* and *privately shared* applications are only available in the AWS Region that they're created in. *Publicly shared* applications are available in all AWS Regions. To learn more about sharing applications, see AWS Serverless Application Repository Application Policy Examples.

Topics

Using AWS SAM with the AWS Serverless Application Repository

The AWS Serverless Application Model (AWS SAM) is an open-source framework that you can use to build serverless applications on AWS. For more information about using AWS SAM to build your serverless application, see the *AWS Serverless Application Model Developer Guide*.

When building applications that will be published to the AWS Serverless Application Repository, you must consider the set of supported AWS Resources and Policy Templates available to use. The sections below describe these topics in more detail.

Supported AWS Resources in the AWS Serverless Application Repository

The AWS Serverless Application Repository supports serverless applications that are composed of many AWS SAM and AWS CloudFormation resources. To see the complete list of AWS resources that are supported by AWS Serverless Application Repository, see List of Supported AWS Resources.

If you want to request support for an additional AWS resource, contact AWS Support.

Important

If your application template contains one of the following custom IAM roles or resource policies, your application doesn't show up in search results by default. Also, customers need to acknowledge the application's custom IAM roles or resource policies before they can deploy the application. For more information, see Acknowledging Application Capabilities. The list of resources that this applies to are:

- **IAM roles:** AWS::IAM::Group, AWS::IAM::InstanceProfile, AWS::IAM::Policy, and AWS::IAM::Role.

- **Resource policies:** AWS::Lambda::LayerVersionPermission, AWS::Lambda::Permission, AWS::Events::EventBusPolicy, AWS::IAM:Policy, AWS::ApplicationAutoScaling::ScalingPolicy, AWS::S3::BucketPolicy, AWS::SQS::QueuePolicy, and AWS::SNS:TopicPolicy.

If your application contains the AWS::Serverless::Application resource, customers need to acknowledge that the application contains a **nested application** before they can deploy the application. For more information about nested applications, see Nested Applications in the *AWS Serverless Application Model Developer Guide*. For more information about acknowledging capabilities, see Acknowledging Application Capabilities.

Policy Templates

AWS SAM provides you with a list of policy templates to scope the permissions of your Lambda functions to the resources that are used by your application. Using policy templates don't require additional customer acknowledgments to search, browse, or deploy the application.

For the list of standard AWS SAM policy templates, see AWS SAM Policy Templates in the *AWS Serverless Application Model Developer Guide*.

List of Supported AWS Resources

This is the complete list of AWS resources that are supported by the 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

How to Publish Applications

This section provides you with procedures for publishing your serverless application to the AWS Serverless Application Repository by using the AWS SAM CLI or the AWS Management Console. It also shows you how to share your application to allow others to deploy it, and deleting your application from the AWS Serverless Application Repository.

> **Important**
>
> The information that you enter when you publish an application isn't encrypted. This information includes data such as the author name. If you have personally identifiable information that you don't want to be stored or made public, we recommend that you don't enter this information when publishing your application.

Publishing an Application (AWS CLI)

The easiest way to publish an application to the AWS Serverless Application Repository is to use a set of AWS SAM CLI commands. For more information, see Publishing an Application Using the AWS SAM CLI in the *AWS Serverless Application Model (AWS SAM) Developer Guide*.

Publishing a New Application (Console)

This section shows you how to use the AWS Management Console to publish a new application to the AWS Serverless Application Repository. For instructions on publishing a new version of an existing application, see Publishing a New Version of an Existing Application.

Prerequisites

Before you publish an application to the AWS Serverless Application Repository, you need the following:

- A valid AWS account.
- A valid AWS Serverless Application Model (AWS SAM) template that defines the AWS resources that are used. For more information about AWS SAM templates, see AWS SAM Template Basics.
- A package for your application that you created by using the AWS CloudFormation `package` command for the AWS CLI. This command packages the local artifacts (local paths) that your AWS SAM template references. For more details, see package in the AWS CloudFormation documentation.
- A URL that points to your application's source code, in case you want to publish your application publicly.
- A readme.txt file. This file should describe how customers can use your application, and how to configure it before deploying it in their own AWS accounts.
- A license.txt file or a valid license identifier from the SPDX website. Note that a license is only required if you want to share your application publicly. If you're going to keep your application private or only share it privately, you don't need to specify a license.
- A valid Amazon S3 bucket policy that grants the service read permissions for artifacts that were uploaded to Amazon S3 when you packaged your application. To set this policy, follow these steps:

  1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
  2. Choose the Amazon S3 bucket that you used to package your application.
  3. Choose the **Permissions** tab.
  4. Choose the **Bucket Policy** button.
  5. Paste the following policy statement into the **Bucket policy editor**. Make sure to substitute your bucket name in the `Resource` element, and your AWS account ID in the `Condition`

element. The expression in the `Condition` element ensure AWS Serverless Application Repository only has permission to access applications from the specified AWS account. For more information about policy statements, see IAM JSON policy elements reference in the *IAM User Guide*.

```
{
    "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. Choose the **Save** button.

Procedure

Create a new application in the AWS Serverless Application Repository by using the following procedure.

To create a new application in the AWS Serverless Application Repository

1. Open the AWS Serverless Application Repository console and choose **Publish applications**.
2. On the **Publish an application** page, enter the following application information, and then choose **Publish application**:

Property	Required	Description
Application name	TRUE	The name of the application.

Property	Required	Description
		Minimum length=1. Maximum length=140. Pattern: "[a-zA-Z0-9\\-]+";
Author	TRUE	The name of the author publishing the application. Minimum length=1. Maximum length=127. Pattern: "^[a-z0-9](([a-z0-9]|-(?!-))*[a-z0-9])?$";
Home page	FALSE	A URL with more informati on about the application—for example, the location of your GitHub repository for the application.
Description	TRUE	The description of the application. Minimum length=1. Maximum length=256.
Labels	FALSE	The labels that improve the discovery of applications in search results. Minimum length=1. Maximum length=127. Maximum number of labels: 10. Pattern: "^[a-zA-Z0-9+\\-_:\\/@]+$";

Property	Required	Description
Spdx license (drop-down list)	FALSE	Choose a valid license identifier from the drop-down that contains licenses that are available on the SPDX website. Choosing an item in the drop-down populates the **License** text box below it. Note: Choosing a license in the drop-down replaces the contents of the **License** text box, and discards any manual edits that you have made.

Property	Required	Description
License	FALSE	Upload a .txt license file, or choose a license from the **Spdx license** drop-down described in the previous row. Choosing a license from the **Spdx license** drop-down automatically populates the **License** text box. You can manually edit the contents of this text box after uploading a license file or choosing one from the **Spdx license** drop-down. However, if another **Spdx license** is chosen from the drop-down, any manual edits that you have made are discarded. This is an optional field, but you must provide a license in order to share the application publicly.
Readme	FALSE	Upload the contents of the Readme file, which can be in text or markdown format. These contents are displayed on the applicati on's detail page in the AWS Serverless Applicati on Repository. You can manually edit the contents of this text box after uploading a file.

Property	Required	Description
Semantic version	FALSE	The semantic version of the application. For more information, see the Semantic Versioning website. You must provide a value for this property in order to make your application public.
Source code Url	FALSE	A link to a public repository for the source code of your application.
SAM template	TRUE	A valid AWS Serverless Application Model (AWS SAM) template that defines the AWS resources that are used.

Sharing an Application

Published applications can have permissions set in one of the three following categories:

- **Private (default)** – Applications that were created with the same account, and haven't been shared with any other AWS account. Only consumers that share your AWS account have permission to deploy private applications.
- **Privately shared** – Applications that the publisher has explicitly shared with a specific set of AWS accounts, or with AWS accounts in an AWS Organization. Consumers have permission to deploy applications that have been shared with their AWS account or AWS Organization. For more information about AWS Organizations, see the *AWS Organizations User Guide*.
- **Publicly shared** – Applications that the publisher has shared with everyone. All consumers have permission to deploy any publicly shared application.

After you have published an application to the AWS Serverless Application Repository, by default it is set to **private**. This section shows you how to share an application privately with specific AWS accounts or an AWS Organization, or share it publicly with everyone.

Sharing an Application Through the Console

You have two options for sharing your application with others: 1) Share it with specific AWS accounts or the AWS accounts within your AWS organization, or 2) Share it publicly with everyone. For more information about AWS Organizations, see the *AWS Organizations User Guide*.

Option 1: To share your application with specific AWS account(s) or accounts within your AWS organization

1. Open the AWS Serverless Application Repository console.
2. On the navigation pane, choose **Published Applications** to bring up the list of applications that you've created.
3. Choose the application that you want to share.
4. Choose the **Sharing** tab.
5. In the **Application policy statements** section, choose the **Create Statement** button.
6. In the **Statement Configuration** window fill out the fields based on how you want to share your application.

   > **Note**
   >
   > If you are sharing with an organization, you can only specify the organization that your AWS account is a member of. If you try to specify an AWS Organization that you are not a member of, an error will result.
   > To share your application with your AWS Organization, you must acknowledge that the `UnshareApplication` action will be added to your policy statement, in case the sharing needs to be revoked in the future.

7. Choose the **Save** button.

Option 2: To share your application publicly with everyone

1. Open the AWS Serverless Application Repository console.

2. On the navigation pane, choose **Published Applications** to bring up the list of applications that you've created.
3. Choose the application that you want to share.
4. Choose the **Sharing** tab.
5. In the **Public Sharing** section, choose the **Edit** button.
6. Under **Public sharing** choose the **Enabled** radio button.
7. In the text box type the name of your application, then choose the **Save** button.

> **Note**
>
> In order to share an application publicly, it must have both the `SemanticVersion` and `LicenseUrl` properties set.

Sharing an Application Through the AWS CLI

To share an application using the AWS CLI you grant permissions using the `put-application-policy` command to specify the AWS account(s) you want to share with as principals.

For more information about sharing your application by using the AWS CLI, see AWS Serverless Application Repository Application Policy Examples.

Unsharing an Application

There are two options for unsharing an application from an AWS Organization:

1. The publisher of the application can remove permissions using the `put-application-policy` command.
2. A user from the *management account* of an AWS Organization can perform an unshare application operation on any application shared with the organization, even if the application was published by a user from a different account.

> **Note**
>
> When an application is unshared from an AWS Organization with the "unshare application" operation, it cannot be shared with AWS Organization again.

For more information about AWS Organizations, see the *AWS Organizations User Guide*.

Publisher Removing Permissions

Publisher Removing Permissions Through the Console

To unshare an application through the AWS Management Console, you remove the policy statement that shares it with other AWS accounts. To do this, follow these steps:

1. Open the AWS Serverless Application Repository console.
2. Choose **Available Applications** in the left navigation pane.
3. Choose the application that you want to unshare.
4. Choose the **Sharing** tab.
5. In the **Application policy statements** section, select the policy statement that is sharing the application with the accounts that you want to unshare from.
6. Choose **Delete**.
7. A confirmation message will appear. Choose **Delete** again.

Publisher Removing Permissions Through the AWS CLI

To unshare an application through the AWS CLI, the publisher can remove or otherwise change permissions using the `put-application-policy` command to make the application private, or share with a different set of AWS accounts.

For more information about changing permissions using the AWS CLI, see AWS Serverless Application Repository Application Policy Examples.

Management account unsharing an application

Management account unsharing an application from an AWS Organization through the console

To unshare an application from an AWS Organization through the AWS Management Console, a user from the *management account* can do the following:

1. Open the AWS Serverless Application Repository console.
2. Choose **Available Applications** in the left navigation pane.

3. In the application's tile, choose **Unshare**.
4. In the unshare message box, confirm you want to unshare the application by entering the Organization ID and application name, then choosing **Save**.

Management account unsharing an application from an AWS Organization Through the AWS CLI

To unshare an application from an AWS Organization, a user from the *management account* can run the `aws serverlessrepo unshare-application` command.

The following command unshares an application from an AWS Organization, where *`application-id`* is the Amazon Resource Name (ARN) of the application, and *`organization-id`* is the AWS Organization ID:

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

Deleting an Application

You can delete applications from the AWS Serverless Application Repository by using either the AWS Management Console or the AWS SAM CLI.

Deleting an Application (Console)

To delete a published application through the AWS Management Console, do the following.

1. Open the AWS Serverless Application Repository console.
2. For **My Applications**, choose the application that you want to delete.
3. In the application's detail page, choose **Delete application**.
4. Choose **Delete application** to complete the deletion.

Deleting an Application (AWS CLI)

To delete a published application using the AWS CLI, run the `aws serverlessrepo delete-application` command.

The following command deletes an application, where *`application-id`* is the Amazon Resource Name (ARN) of the application:

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

Publishing a New Version of an Existing Application

This section shows you how to publish a new version of an existing application to the AWS Serverless Application Repository by using the AWS SAM CLI or the AWS Management Console. For instructions on publishing a new application, see How to Publish Applications.

Publishing a New Version of an Existing Application (AWS CLI)

The easiest way to publish a new version of an existing application is to use a set of AWS SAM CLI commands. For more information, see Publishing an Application Using the AWS SAM CLI in the *AWS Serverless Application Model (AWS SAM) Developer Guide*.

Publishing a New Version of an Existing Application (Console)

To publish a new version of an application that you have previously published, follow these steps:

1. Open the AWS Serverless Application Repository console.
2. In the navigation pane, choose **My Applications** to bring up the list of applications that you've created.
3. Choose the application that you want to publish a new version for.
4. Choose **Publish new version**.
5. In **Versions**, enter the following application information:

Property	Required	Description
Semantic version	TRUE	The semantic version of the application. For more information, see the Semantic Versioning website. You must provide a value for this property in order to make your application public.

Property	Required	Description
Source code Url	FALSE	A link to a public repository for the source code of your application.
SAM template	TRUE	A valid AWS Serverless Application Model (AWS SAM) template that defines the AWS resources that are used.

6. Choose **Publish version**.

Verified Author Badge

Verified authors in the AWS Serverless Application Repository are those for which AWS has made a good faith review, as a reasonable and prudent service provider, of the information provided by the requester, and has confirmed that the requester's identity is as claimed.

The applications of verified authors display a verified author badge, along with a link to the author's public profile. The verified author badge is displayed in both search results and on the application detail page.

Requesting a Verified Author Badge

You can request to be approved as a verified author in the AWS Serverless Application Repository by sending an email to serverlessrepo-verified-author@amazon.com. You need to provide the following information:

- Author name
- AWS account ID
- Publicly accessible profile link, such as your GitHub or LinkedIn profile

After submitting a request for a verified author badge, you can expect a response from AWS within a few days. You might be asked for additional information before your request is approved.

After your request is approved, you can expect that the verified author badge will be displayed for your applications within a day.

> **Note**
>
> The verified author badge is displayed for all applications that match both the AWS account and author name. Because AWS accounts can have multiple authors, badges aren't be displayed on applications that have a different author name. To have author badges displayed on applications with different author names, you must submit another request for that author.

Sharing Lambda Layers

If you've implemented functionality in a Lambda layer, you might want to share your layer without hosting a global instance of it. Sharing layers in this manner enables others to deploy an instance of your layer to their own account. This prevents client applications from depending on a global instance of your layer. The AWS Serverless Application Repository enables you to share Lambda layers in this manner easily.

For more information about Lambda layers, see AWS Lambda Layers in the *AWS Lambda Developer Guide*.

How It Works

The following are the steps for sharing your layer using the AWS Serverless Application Repository. This allows a copy of your layer to be created in the user's AWS account.

1. Define a serverless application with an AWS SAM template that includes your layer as a resource— that is, either an `AWS::Serverless::LayerVersion` or an `AWS::Lambda::LayerVersion` resource.
2. Publish your application to the AWS Serverless Application Repository, and share it (either publicly or privately).
3. A customer deploys your application, which creates a copy of your layer in their own AWS account. The customer can now reference the Amazon Resource Name (ARN) of the layer in their AWS account in their client application.

Example

The following is an example AWS SAM template for an application that contains the Lambda layer that you want to share:

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

When a customer deploys your application from the AWS Serverless Application Repository, a layer is created in their AWS account. The ARN of the layer looks something like the following:

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

The customer can now reference this ARN in their own client application, like in this example:

```
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
```

Deploying Applications

This section helps you learn how to find and deploy serverless applications that have been published to the AWS Serverless Application Repository. You can browse for applications that are publicly available without having an AWS account by visiting the public site. Alternatively, you can browse for applications from within the AWS Lambda console.

Some applications have a **verified author** badge, with a link to the author's profile. An author is considered a **verified author** when AWS has made a good faith review, as a reasonable and prudent service provider, of the information provided by the requester, and has confirmed that the requester's identity is as claimed.

Before deploying applications from the AWS Serverless Application Repository, see the following topics to learn about application deployment permissions and application capabilities.

Topics

Application Deployment Permissions

To deploy an application in the AWS Serverless Application Repository, you must have permission to do so. There are three categories of applications that you have permissions to deploy:

- **Private** – Applications that were created with the same account, and haven't been shared with any other account. You have permission to deploy applications that were created using your AWS account.
- **Privately shared** – Applications that the publisher has explicitly shared with a specific set of AWS accounts. You have permission to deploy applications that have been shared with your AWS account.
- **Publicly shared** – Applications that the publisher has shared with everyone. You have permission to deploy any publicly shared application.

You can only search and browse for applications that you have permissions for. These include applications that were created using your AWS account, privately shared with your AWS account, and publicly shared. All other applications aren't displayed for you.

> **Important**
>
> Applications that contain nested applications inherit the nested applications' sharing restrictions. For example, suppose an application is publicly shared, but it contains a nested application that's only privately shared with the AWS account that created the parent application. In this case, if your AWS account doesn't have permission to deploy the nested application, then you aren't able to deploy the parent application. For more information about nested applications, see Nested Applications in the *AWS Serverless Application Model Developer Guide*.

Application Capabilities: IAM Roles, Resource Policies, and Nested Applications

Before you can deploy an application, the AWS Serverless Application Repository checks the application's template for IAM roles, AWS resource policies, and nested applications that the template specifies that it should create. IAM resources, such as an IAM role with full access, can modify any resource in your AWS account. Therefore, we recommend that you review the permissions associated with the application before proceeding so that you don't unintentionally create resources with escalated permissions. To ensure that you've done so, you must acknowledge that the application contains capabilities before the AWS Serverless Application Repository can deploy the application on your behalf.

Applications can contain any of the following four capabilities: `CAPABILITY_IAM`, `CAPABILITY_NAMED_IAM`, `CAPABILITY_RESOURCE_POLICY`, and `CAPABILITY_AUTO_EXPAND`.

The following resources require you to specify `CAPABILITY_IAM` or `CAPABILITY_NAMED_IAM`: AWS::IAM::Group, AWS::IAM::InstanceProfile, AWS::IAM::Policy, and AWS::IAM::Role. If the application contains IAM resources with custom names, you must specify `CAPABILITY_NAMED_IAM`. For an example of how to specify capabilities, see Finding and Acknowledging Application Capabilities (AWS CLI).

The following resources require you to specify `CAPABILITY_RESOURCE_POLICY`: AWS::Lambda::LayerVersionPermission, AWS::Lambda::Permission, AWS::Events::EventBusPolicy,

AWS::IAM:Policy, AWS::ApplicationAutoScaling::ScalingPolicy, AWS::S3::BucketPolicy, AWS::SQS::QueuePolicy, and AWS::SNS::TopicPolicy.

Applications that contain one or more nested applications require you to specify `CAPABILITY_AUTO_EXPAND`. For more information about nested applications, see Nested Applications in the *AWS Serverless Application Model Developer Guide*.

Finding and Acknowledging Application Capabilities (Console)

You can find applications available in the AWS Serverless Application Repository on the AWS Serverless Application Repository website, or through the Lambda console (on the **Create Function** page under the AWS Serverless Application Repository tab).

Applications that require acknowledgment of capabilities for creating custom IAM roles or resource policies aren't shown in search results by default. To search for applications that contain these capabilities, you must select the **Show apps that create custom IAM roles or resource policies** check box.

You can review the capabilities of an application under the **Permissions** tab when you select the application. To deploy the application, you need to select the **I acknowledge this application creates custom IAM roles or resource policies** check box. If you don't acknowledge these capabilities, you see this error message: **Acknowledgement required. To deploy, check the box in Configure application parameters section**.

Viewing Application Capabilities (AWS CLI)

To view an application's capabilities using the AWS CLI, you first need the application's Amazon Resource Name (ARN). You can then execute the following command:

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

The requiredCapabilities response property contains the list of application capabilities that you need to acknowledge before you can deploy the application. Note that if the requiredCapabilities property is empty, the application has no required capabilities.

How to Deploy Applications

This section provides you with procedures for deploying serverless applications from the AWS Serverless Application Repository by using the AWS Management Console or the AWS CLI.

Deploying a New Application (Console)

This section shows you how to deploy a new application from the AWS Serverless Application Repository using the AWS Management Console. For instructions on deploying a new version of an existing application, see Updating Applications.

Browsing, Searching, and Deploying Applications

Find, configure, and deploy an application in the AWS Serverless Application Repository by using the following procedure.

To find and configure an application in the AWS Serverless Application Repository

1. Open the AWS Serverless Application Repository public home page, or open the AWS Lambda console. Choose **Create function**, and then choose **Browse serverless app repository**.
2. Browse or search for an application.

   > **Note**
   >
   > To show applications that contain custom IAM roles or resource policies, select the **Show apps that create custom IAM roles or resource policies** check box. For more information about custom IAM roles and resource policies, see Acknowledging Application Capabilities.

3. Choose an application to view details such as its permissions, capabilities, and the number of times it has been deployed by AWS customers.

   The deployment counts are shown for the AWS Region that you're trying to deploy the application in.
4. On the application detail page, view the application's permissions and application resources by viewing the AWS SAM template, license, and readme file. On this page, you can also find the **Source code URL** link for applications that are publicly shared. If the application includes any nested applications, you can also view details of the nested applications on this page.

5. Configure the application in the **Application settings** section. For guidance on configuring a particular application, see that application's readme file.

   For example, configuration requirements might include specifying the name of a resource that you want the application to have access to. Such a resource might be an Amazon DynamoDB table, an Amazon S3 bucket, or an Amazon API Gateway API.

6. Choose **Deploy**. Doing this takes you to the **Deployment status** page.

   > **Note**
   >
   > If the application has capabilities that require acknowledgement, you must select the **I acknowledge this application creates custom IAM roles or resource polices** check box before deploying the application. Otherwise, an error will result. For more information about custom IAM roles and resource policies, see Acknowledging Application Capabilities.

7. On the **Deployment status** page, you can view the progress of your deployment. While waiting for your deployment to complete, you can search and browse for other applications, and return to this page through the Lambda console.

After your application has been successfully deployed, you can review and manage the resources that have been created by using existing AWS tools.

Deploying a New Application (AWS CLI)

This section shows you how to deploy a new application from the AWS Serverless Application Repository by using the AWS CLI. For instructions on deploying a new version of an existing application, see Updating Applications.

Finding and Acknowledging Application Capabilities (AWS CLI)

To acknowledge an application's capabilities using the AWS CLI, follow these steps:

1. **Review the application's capabilities.** Use the following AWS CLI command to review an application's capabilities:

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

The requiredCapabilities response property contains the list of application capabilities that you need to acknowledge before you can deploy the application. You can also use the GetApplication API in the AWS SDKs to get this data.

2. **Create the changeset.** You must provide the set of required capabilities when you create the AWS CloudFormation changeset. For example, use the following AWS CLI command to deploy an application by acknowledging its capabilities:

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

The changeset ID is returned when this command is successfully executed. You need the changeset ID for the next step. You can also use the CreateCloudFormationChangeSet API in the AWS SDKs to create the changeset.

For example, the following AWS CLI command acknowledges an application that contains an AWS::IAM::Role resource with a custom name and one or more nested applications:

```
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. **Execute the changeset.** Executing the changeset actually performs the deployment. Provide the changeset ID that was returned when you created the changeset in the previous step.

The following example AWS CLI command executes the application changeset to deploy the application:

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

You can also use the ExecuteChangeSet API in the AWS SDKs to execute the changeset.

Deleting Application Stacks

To delete an application that you previously deployed using the AWS Serverless Application Repository, follow the same procedure as for deleting an AWS CloudFormation stack:

- **AWS Management Console**: To delete an application using the AWS Management Console, see Deleting a Stack on the AWS CloudFormation Console in the *AWS CloudFormation User Guide.*
- **AWS CLI**: To delete an application using the AWS CLI, see Deleting a Stack in the *AWS CloudFormation User Guide.*

Updating Applications

After you've deployed an application from the AWS Serverless Application Repository, you might want to update it. For example, you might want to change an application setting, or you might want to update the application to the latest version that was published.

The following sections describe how to deploy a new version of an application by using either the AWS Management Console or the AWS CLI.

Updating Applications (Console)

To update an application that you previously deployed, use the same procedure as deploying a new application, and *provide the same application name that you originally deployed it with*. In particular, the AWS Serverless Application Repository prepends `serverlessrepo-` to your application name. However, to deploy a new version of your application, you provide the original application name without `serverlessrepo-` prepended.

For example, if you deployed an application with the name `MyApplication`, the stack name would be `serverlessrepo-MyApplication`. To update that application, you would provide the name `MyApplication` again—do *not* specify the full stack name of `serverlessrepo-MyApplication`.

For all other application settings, you can either keep the values the same as the previous deployment, or provide new values.

Updating Applications (AWS CLI)

To update an application that you previously deployed, use the same procedure as deploying a new application, and *provide the same `--stack-name` that you originally deployed it with*. In

particular, AWS Serverless Application Repository prepends `serverlessrepo-` to your stack name. However, to deploy a new version of your application, you provide the original stack name without `serverlessrepo-` prepended.

For example, if you deployed an application with the stack name `MyApplication`, the stack name that is created would be `serverlessrepo-MyApplication`. To update that application, you would provide the name `MyApplication` again—do *not* specify the full stack name of `serverlessrepo-MyApplication`.

Security in the AWS Serverless Application Repository

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS compliance programs. To learn about the compliance programs that apply to the AWS Serverless Application Repository, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using the AWS Serverless Application Repository. The following topics show you how to configure the AWS Serverless Application Repository to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your AWS Serverless Application Repository resources.

Topics

Data Protection in the AWS Serverless Application Repository

The AWS shared responsibility model applies to data protection in AWS Serverless Application Repository. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. You are also responsible for the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail. For information about using CloudTrail trails to capture AWS activities, see Working with CloudTrail trails in the *AWS CloudTrail User Guide*.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-3 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-3.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with AWS Serverless Application Repository or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Encryption in Transit

AWS Serverless Application Repository API endpoints only support secure connections over HTTPS. When you manage AWS Serverless Application Repository resources with the AWS Management Console, AWS SDK, or the AWS Serverless Application Repository API, all communication is encrypted with Transport Layer Security (TLS).

For a full list of API endpoints, see AWS Regions and Endpoints in the *AWS General Reference*.

Encryption at Rest

The AWS Serverless Application Repository encrypts files that you upload to the AWS Serverless Application Repository, including deployment packages and layer archives.

Identity and Access Management for the AWS Serverless Application Repository

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use AWS Serverless Application Repository resources. IAM is an AWS service that you can use with no additional charge.

To get an overview of how IAM works, see Understanding How IAM Works in the *IAM User Guide*.

Topics

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in AWS Serverless Application Repository.

Service user – If you use the AWS Serverless Application Repository service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more AWS Serverless Application Repository features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in AWS Serverless Application Repository, see Troubleshooting AWS Serverless Application Repository Identity and Access.

Service administrator – If you're in charge of AWS Serverless Application Repository resources at your company, you probably have full access to AWS Serverless Application Repository. It's your job to determine which AWS Serverless Application Repository features and resources your service users should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with AWS Serverless Application Repository, see How the AWS Serverless Application Repository Works with IAM.

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to AWS Serverless Application Repository. To view example AWS Serverless Application Repository identity-based policies that you can use in IAM, see AWS Serverless Application Repository Identity-Based Policy Examples.

Authenticating with Identities

Authentication is how you sign in to AWS using your identity credentials. You must be *authenticated* (signed in to AWS) as the AWS account root user, as an IAM user, or by assuming an IAM role.

You can sign in to AWS as a federated identity by using credentials provided through an identity source. AWS IAM Identity Center (IAM Identity Center) users, your company's single sign-on authentication, and your Google or Facebook credentials are examples of federated identities. When you sign in as a federated identity, your administrator previously set up identity federation using IAM roles. When you access AWS by using federation, you are indirectly assuming a role.

Depending on the type of user you are, you can sign in to the AWS Management Console or the AWS access portal. For more information about signing in to AWS, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If you don't use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see AWS Signature Version 4 for API requests in the *IAM User Guide*.

Regardless of the authentication method that you use, you might be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Multi-factor authentication in the *AWS IAM Identity Center User Guide* and AWS Multi-factor authentication in IAM in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see Tasks that require root user credentials in the *IAM User Guide*.

IAM Users and Groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see Rotate access keys regularly for use cases that require long-term credentials in the *IAM User Guide*.

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term

credentials, but roles provide temporary credentials. To learn more, see Use cases for IAM users in the *IAM User Guide*.

IAM Roles

An *IAM role* is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. To temporarily assume an IAM role in the AWS Management Console, you can switch from a user to an IAM role (console). You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see Methods to assume a role in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see Create a role for a third-party identity provider (federation) in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see Permission sets in the *AWS IAM Identity Center User Guide*.
- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
  - **Forward access sessions (FAS)** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that

requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.

- **Service role** – A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Create a role to delegate permissions to an AWS service in the *IAM User Guide*.
- **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Use an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

Managing Access Using Policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when a principal (user, root user, or role session) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

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

By default, users and roles have no permissions. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-Based Policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see Choose between managed policies and inline policies in the *IAM User Guide*.

Resource-Based Policies

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

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Access Control Lists (ACLs)

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

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see Access control list (ACL) overview in the *Amazon Simple Storage Service Developer Guide*.

Other Policy Types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of an entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see Service control policies in the *AWS Organizations User Guide*.
- **Resource control policies (RCPs)** – RCPs are JSON policies that you can use to set the maximum available permissions for resources in your accounts without updating the IAM policies attached to each resource that you own. The RCP limits permissions for resources in member accounts and can impact the effective permissions for identities, including the AWS account root user, regardless of whether they belong to your organization. For more information about Organizations and RCPs, including a list of AWS services that support RCPs, see Resource control policies (RCPs) in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session policies in the *IAM User Guide*.

Multiple Policy Types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

How the AWS Serverless Application Repository Works with IAM

Before you use IAM to manage access to the AWS Serverless Application Repository, you should understand what IAM features are available to use with the AWS Serverless Application Repository.

To get an overview of how IAM works, see Understanding How IAM Works in the *IAM User Guide*. To get a high-level view of how the AWS Serverless Application Repository and other AWS services work with IAM, see AWS Services That Work with IAM in the *IAM User Guide*.

Topics

- AWS Serverless Application Repository Identity-Based Policies
- AWS Serverless Application Repository Application Policies
- Authorization Based on AWS Serverless Application Repository Tags
- AWS Serverless Application Repository IAM Roles

AWS Serverless Application Repository Identity-Based Policies

With IAM identity-based policies, you can specify allowed or denied actions and resources, as well as the conditions under which actions are allowed or denied. The AWS Serverless Application Repository supports specific actions, resources, and condition keys. To learn about all of the elements that you use in a JSON policy, see IAM JSON Policy Elements Reference in the *IAM User Guide*.

The following shows an example of a permissions policy.

```
{

    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CreateApplication",
            "Effect": "Allow",
            "Action": [
```

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

The policy has two statements:

- The first statement grants permissions for the AWS Serverless Application Repository action `serverlessrepo:CreateApplication` on all AWS Serverless Application Repository resources, as specified by the wildcard character (*) as the `Resource` value.
- The second statement grants permission for the AWS Serverless Application Repository action `serverlessrepo:CreateApplicationVersion` on an AWS resource by using the Amazon Resource Name (ARN) for an AWS Serverless Application Repository application. The application is specified by the `Resource` value.

The policy doesn't specify the `Principal` element because in an identity-based policy, you don't specify the principal who gets the permission. When you attach policy to a user, the user is the implicit principal. When you attach a permission policy to an IAM role, the principal identified in the role's trust policy gets the permissions.

For a table showing all of the AWS Serverless Application Repository API operations and the AWS resources that they apply to, see AWS Serverless Application Repository API Permissions: Actions and Resources Reference.

Actions

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

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions in a policy to grant permissions to perform the associated operation.

Policy actions in the AWS Serverless Application Repository use the following prefix before the action: `serverlessrepo:`. For example, to grant someone permission to run an AWS Serverless Application Repository instance with the AWS Serverless Application Repository `SearchApplications` API operation, you include the `serverlessrepo:SearchApplications` action in their policy. Policy statements must include either an `Action` or `NotAction` element. The AWS Serverless Application Repository defines its own set of actions that describe tasks that you can perform with this service.

To specify multiple actions in a single statement, separate them with commas as follows:

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

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word `List`, include the following action:

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

To see a list of AWS Serverless Application Repository actions, see Actions Defined by AWS Serverless Application Repository in the *IAM User Guide*.

Resources

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

The `Resource` JSON policy element specifies the object or objects to which the action applies. Statements must include either a `Resource` or a `NotResource` element. As a best practice,

specify a resource using its Amazon Resource Name (ARN). You can do this for actions that support a specific resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

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

In the AWS Serverless Application Repository, the primary AWS resource is an AWS Serverless Application Repository *application*. AWS Serverless Application Repository applications have unique Amazon Resource Names (ARNs) associated with them, as shown in the following table.

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

For more information about the format of ARNs, see Amazon Resource Names (ARNs) and AWS Service Namespaces.

The following is an example policy that grants permissions for the `serverlessrepo:ListApplications` action on all AWS resources. In the current implementation, the AWS Serverless Application Repository doesn't support identifying specific AWS resources by using the AWS resource ARNs (also referred to as resource-level permissions) for some of the API actions. In these cases, you must specify a wildcard character (*).

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

For a table showing all of the AWS Serverless Application Repository API actions and the AWS resources that they apply to, see AWS Serverless Application Repository API Permissions: Actions and Resources Reference.

Condition Keys

The AWS Serverless Application Repository doesn't provide any service-specific condition keys, but it does support using some global condition keys. To see all AWS global condition keys, see AWS Global Condition Context Keys in the *IAM User Guide*.

Examples

To view examples of AWS Serverless Application Repository identity-based policies, see AWS Serverless Application Repository Identity-Based Policy Examples.

AWS Serverless Application Repository Application Policies

Application policies determine the actions that a specified principal or principalOrg can perform on an AWS Serverless Application Repository application.

You can add permissions to the policy associated with an AWS Serverless Application Repository application. Permissions policies attached to AWS Serverless Application Repository applications are referred to as *application policies*. Application policies are extensions of IAM resource-based policies. The primary resource is the AWS Serverless Application Repository application. You can use AWS Serverless Application Repository application policies to manage application deployment permissions.

AWS Serverless Application Repository application policies are primarily used by publishers to grant permission to consumers to deploy their applications, and related operations such as to search for and view details of those applications. Publishers can set application permissions to the following three categories:

- **Private** – Applications that were created with the same account, and haven't been shared with any other account. You have permission to deploy applications that were created using your AWS account.
- **Privately shared** – Applications that the publisher has explicitly shared with a specific set of AWS accounts or AWS Organizations. You have permission to deploy applications that have been shared with your AWS account or AWS Organization.

- **Publicly shared** – Applications that the publisher has shared with everyone. You have permission to deploy any publicly shared application.

You can grant permissions by using the AWS CLI, the AWS SDKs, or the AWS Management Console.

Examples

To view examples of managing AWS Serverless Application Repository application policies, see AWS Serverless Application Repository Application Policy Examples.

Authorization Based on AWS Serverless Application Repository Tags

The AWS Serverless Application Repository doesn't support controlling access to resources or actions based on tags.

AWS Serverless Application Repository IAM Roles

An IAM role is an entity within your AWS account that has specific permissions.

Using Temporary Credentials with the AWS Serverless Application Repository

You can use temporary credentials to sign in with federation, to assume an IAM role, or to assume a cross-account role. You obtain temporary security credentials by calling AWS STS API operations such as AssumeRole or GetFederationToken.

The AWS Serverless Application Repository supports using temporary credentials.

Service-Linked Roles

The AWS Serverless Application Repository doesn't support service-linked roles.

Service Roles

The AWS Serverless Application Repository doesn't support service roles.

AWS Serverless Application Repository Identity-Based Policy Examples

By default, IAM users and roles don't have permission to create or modify AWS Serverless Application Repository resources. They also can't perform tasks using the AWS Management

Console, AWS CLI, or AWS API. An IAM administrator must create IAM policies that grant users and roles permission to perform specific API operations on the specified resources they need. The administrator must then attach those policies to the IAM users or groups that require those permissions.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Creating Policies on the JSON Tab in the *IAM User Guide*.

Topics

Policy Best Practices

Identity-based policies are very powerful. They determine whether someone can create, access, or delete AWS Serverless Application Repository resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Grant least privilege** – When you create custom policies, grant only the permissions required to perform a task. Start with a minimum set of permissions and grant additional permissions as necessary. Doing so is more secure than starting with permissions that are too lenient and then trying to tighten them later. For more information, see Grant Least Privilege in the *IAM User Guide*.
- **Enable MFA for sensitive operations** – For extra security, require IAM users to use multi-factor authentication (MFA) to access sensitive resources or API operations. For more information, see Using Multi-Factor Authentication (MFA) in AWS in the *IAM User Guide*.
- **Use policy conditions for extra security** – To the extent that it's practical, define the conditions under which your identity-based policies allow access to a resource. For example, you can write conditions to specify a range of allowable IP addresses that a request must come from. You can also write conditions to allow requests only within a specified date or time range, or to require the use of SSL or MFA. For more information, see IAM JSON Policy Elements: Condition in the *IAM User Guide*.

Using the AWS Serverless Application Repository Console

The AWS Serverless Application Repository console provides an integrated environment for you to discover and manage AWS Serverless Application Repository applications. The console provides features and workflows that often require permissions to manage an AWS Serverless Application Repository application in addition to the API-specific permissions documented in the AWS Serverless Application Repository API Permissions: Actions and Resources Reference.

For more information about permissions needed to use the AWS Serverless Application Repository console, see Customer Managed Policy Examples.

Allow Users to View Their Own Permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

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

Customer Managed Policy Examples

The examples in this section provide a group of sample policies that you can attach to a user. If you're new to creating policies, we recommend that you first create an IAM user in your account and attach the policies to the user in sequence. You can also use these examples to create a single customized policy that includes permissions to perform multiple actions, and then attach it to the user.

For more information about how to attach policies to users, see Adding Permissions to a User in the *IAM User Guide*.

Examples

- Publisher Example 1: Allow a Publisher to List Applications
- Publisher Example 2: Allow a Publisher to View Details of an Application or Application Version
- Publisher Example 3: Allow a Publisher to Create an Application or Application Version
- Publisher Example 4: Allow a Publisher to Create an Application Policy to Share Applications with Others
- Consumer Example 1: Allow a Consumer to Search for Applications
- Consumer Example 2: Allow a Consumer to View Details of an Application
- Consumer Example 3: Allow a Consumer to Deploy an Application
- Consumer Example 4: Deny Access to Deployment Assets
- Consumer Example 5: Prevent a Consumer Searching and Deploying Public Applications

Publisher Example 1: Allow a Publisher to List Applications

An IAM user in your account must have permissions for the `serverlessrepo:ListApplications` operation before the user can see anything in the console. When you grant these permissions, the console can show the list of AWS Serverless

Application Repository applications in the AWS account created in the specific AWS Region that the user belongs to.

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

Publisher Example 2: Allow a Publisher to View Details of an Application or Application Version

A user can select an AWS Serverless Application Repository application and view details of the application. Such details include author, description, versions, and other configuration information. To do this, the user needs permissions for the `serverlessrepo:GetApplication` and `serverlessrepo:ListApplicationVersions` API operations for the AWS Serverless Application Repository.

In the following example, these permissions are granted for the specific application whose Amazon Resource Name (ARN) is specified as the `Resource` value.

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

```
        }
    ]
}
```

Publisher Example 3: Allow a Publisher to Create an Application or Application Version

If you want to allow a user to have permissions to create AWS Serverless Application Repository applications, you need to grant permissions to the `serverlessrepo:CreateApplication` and `serverlessrepo:CreateApplicationVersions` operations, as shown in the following policy.

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

Publisher Example 4: Allow a Publisher to Create an Application Policy to Share Applications with Others

In order for users to share applications with others, you must grant them permissions to create application policies, as shown in the following policy.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ShareApplication",
            "Effect": "Allow",
            "Action": [
```

```
                "serverlessrepo:PutApplicationPolicy",
                "serverlessrepo:GetApplicationPolicy",
            ],
            "Resource": "*"
        }
    ]
}
```

Consumer Example 1: Allow a Consumer to Search for Applications

For consumers to search for applications, you must grant them the following permissions.

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

Consumer Example 2: Allow a Consumer to View Details of an Application

A user can select an AWS Serverless Application Repository application and view details of the application, such as author, description, versions, and other configuration information. To do so, the user must have permissions for the following AWS Serverless Application Repository operations.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewApplication",
            "Effect": "Allow",
```

```
            "Action": [
                "serverlessrepo:GetApplication",
                "serverlessrepo:ListApplicationVersions"
            ],
            "Resource": "*"
        }
    ]
}
```

Consumer Example 3: Allow a Consumer to Deploy an Application

For customers to deploy applications, you must grant them permissions to perform a number of operations. The following policy provides customers with the required permissions.

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

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

Note

Deploying an application might require permissions to use additional AWS resources. Because the AWS Serverless Application Repository uses the same underlying deployment mechanism as AWS CloudFormation, see Controlling Access with AWS Identity and Access Management for more information. For help with deployment issues related to permissions, see Troubleshooting: Insufficient IAM Permissions.

Consumer Example 4: Deny Access to Deployment Assets

When an application is privately shared with an AWS account, by default, all users in that account can access the deployment assets of all other users in the same account. The following policy prevents users in an account from accessing deployment assets, which are stored in the Amazon S3 bucket for the AWS Serverless Application Repository.

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

Consumer Example 5: Prevent a Consumer Searching and Deploying Public Applications

You can prevent users from performing certain actions on applications.

The following policy applies to public applications by specifying `serverlessrepo:applicationType` to be `public`. It prevents users from performing a number of actions by specifying `Effect` to be `Deny`. For more information about condition keys available for AWS Serverless Application Repository, see Actions, Resources, and Condition Keys for AWS Serverless Application Repository.

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

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

Note

This policy statement can also be used as a Service Control Policy and applied to an AWS organization. For more information about Service Control Policies, see Service Control Policies in the *AWS Organizations User Guide*.

AWS Serverless Application Repository Application Policy Examples

Permissions policies attached to AWS Serverless Application Repository applications are referred to as *application policies*. Application policies determine the actions that a specified principal or principalOrg can perform on an AWS Serverless Application Repository application.

An AWS Serverless Application Repository *application* is the primary AWS resource in the AWS Serverless Application Repository. AWS Serverless Application Repository application policies are primarily used by publishers to grant permission to consumers to deploy their applications, and related operations such as to search for and view details of those applications.

Publishers can set application permissions to the following three categories:

- **Private** – Applications that were created with the same account, and haven't been shared with any other account. Only consumers that share your AWS account have permission to deploy private applications.
- **Privately shared** – Applications that the publisher has explicitly shared with a specific set of AWS accounts, or with AWS accounts in an AWS organization. Consumers have permission to

deploy applications that have been shared with their AWS account or AWS organization. For more information about AWS organizations, see the *AWS Organizations User Guide*.

- **Publicly shared** – Applications that the publisher has shared with everyone. All consumers have permission to deploy any publicly shared application.

> **Note**
>
> For **privately shared applications**, the AWS Serverless Application Repository only supports *AWS accounts* as principals. Publishers can grant or deny all users within an AWS account as a single group to an AWS Serverless Application Repository application. Publishers cannot grant or deny individual users within an AWS account to an AWS Serverless Application Repository application.

For instructions on setting application permissions using the AWS Management Console, see Sharing an Application.

For instructions on setting application permissions using the AWS CLI and examples, see the following sections.

Application Permissions (AWS CLI and AWS SDKs)

When you're using the AWS CLI or the AWS SDKs to set permissions for an AWS Serverless Application Repository application, you can specify the following actions:

Action	Description
GetApplication	Grants permission to view information about the application.
CreateCloudFormationChangeSet	Grants permission for the application to be deployed. Note: This action does *not* grant any other permission other than to deploy.
CreateCloudFormationTemplate	Grants permission to create an AWS CloudFormation template for the application.
ListApplicationVersions	Grants permission to list the versions of the application.

Action	Description
ListApplicationDependencies	Grants permission to list the list applications that are nested in the containing application.
SearchApplications	Grants permission for the application to be searched for.
Deploy	This action enables all the actions listed earlier in the table. That is, it grants permission for the application to be viewed, for it to be deployed, for versions to be listed, and for it to be searched for.

Application Policy Examples

The following examples show how to grant permissions by using the AWS CLI. For information on how to grant permissions using the AWS Management Console, see Sharing an Application.

All of the examples in this section use these AWS CLI commands to manage permissions policies associated with AWS Serverless Application Repository applications:

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

Topics

Example 1: Share an Application with Another Account

To share an application with another specific account, but keep it from being shared with others, you specify the AWS account ID that you want to share with as the principal. This is also known as setting the application to *privately shared*. To do this, use the following AWS CLI command.

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

> **Note**
>
> *Privately shared* applications can only be used in the same AWS Region where the application is created.

Example 2: Share an Application Publicly

To make an application public, you share it with everyone by specifying "*" as the principal, as in the following example. Applications that are shared publicly are available in all Regions.

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

> **Note**
>
> In order to share an application publicly, it must have both the `SemanticVersion` and `LicenseUrl` properties set.

Example 3: Make an Application Private

You can make an application private, so it's not shared with anyone and can only be deployed by the AWS account that owns it. To do so, you clear out the principals and actions from the policy, which also removes permissions from other accounts within your AWS organization from deploying your application.

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

> **Note**
>
> *Private* applications can only be used in the same AWS Region where the application is created.

Example 4: Specifying Multiple Accounts and Permissions

You can grant multiple permissions, and you can grant them to more than one AWS account at a time. To do this, you specify lists as the principal and actions, as shown in the following example.

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

Example 5: Share an Application with All Accounts in an AWS Organization

Permissions can be granted to all users within an AWS organization. You do this by specifying your organization ID, as in the following example.

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

For more informaton about AWS organizations, see the *AWS Organizations User Guide*.

> **Note**
>
> You can only specify the AWS organization that your AWS account is a member of. If you try to specify an AWS organization that you are not a member of, an error will result.

To share your application with your AWS organization, you must include permission for the `UnshareApplication` action, in case the sharing needs to be revoked in the future.

Example 6: Sharing an Application with Some Accounts in an AWS Organization

Permissions can be granted to specific accounts within an AWS organization. You do this by specifying a list of AWS accounts as the principal, and your organization ID, as in the following example.

```
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

You can only specify the AWS organization that your AWS account is a member of. If you try to specify an AWS organization that you are not a member of, an error will result.
To share your application with your AWS organization, you must include permission for the `UnshareApplication` action, in case the sharing needs to be revoked in the future.

Example 7: Retrieve an Application Policy

To view an application's current policy, for example to see whether it's currently being shared, you use the `get-application-policy` command, like in the following example.

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

Example 8: Allow Application to Be Nested by Specific Accounts

Public applications are allowed to be nested by anyone. If you want to only allow your application to be nested by specific accounts, you must set the following minimal permissions, as shown in the following example.

```
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 Permissions: Actions and Resources Reference

When you set up access control and write permissions policies that you can attach to an IAM identity (identity-based policies), you can use the following table as a reference. The each AWS Serverless Application Repository API operation, the corresponding actions that you can grant permissions to perform the action, and the AWS resource that you can grant the permissions. You specify the actions in the policy's `Action` field, and you specify the resource value in the policy's `Resource` field.

To specify an action, use the `serverlessrepo:` prefix followed by the API operation name (for example, `serverlessrepo:ListApplications`).

Operation	URI	Method	AWS Resources (ARNs)
Operation: ListApplications **Required Permissions:** serverlessrepo:ListApplications	/applications	GET	*
Operation: CreateApplication **Required Permissions:** serverlessrepo:CreateApplication	/applications	POST	*
Operation: GetApplication	/applications/*application-id*	GET	arn:aws:serverlessrepo:*region*:*account-id* :applicat

Operation	URI	Method	AWS Resources (ARNs)
Required Permissions: serverlessrepo:GetApplication			ions/*application-name*
Operation: DeleteApplication **Required Permissions:** serverlessrepo:DeleteApplication	/applications/*application-id*	DELETE	arn:aws:serverlessrepo:*region*:*account-id* :applications/*application-name*
Operation: UpdateApplication **Required Permissions:** serverlessrepo:UpdateApplication	/applications/*application-id*	PATCH	arn:aws:serverlessrepo:*region*:*account-id* :applications/*application-name*
Operation: CreateCloudFormationChangeSet **Required Permissions:** serverlessrepo:CreateCloudFormationChangeSet	/applications/*application-id* /changesets	POST	arn:aws:serverlessrepo:*region*:*account-id* :applications/*application-name*
Operation: GetApplicationPolicy **Required Permissions:** serverlessrepo:GetApplicationPolicy	/applications/*application-id* /policy	GET	arn:aws:serverlessrepo:*region*:*account-id* :applications/*application-name*

Operation	URI	Method	AWS Resources (ARNs)
Operation: PutApplicationPolicy **Required Permissions:** serverlessrepo:PutApplicationPolicy	/applications/*application-id* /policy	PUT	arn:aws:serverlessrepo:*region*:*account-id* :applications/*application-name*
Operation: ListApplicationVersions **Required Permissions:** serverlessrepo:ListApplicationVersions	/applications/*application-id* /versions	GET	arn:aws:serverlessrepo:*region*:*account-id* :applications/*application-name*
Operation: CreateApplicationVersion **Required Permissions:** serverlessrepo:CreateApplicationVersion	/applications/*application-id* /versions/*semantic-version*	PUT	arn:aws:serverlessrepo:*region*:*account-id* :applications/*application-name*
Operation: ListApplicationDependencies **Required Permissions:** serverlessrepo:ListApplicationDependencies	/applications/*application-id* /dependencies	GET	arn:aws:serverlessrepo:*region*:*account-id* :applications/*application-name*

Operation	URI	Method	AWS Resources (ARNs)
Operation: SearchApplications **Required Permissions:** serverlessrepo:SearchApplications	n/a	n/a	*

Troubleshooting AWS Serverless Application Repository Identity and Access

Use the following information to help you diagnose and fix common issues that you might encounter when working with the AWS Serverless Application Repository and IAM.

Topics

- I'm Not Authorized to Perform an Action in the AWS Serverless Application Repository
- I'm Not Authorized to Perform iam:PassRole
- I'm an Administrator and Want to Allow Others to Access the AWS Serverless Application Repository
- I Want to Allow People Outside of My AWS Account to Access My AWS Serverless Application Repository Resources

I'm Not Authorized to Perform an Action in the AWS Serverless Application Repository

If the AWS Management Console tells you that you're not authorized to perform an action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about an application but doesn't have `serverlessrepo:GetApplication` permissions.

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

In this case, Mateo asks his administrator to update his policies to allow him to access the *my-example-application* resource by using the `serverlessrepo:GetApplication` operation.

I'm Not Authorized to Perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to AWS Serverless Application Repository.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in AWS Serverless Application Repository. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

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

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I'm an Administrator and Want to Allow Others to Access the AWS Serverless Application Repository

To allow others to access AWS Serverless Application Repository, you must grant permission to the people or applications that need access. If you are using AWS IAM Identity Center to manage people and applications, you assign permission sets to users or groups to define their level of access. Permission sets automatically create and assign IAM policies to IAM roles that are associated with the person or application. For more information, see Permission sets in the *AWS IAM Identity Center User Guide*.

If you are not using IAM Identity Center, you must create IAM entities (users or roles) for the people or applications that need access. You must then attach a policy to the entity that grants them the

correct permissions in AWS Serverless Application Repository. After the permissions are granted, provide the credentials to the user or application developer. They will use those credentials to access AWS. To learn more about creating IAM users, groups, policies, and permissions, see IAM Identities and Policies and permissions in IAM in the *IAM User Guide*.

I Want to Allow People Outside of My AWS Account to Access My AWS Serverless Application Repository Resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether AWS Serverless Application Repository supports these features, see How the AWS Serverless Application Repository Works with IAM.
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.

Logging and Monitoring in the AWS Serverless Application Repository

Monitoring is an important part of maintaining the reliability, availability, and performance of your AWS solutions. You should collect monitoring data from all of the parts of your AWS solution so that you can more easily debug a multipoint failure if one occurs. AWS provides several tools for monitoring your AWS Serverless Application Repository resources and responding to potential incidents, such as the following:

AWS CloudTrail Logs

The AWS Serverless Application Repository is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in the AWS Serverless Application Repository. CloudTrail captures all API calls for the AWS Serverless Application Repository as events.

Topics

Logging AWS Serverless Application Repository API Calls with AWS CloudTrail

AWS Serverless Application Repository is integrated with AWS CloudTrail, which is a service that provides a record of actions taken by a user, role, or an AWS service in the AWS Serverless Application Repository. CloudTrail captures all API calls for the AWS Serverless Application Repository as events. The calls captured include calls from the AWS Serverless Application Repository console and code calls to the AWS Serverless Application Repository API operations.

If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for the AWS Serverless Application Repository. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**.

Using the information collected by CloudTrail, you can determine the request that was made to the AWS Serverless Application Repository. You can also determine the IP address from which the request was made, who made the request, when the request was made, and additional details.

To learn more about CloudTrail, see the *AWS CloudTrail User Guide*.

AWS Serverless Application Repository Information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in the AWS Serverless Application Repository, that activity is recorded in a CloudTrail event, along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Viewing Events with CloudTrail Event History.

For an ongoing record of events in your AWS account, including events for the AWS Serverless Application Repository, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The

trail logs events from all AWS Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- Overview for Creating a Trail
- CloudTrail Supported Services and Integrations
- Configuring Amazon SNS Notifications for CloudTrail
- Receiving CloudTrail Log Files from Multiple Regions and Receiving CloudTrail Log Files from Multiple Accounts

All AWS Serverless Application Repository actions are logged by CloudTrail and are documented on the AWS Serverless Application Repository Resources page. For example, calls to the `CreateApplication`, `UpdateApplications`, and `ListApplications` operations generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity Element.

Understanding AWS Serverless Application Repository Log File Entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the `CreateApplication` action.

```
{
    "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"
}
```

Compliance Validation for the AWS Serverless Application Repository

Third-party auditors assess the security and compliance of the AWS Serverless Application Repository as part of multiple AWS compliance programs. These include SOC, PCI, FedRAMP, and others.

For a list of AWS services that are in scope of specific compliance programs, see AWS Services in Scope by Compliance Program. For general information, see AWS Compliance Programs.

You can download third-party audit reports by using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

Your compliance responsibility when using the AWS Serverless Application Repository is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- Security and Compliance Quick Start Guides – These deployment guides discuss architectural considerations and provide steps for deploying security-focused and compliance-focused baseline environments on AWS.
- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- AWS Config – This AWS service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

Resilience in the AWS Serverless Application Repository

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between Availability Zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

Infrastructure Security in the AWS Serverless Application Repository

As a managed service, AWS Serverless Application Repository is protected by AWS global network security. For information about AWS security services and how AWS protects infrastructure, see AWS Cloud Security. To design your AWS environment using the best practices for infrastructure security, see Infrastructure Protection in *Security Pillar AWS Well-Architected Framework*.

You use AWS published API calls to access AWS Serverless Application Repository through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

Access AWS Serverless Application Repository using an interface endpoint (AWS PrivateLink)

You can use AWS PrivateLink to create a private connection between your VPC and AWS Serverless Application Repository. You can access AWS Serverless Application Repository as if it were in your VPC, without the use of an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Instances in your VPC don't need public IP addresses to access AWS Serverless Application Repository.

You establish this private connection by creating an *interface endpoint*, powered by AWS PrivateLink. We create an endpoint network interface in each subnet that you enable for the interface endpoint. These are requester-managed network interfaces that serve as the entry point for traffic destined for AWS Serverless Application Repository.

For more information, see Access AWS services through AWS PrivateLink in the *AWS PrivateLink Guide*.

Considerations for AWS Serverless Application Repository

Before you set up an interface endpoint for AWS Serverless Application Repository, review Considerations in the *AWS PrivateLink Guide*.

AWS Serverless Application Repository supports making calls to all of its API actions through the interface endpoint.

Create an interface endpoint for AWS Serverless Application Repository

You can create an interface endpoint for AWS Serverless Application Repository using either the Amazon VPC console or the AWS Command Line Interface (AWS CLI). For more information, see Create an interface endpoint in the *AWS PrivateLink Guide*.

Create an interface endpoint for AWS Serverless Application Repository using the following service name:

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

If you enable private DNS for the interface endpoint, you can make API requests to AWS Serverless Application Repository using its default Regional DNS name. For example, `serverlessrepo.us-east-1.amazonaws.com`.

Create an endpoint policy for your interface endpoint

An endpoint policy is an IAM resource that you can attach to an interface endpoint. The default endpoint policy allows full access to AWS Serverless Application Repository through the interface endpoint. To control the access allowed to AWS Serverless Application Repository from your VPC, attach a custom endpoint policy to the interface endpoint.

An endpoint policy specifies the following information:

- The principals that can perform actions (AWS accounts, IAM users, and IAM roles).
- The actions that can be performed.
- The resources on which the actions can be performed.

For more information, see Control access to services using endpoint policies in the *AWS PrivateLink Guide*.

Example: VPC endpoint policy for AWS Serverless Application Repository actions

The following is an example of a custom endpoint policy. When you attach this policy to your interface endpoint, it grants access to the listed AWS Serverless Application Repository action for all principals on all resources. The following example allows all users the permission to create applications through the VPC endpoint.

```
{
```

```
   "Statement": [
      {
         "Principal": "*",
         "Effect": "Allow",
         "Action": [
            "serverlessrepo:CreateApplication"
         ],
         "Resource":"*"
      }
   ]
}
```

AWS Serverless Application Repository Quotas

The AWS Serverless Application Repository has a quota for the number of public applications that an AWS account can have in each AWS Region. This quota applies per Region and can be increased. To request an increase, use the Support Center console.

Resource	Default Quota
Public applications (per AWS account per AWS Region)	100

The following quotas apply to storage that is available for code packages and application policies. You can't change these quotas.

Resource	Quota
Free Amazon S3 storage for code packages (per AWS account per AWS Region)	5 GB
Application policy length	6,144 characters

Troubleshooting the AWS Serverless Application Repository

When you use the AWS Serverless Application Repository, you might encounter issues when you create, update, or delete your applications. Use this section to help troubleshoot common issues that you might encounter. You can also search for answers and post questions in the AWS Serverless Application Repository forums.

> **Note**
>
> Applications in the AWS Serverless Application Repository are deployed by using AWS CloudFormation. For information on troubleshooting AWS CloudFormation issues, see the *AWS CloudFormation Troubleshooting Guide*.

Topics

You Can't Make an Application Public

If you can't make your application public, you might be missing a license file for your application that is approved by the Open Source Initiative (OSI).

To make your application public, you need an OSI-approved license file, and also a successfully published version of the application with a source code URL for the version. You can't update the license of an application after the application is created.

If you can't make your application public because you are missing a license file, delete the application and create a new one with the same name. Make sure that you provide it with one or more open-source licenses approved by the Open Source Initiative (OSI) organization.

A Quota Was Exceeded

If you receive an error message that indicates that a quota was exceeded, check to see if you reached a resource quota. For AWS Serverless Application Repository quotas, see AWS Serverless Application Repository Quotas.

An Updated Readme File Doesn't Appear Immediately

When you make your application public, the contents of your application can take up to 24 hours to update. If you experience delays longer than 24 hours, try contacting AWS Support for help. For details, see following.

You Can't Deploy an Application Due to Insufficient IAM Permissions

To deploy an AWS Serverless Application Repository application, you need permissions to AWS Serverless Application Repository resources and AWS CloudFormation stacks. You might also need permission to use the underlying services described in the application. For example, if you're creating an Amazon S3 bucket or an Amazon DynamoDB table, you need permissions to Amazon S3 or DynamoDB.

If you run into this type of issue, review your AWS Identity and Access Management (IAM) policy and verify that you have the necessary permissions. For more information, see Controlling Access with AWS Identity and Access Management.

You Can't Deploy the Same Application Twice

The application name that you provide is used as the name of the AWS CloudFormation stack. If you have problems deploying an application, make sure that you don't have an existing AWS CloudFormation stack with the same name. If you do, provide a different application name or delete the existing stack to deploy the application with the same name.

Why Is My Application Not Publicly Available

Applications are private by default. In order to make your application public, follow the steps here.

Contacting Support

In some cases, you might not be able to find troubleshooting solutions in this section or through the AWS Serverless Application Repository forums. If you have AWS Premium Support, you can create a technical support case at AWS Support.

Before you contact AWS Support, make sure to get the Amazon Resource Name (ARN) for the application that you have questions about. You can find the application ARN in the AWS Serverless Application Repository console.

Operations

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

- CreateApplication

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

  Creates an application version.
- CreateCloudFormationChangeSet

  Creates an AWS CloudFormation change set for the given application.
- CreateCloudFormationTemplate

  Creates an AWS CloudFormation template.
- DeleteApplication

  Deletes the specified application.
- GetApplication

  Gets the specified application.
- GetApplicationPolicy

  Retrieves the policy for the application.
- GetCloudFormationTemplate

  Gets the specified AWS CloudFormation template.
- ListApplicationDependencies

  Retrieves the list of applications nested in the containing application.
- ListApplications

  Lists applications owned by the requester.
- ListApplicationVersions

  Lists versions for the specified application.

- PutApplicationPolicy

  Sets the permission policy for an application. For the list of actions supported for this operation, see Application Permissions .
- UnshareApplication

  Unshares an application from an AWS Organization.

  This operation can be called only from the organization's management account.
- UpdateApplication

  Updates the specified application.

Resources

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

Topics

Applications

URI

`/applications`

HTTP methods

GET

Operation ID: `ListApplications`

Lists applications owned by the requester.

Query parameters

Name	Type	Required	Description
`maxItems`	String	False	The total number of items to return.

Name	Type	Required	Description
nextToken	String	False	A token to specify where to start paginating.

Responses

Status code	Response model	Description
200	ApplicationPage	Success
400	BadRequestException	One of the parameters in the request is invalid.
403	ForbiddenException	The client is not authentic ated.
404	NotFoundException	The resource (for example, an access policy statement) specified in the request doesn't exist.
500	InternalServerErro rException	The AWS Serverless Applicati on Repository service encountered an internal error.

POST

Operation ID: CreateApplication

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

Responses

Status code	Response model	Description
201	Application	Success

Status code	Response model	Description
400	BadRequestException	One of the parameters in the request is invalid.
403	ForbiddenException	The client is not authentic ated.
409	ConflictException	The resource already exists.
429	TooManyRequestsExc eption	The client is sending more than the allowed number of requests per unit of time.
500	InternalServerErro rException	The AWS Serverless Applicati on Repository service encountered an internal error.

OPTIONS

Responses

Status code	Response model	Description
200	None	200 response

Schemas

Request bodies

POST schema

```
{
  "name": "string",
  "description": "string",
  "author": "string",
  "spdxLicenseId": "string",
  "licenseBody": "string",
  "licenseUrl": "string",
```

```
  "readmeBody": "string",
  "readmeUrl": "string",
  "labels": [
    "string"
  ],
  "homePageUrl": "string",
  "semanticVersion": "string",
  "templateBody": "string",
  "templateUrl": "string",
  "sourceCodeUrl": "string",
  "sourceCodeArchiveUrl": "string"
}
```

Response bodies

ApplicationPage schema

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

Application schema

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

```
  "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 schema

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

ForbiddenException schema

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

NotFoundException schema

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

ConflictException schema

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

TooManyRequestsException schema

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

InternalServerErrorException schema

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

Properties

Application

Details about the application.

applicationId

The application Amazon Resource Name (ARN).

Type: string
Required: True

name

The name of the application.

Minimum length=1. Maximum length=140

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

Type: string
Required: True

description

The description of the application.

Minimum length=1. Maximum length=256

Type: string
Required: True

author

The name of the author publishing the app.

Minimum length=1. Maximum length=127.

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

Type: string
Required: True

isVerifiedAuthor

Specifies whether the author of this application has been verified. This means that AWS has made a good faith review, as a reasonable and prudent service provider, of the information provided by the requester and has confirmed that the requester's identity is as claimed.

Type: boolean
Required: False

verifiedAuthorUrl

The URL to the public profile of a verified author. This URL is submitted by the author.

Type: string
Required: False

spdxLicenseId

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

Type: string
Required: False

licenseUrl

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

Maximum size 5 MB

Type: string
Required: False

readmeUrl

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

Maximum size 5 MB

Type: string
Required: False

labels

Labels to improve discovery of apps in search results.

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

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

Type: Array of type string
Required: False

creationTime

The date and time this resource was created.

Type: string
Required: False

homePageUrl

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

Type: string
Required: False

version

Version information about the application.

Type: Version
Required: False

ApplicationPage

A list of application details.

applications

An array of application summaries.

Type: Array of type ApplicationSummary
Required: True

nextToken

The token to request the next page of results.

Type: string
Required: False

ApplicationSummary

Summary of details about the application.

applicationId

The application Amazon Resource Name (ARN).

Type: string
Required: True

name

The name of the application.

Minimum length=1. Maximum length=140

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

Type: string
Required: True

description

The description of the application.

Minimum length=1. Maximum length=256

Type: string
Required: True

author

The name of the author publishing the app.

Minimum length=1. Maximum length=127.

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

Type: string
Required: True

spdxLicenseId

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

Type: string
Required: False

labels

Labels to improve discovery of apps in search results.

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

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

Type: Array of type string
Required: False

creationTime

The date and time this resource was created.

Type: string
Required: False

homePageUrl

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

Type: string
Required: False

BadRequestException

One of the parameters in the request is invalid.

message

One of the parameters in the request is invalid.

Type: string
Required: False

errorCode

400

Type: string
Required: False

Capability

Values that must be specified in order to deploy some applications.

CAPABILITY_IAM

CAPABILITY_NAMED_IAM

CAPABILITY_AUTO_EXPAND

CAPABILITY_RESOURCE_POLICY

ConflictException

The resource already exists.

message

The resource already exists.

Type: string

Required: False

errorCode

409

Type: string

Required: False

CreateApplicationInput

Create an application request.

name

The name of the application that you want to publish.

Minimum length=1. Maximum length=140

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

Type: string

Required: True

description

The description of the application.

Minimum length=1. Maximum length=256

Type: string
Required: True

author

The name of the author publishing the app.

Minimum length=1. Maximum length=127.

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

Type: string
Required: True

spdxLicenseId

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

Type: string
Required: False

licenseBody

A local text file that contains the license of the app that matches the spdxLicenseID value of your application. The file has the format `file://<path>/<filename>`.

Maximum size 5 MB

You can specify only one of `licenseBody` and `licenseUrl`; otherwise, an error results.

Type: string
Required: False

licenseUrl

A link to the S3 object that contains the license of the app that matches the spdxLicenseID value of your application.

Maximum size 5 MB

You can specify only one of `licenseBody` and `licenseUrl`; otherwise, an error results.

Type: string
Required: False

readmeBody

A local text readme file in Markdown language that contains a more detailed description of the application and how it works. The file has the format `file://<path>/<filename>`.

Maximum size 5 MB

You can specify only one of `readmeBody` and `readmeUrl`; otherwise, an error results.

Type: string
Required: False

readmeUrl

A link to the S3 object in Markdown language that contains a more detailed description of the application and how it works.

Maximum size 5 MB

You can specify only one of `readmeBody` and `readmeUrl`; otherwise, an error results.

Type: string
Required: False

labels

Labels to improve discovery of apps in search results.

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

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

Type: Array of type string
Required: False

homePageUrl

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

Type: string
Required: False

semanticVersion

The semantic version of the application:

https://semver.org/

Type: string
Required: False

templateBody

The local raw packaged AWS SAM template file of your application. The file has the format `file://<path>/<filename>`.

You can specify only one of `templateBody` and `templateUrl`; otherwise an error results.

Type: string
Required: False

templateUrl

A link to the S3 object containing the packaged AWS SAM template of your application.

You can specify only one of `templateBody` and `templateUrl`; otherwise an error results.

Type: string
Required: False

sourceCodeUrl

A link to a public repository for the source code of your application, for example the URL of a specific GitHub commit.

Type: string
Required: False

sourceCodeArchiveUrl

A link to the S3 object that contains the ZIP archive of the source code for this version of your application.

Maximum size 50 MB

Type: string
Required: False

ForbiddenException

The client is not authenticated.

message

The client is not authenticated.

Type: string
Required: False

errorCode

403

Type: string
Required: False

InternalServerErrorException

The AWS Serverless Application Repository service encountered an internal error.

message

The AWS Serverless Application Repository service encountered an internal error.

Type: string
Required: False

errorCode

500

Type: string
Required: False

NotFoundException

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

message

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

Type: string
Required: False

errorCode

404

Type: string
Required: False

ParameterDefinition

Parameters supported by the application.

name

The name of the parameter.

Type: string
Required: True

defaultValue

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

Type: string
Required: False

description

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

Type: string
Required: False

type

The type of the parameter.

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

`String`: A literal string.

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

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

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

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

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

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

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

Type: string
Required: False

noEcho

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

Type: boolean
Required: False

allowedPattern

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

Type: string
Required: False

constraintDescription

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

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

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

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

Type: string
Required: False

minValue

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

Type: integer
Required: False

maxValue

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

Type: integer
Required: False

minLength

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

Type: integer
Required: False

maxLength

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

Type: integer
Required: False

allowedValues

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

Type: Array of type string
Required: False

referencedByResources

A list of AWS SAM resources that use this parameter.

Type: Array of type string
Required: True

TooManyRequestsException

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

message

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

Type: string
Required: False

errorCode

429

Type: string
Required: False

Version

Application version details.

applicationId

The application Amazon Resource Name (ARN).

Type: string
Required: True

semanticVersion

The semantic version of the application:

https://semver.org/

Type: string
Required: True

sourceCodeUrl

A link to a public repository for the source code of your application, for example the URL of a specific GitHub commit.

Type: string
Required: False

sourceCodeArchiveUrl

A link to the S3 object that contains the ZIP archive of the source code for this version of your application.

Maximum size 50 MB

Type: string
Required: False

templateUrl

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

Type: string
Required: True

creationTime

The date and time this resource was created.

Type: string

Required: True

parameterDefinitions

An array of parameter types supported by the application.

Type: Array of type ParameterDefinition
Required: True

requiredCapabilities

A list of values that you must specify before you can deploy certain applications. Some applications might include resources that can affect permissions in your AWS account, for example, by creating new AWS Identity and Access Management (IAM) users. For those applications, you must explicitly acknowledge their capabilities by specifying this parameter.

The only valid values are `CAPABILITY_IAM`, `CAPABILITY_NAMED_IAM`, `CAPABILITY_RESOURCE_POLICY`, and `CAPABILITY_AUTO_EXPAND`.

The following resources require you to specify `CAPABILITY_IAM` or `CAPABILITY_NAMED_IAM`: AWS::IAM::Group, AWS::IAM::InstanceProfile, AWS::IAM::Policy, and AWS::IAM::Role. If the application contains IAM resources, you can specify either `CAPABILITY_IAM` or `CAPABILITY_NAMED_IAM`. If the application contains IAM resources with custom names, you must specify `CAPABILITY_NAMED_IAM`.

The following resources require you to specify `CAPABILITY_RESOURCE_POLICY`: AWS::Lambda::Permission, AWS::IAM:Policy, AWS::ApplicationAutoScaling::ScalingPolicy, AWS::S3::BucketPolicy, AWS::SQS::QueuePolicy, and AWS::SNS::TopicPolicy.

Applications that contain one or more nested applications require you to specify `CAPABILITY_AUTO_EXPAND`.

If your application template contains any of the above resources, we recommend that you review all permissions associated with the application before deploying. If you don't specify this parameter for an application that requires capabilities, the call will fail.

Type: Array of type Capability
Required: True

resourcesSupported

Whether all of the AWS resources contained in this application are supported in the region in which it is being retrieved.

Type: boolean
Required: True

See also

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

ListApplications

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CreateApplication

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3

- AWS SDK for Python
- AWS SDK for Ruby V3

Applications applicationId

URI

`/applications/applicationId`

HTTP methods

GET

Operation ID: `GetApplication`

Gets the specified application.

Path parameters

Name	Type	Required	Description
applicationId	String	True	The Amazon Resource Name (ARN) of the application.

Query parameters

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

Responses

Status code	Response model	Description
200	Application	Success

Status code	Response model	Description
400	BadRequestException	One of the parameters in the request is invalid.
403	ForbiddenException	The client is not authentic ated.
404	NotFoundException	The resource (for example, an access policy statement) specified in the request doesn't exist.
429	TooManyRequestsExc eption	The client is sending more than the allowed number of requests per unit of time.
500	InternalServerErro rException	The AWS Serverless Applicati on Repository service encountered an internal error.

DELETE

Operation ID: `DeleteApplication`

Deletes the specified application.

Path parameters

Name	Type	Required	Description
applicationId	String	True	The Amazon Resource Name (ARN) of the application.

Responses

Status code	Response model	Description
204	None	Success
400	BadRequestException	One of the parameters in the request is invalid.
403	ForbiddenException	The client is not authentic ated.
404	NotFoundException	The resource (for example, an access policy statement) specified in the request doesn't exist.
409	ConflictException	The resource already exists.
429	TooManyRequestsExc eption	The client is sending more than the allowed number of requests per unit of time.
500	InternalServerErro rException	The AWS Serverless Applicati on Repository service encountered an internal error.

OPTIONS

Path parameters

Name	Type	Required	Description
applicationId	String	True	The Amazon Resource Name (ARN) of the application.

Responses

Status code	Response model	Description
200	None	200 response

PATCH

Operation ID: `UpdateApplication`

Updates the specified application.

Path parameters

Name	Type	Required	Description
applicationId	String	True	The Amazon Resource Name (ARN) of the application.

Responses

Status code	Response model	Description
200	Application	Success
400	BadRequestException	One of the parameters in the request is invalid.
403	ForbiddenException	The client is not authentic ated.
404	NotFoundException	The resource (for example, an access policy statement) specified in the request doesn't exist.
409	ConflictException	The resource already exists.

Status code	Response model	Description
429	TooManyRequestsException	The client is sending more than the allowed number of requests per unit of time.
500	InternalServerErrorException	The AWS Serverless Application Repository service encountered an internal error.

Schemas

Request bodies

PATCH schema

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

Response bodies

Application schema

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

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

ForbiddenException schema

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

NotFoundException schema

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

ConflictException schema

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

TooManyRequestsException schema

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

InternalServerErrorException schema

```
{
```

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

Properties

Application

Details about the application.

applicationId

The application Amazon Resource Name (ARN).

Type: string
Required: True

name

The name of the application.

Minimum length=1. Maximum length=140

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

Type: string
Required: True

description

The description of the application.

Minimum length=1. Maximum length=256

Type: string
Required: True

author

The name of the author publishing the app.

Minimum length=1. Maximum length=127.

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

Type: string
Required: True

isVerifiedAuthor

Specifies whether the author of this application has been verified. This means that AWS has made a good faith review, as a reasonable and prudent service provider, of the information provided by the requester and has confirmed that the requester's identity is as claimed.

Type: boolean
Required: False

verifiedAuthorUrl

The URL to the public profile of a verified author. This URL is submitted by the author.

Type: string
Required: False

spdxLicenseId

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

Type: string
Required: False

licenseUrl

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

Maximum size 5 MB

Type: string
Required: False

readmeUrl

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

Maximum size 5 MB

Type: string
Required: False

labels

Labels to improve discovery of apps in search results.

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

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

Type: Array of type string
Required: False

creationTime

The date and time this resource was created.

Type: string
Required: False

homePageUrl

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

Type: string
Required: False

version

Version information about the application.

Type: Version
Required: False

BadRequestException

One of the parameters in the request is invalid.

message

One of the parameters in the request is invalid.

Type: string
Required: False

errorCode

400

Type: string
Required: False

Capability

Values that must be specified in order to deploy some applications.

```
CAPABILITY_IAM
CAPABILITY_NAMED_IAM
CAPABILITY_AUTO_EXPAND
CAPABILITY_RESOURCE_POLICY
```

ConflictException

The resource already exists.

message

The resource already exists.

Type: string
Required: False

errorCode

409

Type: string
Required: False

ForbiddenException

The client is not authenticated.

message

The client is not authenticated.

Type: string
Required: False

errorCode

403

Type: string
Required: False

InternalServerErrorException

The AWS Serverless Application Repository service encountered an internal error.

message

The AWS Serverless Application Repository service encountered an internal error.

Type: string
Required: False

errorCode

500

Type: string
Required: False

NotFoundException

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

message

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

Type: string
Required: False

errorCode

404

Type: string
Required: False

ParameterDefinition

Parameters supported by the application.

name

The name of the parameter.

Type: string
Required: True

defaultValue

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

Type: string

Required: False

description

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

Type: string
Required: False

type

The type of the parameter.

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

`String`: A literal string.

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

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

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

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

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

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

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

Type: string

Required: False

noEcho

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

Type: boolean
Required: False

allowedPattern

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

Type: string
Required: False

constraintDescription

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

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

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

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

Type: string
Required: False

minValue

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

Type: integer

Required: False

maxValue

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

Type: integer
Required: False

minLength

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

Type: integer
Required: False

maxLength

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

Type: integer
Required: False

allowedValues

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

Type: Array of type string
Required: False

referencedByResources

A list of AWS SAM resources that use this parameter.

Type: Array of type string
Required: True

TooManyRequestsException

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

message

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

Type: string
Required: False

errorCode

429

Type: string
Required: False

UpdateApplicationInput

Update the application request.

description

The description of the application.

Minimum length=1. Maximum length=256

Type: string
Required: False

author

The name of the author publishing the app.

Minimum length=1. Maximum length=127.

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

Type: string

Required: False

readmeBody

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

Maximum size 5 MB

Type: string
Required: False

readmeUrl

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

Maximum size 5 MB

Type: string
Required: False

labels

Labels to improve discovery of apps in search results.

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

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

Type: Array of type string
Required: False

homePageUrl

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

Type: string
Required: False

Version

Application version details.

applicationId

The application Amazon Resource Name (ARN).

Type: string
Required: True

semanticVersion

The semantic version of the application:

https://semver.org/

Type: string
Required: True

sourceCodeUrl

A link to a public repository for the source code of your application, for example the URL of a specific GitHub commit.

Type: string
Required: False

sourceCodeArchiveUrl

A link to the S3 object that contains the ZIP archive of the source code for this version of your application.

Maximum size 50 MB

Type: string
Required: False

templateUrl

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

Type: string
Required: True

creationTime

The date and time this resource was created.

Type: string
Required: True

parameterDefinitions

An array of parameter types supported by the application.

Type: Array of type ParameterDefinition
Required: True

requiredCapabilities

A list of values that you must specify before you can deploy certain applications. Some applications might include resources that can affect permissions in your AWS account, for example, by creating new AWS Identity and Access Management (IAM) users. For those applications, you must explicitly acknowledge their capabilities by specifying this parameter.

The only valid values are `CAPABILITY_IAM`, `CAPABILITY_NAMED_IAM`, `CAPABILITY_RESOURCE_POLICY`, and `CAPABILITY_AUTO_EXPAND`.

The following resources require you to specify `CAPABILITY_IAM` or `CAPABILITY_NAMED_IAM`: AWS::IAM::Group, AWS::IAM::InstanceProfile, AWS::IAM::Policy, and AWS::IAM::Role. If the application contains IAM resources, you can specify either `CAPABILITY_IAM` or `CAPABILITY_NAMED_IAM`. If the application contains IAM resources with custom names, you must specify `CAPABILITY_NAMED_IAM`.

The following resources require you to specify `CAPABILITY_RESOURCE_POLICY`: AWS::Lambda::Permission, AWS::IAM:Policy, AWS::ApplicationAutoScaling::ScalingPolicy, AWS::S3::BucketPolicy, AWS::SQS::QueuePolicy, and AWS::SNS::TopicPolicy.

Applications that contain one or more nested applications require you to specify `CAPABILITY_AUTO_EXPAND`.

If your application template contains any of the above resources, we recommend that you review all permissions associated with the application before deploying. If you don't specify this parameter for an application that requires capabilities, the call will fail.

Type: Array of type Capability
Required: True

resourcesSupported

Whether all of the AWS resources contained in this application are supported in the region in which it is being retrieved.

Type: boolean
Required: True

See also

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

GetApplication

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DeleteApplication

- AWS Command Line Interface

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

UpdateApplication

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

Applications applicationId Changesets

URI

`/applications/applicationId/changesets`

HTTP methods

POST

Operation ID: `CreateCloudFormationChangeSet`

Creates an AWS CloudFormation change set for the given application.

Path parameters

Name	Type	Required	Description
applicationId	String	True	The Amazon Resource Name (ARN) of the application.

Responses

Status code	Response model	Description
201	ChangeSetDetails	Success
400	BadRequestException	One of the parameters in the request is invalid.
403	ForbiddenException	The client is not authentic ated.
429	TooManyRequestsExc eption	The client is sending more than the allowed number of requests per unit of time.
500	InternalServerErro rException	The AWS Serverless Applicati on Repository service encountered an internal error.

OPTIONS

Path parameters

Name	Type	Required	Description
applicationId	String	True	The Amazon Resource Name (ARN) of the application.

Responses

Status code	Response model	Description
200	None	200 response

Schemas

Request bodies

POST schema

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

Response bodies

ChangeSetDetails schema

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

BadRequestException schema

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

ForbiddenException schema

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

TooManyRequestsException schema

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

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

InternalServerErrorException schema

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

Properties

BadRequestException

One of the parameters in the request is invalid.

message

One of the parameters in the request is invalid.

Type: string
Required: False

errorCode

400

Type: string
Required: False

ChangeSetDetails

Details of the change set.

applicationId

The application Amazon Resource Name (ARN).

Type: string

Required: True

semanticVersion

The semantic version of the application:

https://semver.org/

Type: string
Required: True

changeSetId

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

Length constraints: Minimum length of 1.

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

Type: string
Required: True

stackId

The unique ID of the stack.

Type: string
Required: True

CreateCloudFormationChangeSetInput

Create an application change set request.

stackName

This property corresponds to the parameter of the same name for the *AWS CloudFormation* *CreateChangeSet* API.

Type: string

Required: True

semanticVersion

The semantic version of the application:

https://semver.org/

Type: string
Required: False

templateId

The UUID returned by CreateCloudFormationTemplate.

Pattern: [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}

Type: string
Required: False

parameterOverrides

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

Type: Array of type ParameterValue
Required: False

capabilities

A list of values that you must specify before you can deploy certain applications. Some applications might include resources that can affect permissions in your AWS account, for example, by creating new AWS Identity and Access Management (IAM) users. For those applications, you must explicitly acknowledge their capabilities by specifying this parameter.

The only valid values are `CAPABILITY_IAM`, `CAPABILITY_NAMED_IAM`, `CAPABILITY_RESOURCE_POLICY`, and `CAPABILITY_AUTO_EXPAND`.

The following resources require you to specify `CAPABILITY_IAM` or `CAPABILITY_NAMED_IAM`: AWS::IAM::Group, AWS::IAM::InstanceProfile, AWS::IAM::Policy, and AWS::IAM::Role. If

the application contains IAM resources, you can specify either `CAPABILITY_IAM` or `CAPABILITY_NAMED_IAM`. If the application contains IAM resources with custom names, you must specify `CAPABILITY_NAMED_IAM`.

The following resources require you to specify `CAPABILITY_RESOURCE_POLICY`: AWS::Lambda::Permission, AWS::IAM:Policy, AWS::ApplicationAutoScaling::ScalingPolicy, AWS::S3::BucketPolicy, AWS::SQS::QueuePolicy, and AWS::SNS:TopicPolicy.

Applications that contain one or more nested applications require you to specify `CAPABILITY_AUTO_EXPAND`.

If your application template contains any of the above resources, we recommend that you review all permissions associated with the application before deploying. If you don't specify this parameter for an application that requires capabilities, the call will fail.

Type: Array of type string
Required: False

changeSetName

This property corresponds to the parameter of the same name for the *AWS CloudFormation CreateChangeSet* API.

Type: string
Required: False

clientToken

This property corresponds to the parameter of the same name for the *AWS CloudFormation CreateChangeSet* API.

Type: string
Required: False

description

This property corresponds to the parameter of the same name for the *AWS CloudFormation CreateChangeSet* API.

Type: string
Required: False

notificationArns

This property corresponds to the parameter of the same name for the *AWS CloudFormation* *CreateChangeSet* API.

Type: Array of type string
Required: False

resourceTypes

This property corresponds to the parameter of the same name for the *AWS CloudFormation* *CreateChangeSet* API.

Type: Array of type string
Required: False

rollbackConfiguration

This property corresponds to the parameter of the same name for the *AWS CloudFormation* *CreateChangeSet* API.

Type: RollbackConfiguration
Required: False

tags

This property corresponds to the parameter of the same name for the *AWS CloudFormation* *CreateChangeSet* API.

Type: Array of type Tag
Required: False

ForbiddenException

The client is not authenticated.

message

The client is not authenticated.

Type: string
Required: False

errorCode

403

Type: string
Required: False

InternalServerErrorException

The AWS Serverless Application Repository service encountered an internal error.

message

The AWS Serverless Application Repository service encountered an internal error.

Type: string
Required: False

errorCode

500

Type: string
Required: False

ParameterValue

Parameter value of the application.

name

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

Type: string
Required: True

value

The input value associated with the parameter.

Type: string
Required: True

RollbackConfiguration

This property corresponds to the *AWS CloudFormation* *RollbackConfiguration* Data Type.

rollbackTriggers

This property corresponds to the content of the same name for the *AWS CloudFormation* *RollbackConfiguration* Data Type.

Type: Array of type RollbackTrigger
Required: False

monitoringTimeInMinutes

This property corresponds to the content of the same name for the *AWS CloudFormation* *RollbackConfiguration* Data Type.

Type: integer
Required: False

RollbackTrigger

This property corresponds to the *AWS CloudFormation* *RollbackTrigger* Data Type.

arn

This property corresponds to the content of the same name for the *AWS CloudFormation* *RollbackTrigger* Data Type.

Type: string

Required: True

type

This property corresponds to the content of the same name for the *AWS CloudFormation* *RollbackTrigger* Data Type.

Type: string
Required: True

Tag

This property corresponds to the *AWS CloudFormation* *Tag* Data Type.

key

This property corresponds to the content of the same name for the *AWS CloudFormation* *Tag* Data Type.

Type: string
Required: True

value

This property corresponds to the content of the same name for the *AWS CloudFormation* *Tag* Data Type.

Type: string
Required: True

TooManyRequestsException

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

message

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

Type: string
Required: False

errorCode

429

Type: string
Required: False

See also

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

CreateCloudFormationChangeSet

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

Applications applicationId Dependencies

URI

`/applications/applicationId/dependencies`

HTTP methods

GET

Operation ID: `ListApplicationDependencies`

Retrieves the list of applications nested in the containing application.

Path parameters

Name	Type	Required	Description
applicationId	String	True	The Amazon Resource Name (ARN) of the application.

Query parameters

Name	Type	Required	Description
nextToken	String	False	A token to specify where to start paginating.
maxItems	String	False	The total number of items to return.
semanticVersion	String	False	The semantic version of the application to get.

Responses

Status code	Response model	Description
200	ApplicationDependencyPage	Success
400	BadRequestException	One of the parameters in the request is invalid.
403	ForbiddenException	The client is not authenticated.
404	NotFoundException	The resource (for example, an access policy statement

Status code	Response model	Description
		) specified in the request doesn't exist.
429	TooManyRequestsException	The client is sending more than the allowed number of requests per unit of time.
500	InternalServerErrorException	The AWS Serverless Application Repository service encountered an internal error.

OPTIONS

Path parameters

Name	Type	Required	Description
applicationId	String	True	The Amazon Resource Name (ARN) of the application.

Responses

Status code	Response model	Description
200	None	200 response

Schemas

Response bodies

ApplicationDependencyPage schema

```
{
  "dependencies": [
    {
```

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

BadRequestException schema

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

ForbiddenException schema

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

NotFoundException schema

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

TooManyRequestsException schema

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

InternalServerErrorException schema

```
{
```

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

Properties

ApplicationDependencyPage

A list of application summaries nested in the application.

dependencies

An array of application summaries nested in the application.

Type: Array of type ApplicationDependencySummary
Required: True

nextToken

The token to request the next page of results.

Type: string
Required: False

ApplicationDependencySummary

A nested application summary.

applicationId

The Amazon Resource Name (ARN) of the nested application.

Type: string
Required: True

semanticVersion

The semantic version of the nested application.

Type: string
Required: True

BadRequestException

One of the parameters in the request is invalid.

message

One of the parameters in the request is invalid.

Type: string
Required: False

errorCode

400

Type: string
Required: False

ForbiddenException

The client is not authenticated.

message

The client is not authenticated.

Type: string
Required: False

errorCode

403

Type: string
Required: False

InternalServerErrorException

The AWS Serverless Application Repository service encountered an internal error.

message

The AWS Serverless Application Repository service encountered an internal error.

Type: string
Required: False

errorCode

500

Type: string
Required: False

NotFoundException

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

message

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

Type: string
Required: False

errorCode

404

Type: string
Required: False

TooManyRequestsException

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

message

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

Type: string
Required: False

errorCode

429

Type: string
Required: False

See also

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

ListApplicationDependencies

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

Applications applicationId Policy

URI

`/applications/applicationId/policy`

HTTP methods

GET

Operation ID: `GetApplicationPolicy`

Retrieves the policy for the application.

Path parameters

Name	Type	Required	Description
applicationId	String	True	The Amazon Resource Name (ARN) of the application.

Responses

Status code	Response model	Description
200	ApplicationPolicy	Success
400	BadRequestException	One of the parameters in the request is invalid.
403	ForbiddenException	The client is not authentic ated.
404	NotFoundException	The resource (for example, an access policy statement) specified in the request doesn't exist.
429	TooManyRequestsExc eption	The client is sending more than the allowed number of requests per unit of time.
500	InternalServerErro rException	The AWS Serverless Applicati on Repository service encountered an internal error.

PUT

Operation ID: `PutApplicationPolicy`

Sets the permission policy for an application. For the list of actions supported for this operation, see Application Permissions .

Path parameters

Name	Type	Required	Description
applicationId	String	True	The Amazon Resource Name (ARN) of the application.

Responses

Status code	Response model	Description
200	ApplicationPolicy	Success
400	BadRequestException	One of the parameters in the request is invalid.
403	ForbiddenException	The client is not authentic ated.
404	NotFoundException	The resource (for example, an access policy statement) specified in the request doesn't exist.
429	TooManyRequestsExc eption	The client is sending more than the allowed number of requests per unit of time.
500	InternalServerErro rException	The AWS Serverless Applicati on Repository service encountered an internal error.

OPTIONS

Path parameters

Name	Type	Required	Description
applicationId	String	True	The Amazon Resource Name (ARN) of the application.

Responses

Status code	Response model	Description
200	None	200 response

Schemas

Request bodies

PUT schema

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

Response bodies

ApplicationPolicy schema

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

BadRequestException schema

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

ForbiddenException schema

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

NotFoundException schema

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

TooManyRequestsException schema

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

InternalServerErrorException schema

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

Properties

ApplicationPolicy

Policy statements applied to the application.

statements

An array of policy statements applied to the application.

Type: Array of type ApplicationPolicyStatement
Required: True

ApplicationPolicyStatement

Policy statement applied to the application.

statementId

A unique ID for the statement.

Type: string
Required: False

principals

An array of AWS account IDs to share the application with, or * to make the application public.

Type: Array of type string
Required: True

actions

For the list of actions supported for this operation, see Application Permissions.

Type: Array of type string
Required: True

principalOrgIDs

The AWS Organizations ID to share the application with.

Type: Array of type string
Required: False

BadRequestException

One of the parameters in the request is invalid.

message

One of the parameters in the request is invalid.

Type: string
Required: False

errorCode

400

Type: string
Required: False

ForbiddenException

The client is not authenticated.

message

The client is not authenticated.

Type: string
Required: False

errorCode

403

Type: string
Required: False

InternalServerErrorException

The AWS Serverless Application Repository service encountered an internal error.

message

The AWS Serverless Application Repository service encountered an internal error.

Type: string
Required: False

errorCode

500

Type: string
Required: False

NotFoundException

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

message

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

Type: string
Required: False

errorCode

404

Type: string
Required: False

TooManyRequestsException

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

message

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

Type: string
Required: False

errorCode

429

Type: string
Required: False

See also

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

GetApplicationPolicy

- AWS Command Line Interface
- AWS SDK for .NET

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

PutApplicationPolicy

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

Applications applicationId Templates

URI

`/applications/applicationId/templates`

HTTP methods

POST

Operation ID: `CreateCloudFormationTemplate`

Creates an AWS CloudFormation template.

Path parameters

Name	Type	Required	Description
applicationId	String	True	The Amazon Resource Name (ARN) of the application.

Responses

Status code	Response model	Description
201	TemplateDetails	Success
400	BadRequestException	One of the parameters in the request is invalid.
403	ForbiddenException	The client is not authentic ated.
404	NotFoundException	The resource (for example, an access policy statement) specified in the request doesn't exist.
429	TooManyRequestsExc eption	The client is sending more than the allowed number of requests per unit of time.
500	InternalServerErro rException	The AWS Serverless Applicati on Repository service encountered an internal error.

OPTIONS

Path parameters

Name	Type	Required	Description
applicationId	String	True	The Amazon Resource Name (ARN) of the application.

Responses

Status code	Response model	Description
200	None	200 response

Schemas

Request bodies

POST schema

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

Response bodies

TemplateDetails schema

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

```
}
```

BadRequestException schema

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

ForbiddenException schema

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

NotFoundException schema

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

TooManyRequestsException schema

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

InternalServerErrorException schema

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

Properties

BadRequestException

One of the parameters in the request is invalid.

message

One of the parameters in the request is invalid.

Type: string
Required: False

errorCode

400

Type: string
Required: False

CreateCloudFormationTemplateInput

Create a template request.

semanticVersion

The semantic version of the application:

https://semver.org/

Type: string
Required: False

ForbiddenException

The client is not authenticated.

message

The client is not authenticated.

Type: string
Required: False

errorCode

403

Type: string
Required: False

InternalServerErrorException

The AWS Serverless Application Repository service encountered an internal error.

message

The AWS Serverless Application Repository service encountered an internal error.

Type: string
Required: False

errorCode

500

Type: string
Required: False

NotFoundException

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

message

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

Type: string
Required: False

errorCode

404

Type: string
Required: False

TemplateDetails

Details of the template.

templateId

The UUID returned by CreateCloudFormationTemplate.

Pattern: [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}

Type: string
Required: True

templateUrl

A link to the template that can be used to deploy the application using AWS CloudFormation.

Type: string
Required: True

applicationId

The application Amazon Resource Name (ARN).

Type: string
Required: True

semanticVersion

The semantic version of the application:

https://semver.org/

Type: string

Required: True

status

Status of the template creation workflow.

Possible values: `PREPARING | ACTIVE | EXPIRED`

Type: string
Required: True
Values: `PREPARING | ACTIVE | EXPIRED`

creationTime

The date and time this resource was created.

Type: string
Required: True

expirationTime

The date and time this template expires. Templates expire 1 hour after creation.

Type: string
Required: True

TooManyRequestsException

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

message

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

Type: string
Required: False

errorCode

429

Type: string

Required: False

See also

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

CreateCloudFormationTemplate

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

Applications applicationId Templates templateId

URI

`/applications/applicationId/templates/templateId`

HTTP methods

GET

Operation ID: `GetCloudFormationTemplate`

Gets the specified AWS CloudFormation template.

Path parameters

Name	Type	Required	Description
applicationId	String	True	The Amazon Resource Name (ARN) of the application.
templateId	String	True	The UUID returned by CreateCloudFormationTemplate. Pattern: [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}

Responses

Status code	Response model	Description
200	TemplateDetails	Success
400	BadRequestException	One of the parameters in the request is invalid.
403	ForbiddenException	The client is not authenticated.
404	NotFoundException	The resource (for example, an access policy statement) specified in the request doesn't exist.
429	TooManyRequestsException	The client is sending more than the allowed number of requests per unit of time.

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

OPTIONS

Path parameters

Name	Type	Required	Description
applicationId	String	True	The Amazon Resource Name (ARN) of the application.
templateId	String	True	The UUID returned by CreateCloudFormationTemplate. Pattern: [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}

Responses

Status code	Response model	Description
200	None	200 response

Schemas

Response bodies

TemplateDetails schema

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

BadRequestException schema

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

ForbiddenException schema

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

NotFoundException schema

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

TooManyRequestsException schema

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

```
}
```

InternalServerErrorException schema

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

Properties

BadRequestException

One of the parameters in the request is invalid.

message

One of the parameters in the request is invalid.

Type: string
Required: False

errorCode

400

Type: string
Required: False

ForbiddenException

The client is not authenticated.

message

The client is not authenticated.

Type: string
Required: False

errorCode

403

Type: string
Required: False

InternalServerErrorException

The AWS Serverless Application Repository service encountered an internal error.

message

The AWS Serverless Application Repository service encountered an internal error.

Type: string
Required: False

errorCode

500

Type: string
Required: False

NotFoundException

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

message

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

Type: string
Required: False

errorCode

404

Type: string
Required: False

TemplateDetails

Details of the template.

templateId

The UUID returned by CreateCloudFormationTemplate.

Pattern: [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}

Type: string
Required: True

templateUrl

A link to the template that can be used to deploy the application using AWS CloudFormation.

Type: string
Required: True

applicationId

The application Amazon Resource Name (ARN).

Type: string
Required: True

semanticVersion

The semantic version of the application:

https://semver.org/

Type: string
Required: True

status

Status of the template creation workflow.

Possible values: `PREPARING | ACTIVE | EXPIRED`

Type: string
Required: True
Values: `PREPARING | ACTIVE | EXPIRED`

creationTime

The date and time this resource was created.

Type: string
Required: True

expirationTime

The date and time this template expires. Templates expire 1 hour after creation.

Type: string
Required: True

TooManyRequestsException

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

message

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

Type: string
Required: False

errorCode

429

Type: string

Required: False

See also

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

GetCloudFormationTemplate

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

Applications applicationId Unshare

URI

`/applications/applicationId/unshare`

HTTP methods

POST

Operation ID: `UnshareApplication`

Unshares an application from an AWS Organization.

This operation can be called only from the organization's management account.

Path parameters

Name	Type	Required	Description
applicationId	String	True	The Amazon Resource Name (ARN) of the application.

Responses

Status code	Response model	Description
204	None	Success
400	BadRequestException	One of the parameters in the request is invalid.
403	ForbiddenException	The client is not authentic ated.
404	NotFoundException	The resource (for example, an access policy statement) specified in the request doesn't exist.
429	TooManyRequestsExc eption	The client is sending more than the allowed number of requests per unit of time.
500	InternalServerErro rException	The AWS Serverless Applicati on Repository service encountered an internal error.

OPTIONS

Path parameters

Name	Type	Required	Description
applicationId	String	True	The Amazon Resource Name (ARN) of the application.

Responses

Status code	Response model	Description
200	None	200 response

Schemas

Request bodies

POST schema

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

Response bodies

BadRequestException schema

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

ForbiddenException schema

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

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

NotFoundException schema

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

TooManyRequestsException schema

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

InternalServerErrorException schema

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

Properties

BadRequestException

One of the parameters in the request is invalid.

message

One of the parameters in the request is invalid.

Type: string
Required: False

errorCode

400

Type: string
Required: False

ForbiddenException

The client is not authenticated.

message

The client is not authenticated.

Type: string
Required: False

errorCode

403

Type: string
Required: False

InternalServerErrorException

The AWS Serverless Application Repository service encountered an internal error.

message

The AWS Serverless Application Repository service encountered an internal error.

Type: string
Required: False

errorCode

500

Type: string
Required: False

NotFoundException

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

message

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

Type: string
Required: False

errorCode

404

Type: string
Required: False

TooManyRequestsException

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

message

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

Type: string
Required: False

errorCode

429

Type: string
Required: False

UnshareApplicationInput

Unshare application request.

organizationId

The AWS Organizations ID to unshare the application from.

Type: string
Required: True

See also

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

UnshareApplication

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

Applications applicationId Versions

URI

`/applications/applicationId/versions`

HTTP methods

GET

Operation ID: `ListApplicationVersions`

Lists versions for the specified application.

Path parameters

Name	Type	Required	Description
applicationId	String	True	The Amazon Resource Name (ARN) of the application.

Query parameters

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

Responses

Status code	Response model	Description
200	ApplicationVersionPage	Success
400	BadRequestException	One of the parameters in the request is invalid.
403	ForbiddenException	The client is not authentic ated.
404	NotFoundException	The resource (for example, an access policy statement) specified in the request doesn't exist.

Status code	Response model	Description
429	TooManyRequestsException	The client is sending more than the allowed number of requests per unit of time.
500	InternalServerErrorException	The AWS Serverless Application Repository service encountered an internal error.

OPTIONS

Path parameters

Name	Type	Required	Description
applicationId	String	True	The Amazon Resource Name (ARN) of the application.

Responses

Status code	Response model	Description
200	None	200 response

Schemas

Response bodies

ApplicationVersionPage schema

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

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

BadRequestException schema

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

ForbiddenException schema

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

NotFoundException schema

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

TooManyRequestsException schema

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

InternalServerErrorException schema

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

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

Properties

ApplicationVersionPage

A list of version summaries for the application.

versions

An array of version summaries for the application.

Type: Array of type VersionSummary
Required: True

nextToken

The token to request the next page of results.

Type: string
Required: False

BadRequestException

One of the parameters in the request is invalid.

message

One of the parameters in the request is invalid.

Type: string
Required: False

errorCode

400

Type: string
Required: False

ForbiddenException

The client is not authenticated.

message

The client is not authenticated.

Type: string
Required: False

errorCode

403

Type: string
Required: False

InternalServerErrorException

The AWS Serverless Application Repository service encountered an internal error.

message

The AWS Serverless Application Repository service encountered an internal error.

Type: string
Required: False

errorCode

500

Type: string
Required: False

NotFoundException

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

message

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

Type: string
Required: False

errorCode

404

Type: string
Required: False

TooManyRequestsException

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

message

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

Type: string
Required: False

errorCode

429

Type: string
Required: False

VersionSummary

An application version summary.

applicationId

The application Amazon Resource Name (ARN).

Type: string

Required: True

semanticVersion

The semantic version of the application:

https://semver.org/

Type: string
Required: True

sourceCodeUrl

A link to a public repository for the source code of your application, for example the URL of a specific GitHub commit.

Type: string
Required: False

creationTime

The date and time this resource was created.

Type: string
Required: True

See also

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

ListApplicationVersions

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2

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

Applications applicationId Versions semanticVersion

URI

`/applications/applicationId/versions/semanticVersion`

HTTP methods

PUT

Operation ID: `CreateApplicationVersion`

Creates an application version.

Path parameters

Name	Type	Required	Description
applicationId	String	True	The Amazon Resource Name (ARN) of the application.
semanticVersion	String	True	The semantic version of the new version.

Responses

Status code	Response model	Description
201	Version	Success
400	BadRequestException	One of the parameters in the request is invalid.

Status code	Response model	Description
403	ForbiddenException	The client is not authenticated.
409	ConflictException	The resource already exists.
429	TooManyRequestsException	The client is sending more than the allowed number of requests per unit of time.
500	InternalServerErrorException	The AWS Serverless Application Repository service encountered an internal error.

OPTIONS

Path parameters

Name	Type	Required	Description
applicationId	String	True	The Amazon Resource Name (ARN) of the application.
semanticVersion	String	True	The semantic version of the new version.

Responses

Status code	Response model	Description
200	None	200 response

Schemas

Request bodies

PUT schema

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

Response bodies

Version schema

```
{
  "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 schema

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

ForbiddenException schema

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

ConflictException schema

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

TooManyRequestsException schema

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

InternalServerErrorException schema

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

```
}
```

Properties

BadRequestException

One of the parameters in the request is invalid.

message

One of the parameters in the request is invalid.

Type: string
Required: False

errorCode

400

Type: string
Required: False

Capability

Values that must be specified in order to deploy some applications.

```
CAPABILITY_IAM
CAPABILITY_NAMED_IAM
CAPABILITY_AUTO_EXPAND
CAPABILITY_RESOURCE_POLICY
```

ConflictException

The resource already exists.

message

The resource already exists.

Type: string

Required: False

errorCode

409

Type: string
Required: False

CreateApplicationVersionInput

Create a version request.

templateBody

The raw packaged AWS SAM template of your application.

Type: string
Required: False

templateUrl

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

Type: string
Required: False

sourceCodeUrl

A link to a public repository for the source code of your application, for example the URL of a specific GitHub commit.

Type: string
Required: False

sourceCodeArchiveUrl

A link to the S3 object that contains the ZIP archive of the source code for this version of your application.

Maximum size 50 MB

Type: string
Required: False

ForbiddenException

The client is not authenticated.

message

The client is not authenticated.

Type: string
Required: False

errorCode

403

Type: string
Required: False

InternalServerErrorException

The AWS Serverless Application Repository service encountered an internal error.

message

The AWS Serverless Application Repository service encountered an internal error.

Type: string
Required: False

errorCode

500

Type: string
Required: False

ParameterDefinition

Parameters supported by the application.

name

The name of the parameter.

Type: string
Required: True

defaultValue

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

Type: string
Required: False

description

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

Type: string
Required: False

type

The type of the parameter.

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

`String`: A literal string.

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

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

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

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

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

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

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

Type: string
Required: False

noEcho

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

Type: boolean
Required: False

allowedPattern

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

Type: string
Required: False

constraintDescription

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

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

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

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

Type: string
Required: False

minValue

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

Type: integer
Required: False

maxValue

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

Type: integer
Required: False

minLength

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

Type: integer
Required: False

maxLength

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

Type: integer
Required: False

allowedValues

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

Type: Array of type string
Required: False

referencedByResources

A list of AWS SAM resources that use this parameter.

Type: Array of type string
Required: True

TooManyRequestsException

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

message

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

Type: string
Required: False

errorCode

429

Type: string
Required: False

Version

Application version details.

applicationId

The application Amazon Resource Name (ARN).

Type: string
Required: True

semanticVersion

The semantic version of the application:

https://semver.org/

Type: string
Required: True

sourceCodeUrl

A link to a public repository for the source code of your application, for example the URL of a specific GitHub commit.

Type: string
Required: False

sourceCodeArchiveUrl

A link to the S3 object that contains the ZIP archive of the source code for this version of your application.

Maximum size 50 MB

Type: string
Required: False

templateUrl

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

Type: string
Required: True

creationTime

The date and time this resource was created.

Type: string
Required: True

parameterDefinitions

An array of parameter types supported by the application.

Type: Array of type ParameterDefinition
Required: True

requiredCapabilities

A list of values that you must specify before you can deploy certain applications. Some applications might include resources that can affect permissions in your AWS account, for example, by creating new AWS Identity and Access Management (IAM) users. For those applications, you must explicitly acknowledge their capabilities by specifying this parameter.

The only valid values are `CAPABILITY_IAM`, `CAPABILITY_NAMED_IAM`, `CAPABILITY_RESOURCE_POLICY`, and `CAPABILITY_AUTO_EXPAND`.

The following resources require you to specify `CAPABILITY_IAM` or `CAPABILITY_NAMED_IAM`: AWS::IAM::Group, AWS::IAM::InstanceProfile, AWS::IAM::Policy, and AWS::IAM::Role. If the application contains IAM resources, you can specify either `CAPABILITY_IAM` or `CAPABILITY_NAMED_IAM`. If the application contains IAM resources with custom names, you must specify `CAPABILITY_NAMED_IAM`.

The following resources require you to specify `CAPABILITY_RESOURCE_POLICY`: AWS::Lambda::Permission, AWS::IAM:Policy, AWS::ApplicationAutoScaling::ScalingPolicy, AWS::S3::BucketPolicy, AWS::SQS::QueuePolicy, and AWS::SNS::TopicPolicy.

Applications that contain one or more nested applications require you to specify `CAPABILITY_AUTO_EXPAND`.

If your application template contains any of the above resources, we recommend that you review all permissions associated with the application before deploying. If you don't specify this parameter for an application that requires capabilities, the call will fail.

Type: Array of type Capability
Required: True

resourcesSupported

Whether all of the AWS resources contained in this application are supported in the region in which it is being retrieved.

Type: boolean
Required: True

See also

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

CreateApplicationVersion

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

Document History

- **API version: latest**
- **Latest documentation update:** March 10, 2020

The following table describes the important changes in each release of the *AWS Serverless Application Repository Developer Guide*. For notification about updates to this documentation, you can subscribe to an RSS feed.

Change	Description	Date
Updates to sharing and restricting access to applications	Added support for sharing applications to accounts in an AWS Organization, and restricting access to public applications for AWS accounts and AWS Organizations. For more examples for sharing applications to users in an organization, see AWS Serverless Application Repository Application Policy Examples. For examples for restricting access to public applications, see AWS Serverless Application Repository Identity-Based Policy Examples.	March 10, 2020
New supported resources	Added support for a number of additional resources . For the complete list of supported resources, see List of Supported AWS Resources.	January 17, 2020

China Regions	The AWS Serverless Application Repository is now available in the China Regions, Beijing and Ningxia. For more information about AWS Serverless Application Repository regions and endpoints, see Regions and Endpoints in the *AWS General Reference*.	January 15, 2020
Updated Security section for consistency with other AWS services.	For more information, see Security.	January 2, 2020
Simplified process for publishing applications	The new `sam publish` command in the AWS SAM CLI simplifies the process for publishing serverless applications in the AWS Serverless Application Repository. For an end-to-end tutorial on downloading and publishing a sample application, see Quick Start: Publishing Applications. For instructions on publishing an application that you have already developed and tested in the AWS Cloud, see Publishing an Application through the AWS SAM CLI.	December 21, 2018

Nested Application and Layers support	Added support for Nested Applications and Layers. This includes updates to Supported AWS Resources and Acknowledging Application Capabilities.	November 29, 2018
Publishing applications with custom IAM roles and resource policies	Added support for publishing applications with custom IAM roles and resource policies. This includes updates to the Consuming Applications and Publishing Applications workflows and updates to Supported AWS Resources and API Reference in the *AWS Serverless Application Repository Developer Guide*.	November 16, 2018
Policy Template updates	Updates to supported Policy Templates in the *AWS Serverless Application Repository Developer Guide*.	September 26, 2018
Documentation updates	Added Authentication and Access Control topic to the *AWS Serverless Application Repository Developer Guide*.	July 2, 2018

Public release	Public release of the AWS Serverless Application Repository, which is now available in 14 AWS Regions. For more information about the AWS Regions where the AWS Serverless Application Repository is available and AWS Serverless Application Repository endpoints, see Regions and Endpoints in the *AWS General Reference*.	February 20, 2018
New guide	This is the first, preview release of the *AWS Serverless Application Repository Developer Guide*.	November 30, 2017

AWS Glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.