AWS Serverless Application Model: Developer Guide
Copyright © 2021 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.
# Table of Contents

What is AWS SAM? .......................................................... 1  
Benefits of using AWS SAM .......................................................... 1  
Next step .................................................................. 2  
Getting started ................................................................. 3  
Installing the AWS SAM CLI ..................................................... 3  
Linux ........................................................................ 3  
Windows ....................................................................... 9  
macOS ......................................................................... 11  
Setting up AWS credentials ...................................................... 13  
Using the AWS CLI ............................................................... 14  
Not using the AWS CLI ......................................................... 14  
Tutorial: Hello World application .................................................. 14  
Prerequisites .................................................................. 15  
Step 1: Download a sample AWS SAM application ....................... 16  
Step 2: Build your application .................................................. 17  
Step 3: Deploy your application to the AWS Cloud ....................... 17  
Step 4: (Optional) Test your application locally .......................... 20  
Troubleshooting .................................................................. 22  
Clean up .......................................................................... 24  
Conclusion ......................................................................... 24  
Next steps .......................................................................... 24  
AWS SAM specification .......................................................... 26  
Template anatomy .................................................................. 26  
YAML ............................................................................ 26  
Template sections .................................................................. 27  
Next steps .......................................................................... 28  
Globals ............................................................................. 28  
Resource and property reference .................................................. 32  
AWS::Serverless::Api ......................................................... 32  
AWS::Serverless::Application .................................................. 63  
AWS::Serverless::Function ..................................................... 67  
AWS::Serverless::HttpApi ..................................................... 128  
AWS::Serverless::LayerVersion .................................................. 147  
AWS::Serverless::SimpleTable ............................................... 151  
AWS::Serverless::StateMachine .................................................. 154  
Resource attributes ................................................................ 173  
Intrinsic functions .............................................................. 174  
Generated resources ........................................................... 174  
Referencing generated AWS CloudFormation resources .................. 174  
Generated AWS CloudFormation resource reference ..................... 175  
AWS::Serverless::Api ......................................................... 176  
AWS::Serverless::Function ..................................................... 177  
AWS::Serverless::HttpApi ..................................................... 179  
API Gateway extensions ....................................................... 180  
Authoring .......................................................................... 182  
Validating AWS SAM template files .............................................. 182  
Working with layers .............................................................. 182  
Including layers in your application ............................................ 183  
How layers are cached locally .................................................. 183  
Using nested applications .......................................................... 184  
Defining a nested application from the AWS Serverless Application Repository .................................................. 185  
Defining a nested application from the local file system .................. 186  
Deploying nested applications .................................................. 186  
Controlling access to APIs ....................................................... 186  

AWS Serverless Application Model Developer Guide
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Publishing</td>
<td>225</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>225</td>
</tr>
<tr>
<td>Publishing a new application</td>
<td>226</td>
</tr>
<tr>
<td>Step 1: Add a Metadata section to the AWS SAM template</td>
<td>226</td>
</tr>
<tr>
<td>Step 2: Package the application</td>
<td>226</td>
</tr>
<tr>
<td>Step 3: Publish the application</td>
<td>227</td>
</tr>
<tr>
<td>Step 4: Share the application (optional)</td>
<td>227</td>
</tr>
<tr>
<td>Publishing a new version of an existing application</td>
<td>227</td>
</tr>
<tr>
<td>Additional topics</td>
<td>228</td>
</tr>
<tr>
<td>Metadata section properties</td>
<td>228</td>
</tr>
<tr>
<td>Properties</td>
<td>228</td>
</tr>
<tr>
<td>Use cases</td>
<td>229</td>
</tr>
<tr>
<td>Example</td>
<td>230</td>
</tr>
<tr>
<td>Example applications</td>
<td>231</td>
</tr>
<tr>
<td>Process DynamoDB events</td>
<td>231</td>
</tr>
<tr>
<td>Before you begin</td>
<td>231</td>
</tr>
<tr>
<td>Step 1: Initialize the application</td>
<td>231</td>
</tr>
<tr>
<td>Step 2: Test the application locally</td>
<td>231</td>
</tr>
<tr>
<td>Step 3: Package the application</td>
<td>232</td>
</tr>
<tr>
<td>Step 4: Deploy the application</td>
<td>232</td>
</tr>
<tr>
<td>Next steps</td>
<td>232</td>
</tr>
<tr>
<td>Process Amazon S3 events</td>
<td>233</td>
</tr>
<tr>
<td>Before you begin</td>
<td>233</td>
</tr>
<tr>
<td>Step 1: Initialize the application</td>
<td>233</td>
</tr>
<tr>
<td>Step 2: Package the application</td>
<td>234</td>
</tr>
<tr>
<td>Step 3: Deploy the application</td>
<td>234</td>
</tr>
<tr>
<td>Step 4: Test the application locally</td>
<td>235</td>
</tr>
<tr>
<td>Next steps</td>
<td>235</td>
</tr>
<tr>
<td>AWS SAM reference</td>
<td>236</td>
</tr>
<tr>
<td>AWS SAM specification</td>
<td>236</td>
</tr>
<tr>
<td>AWS SAM CLI command reference</td>
<td>236</td>
</tr>
<tr>
<td>AWS SAM policy templates</td>
<td>236</td>
</tr>
<tr>
<td>Topics</td>
<td>236</td>
</tr>
<tr>
<td>AWS SAM CLI command reference</td>
<td>237</td>
</tr>
<tr>
<td>sam build</td>
<td>237</td>
</tr>
<tr>
<td>sam deploy</td>
<td>241</td>
</tr>
<tr>
<td>sam init</td>
<td>244</td>
</tr>
<tr>
<td>sam local generate-event</td>
<td>247</td>
</tr>
<tr>
<td>sam local invoke</td>
<td>248</td>
</tr>
<tr>
<td>sam local start-api</td>
<td>250</td>
</tr>
<tr>
<td>sam local start-lambda</td>
<td>252</td>
</tr>
<tr>
<td>sam logs</td>
<td>254</td>
</tr>
<tr>
<td>sam package</td>
<td>255</td>
</tr>
<tr>
<td>sam publish</td>
<td>257</td>
</tr>
<tr>
<td>sam validate</td>
<td>258</td>
</tr>
<tr>
<td>AWS SAM CLI configuration file</td>
<td>259</td>
</tr>
<tr>
<td>Example</td>
<td>259</td>
</tr>
<tr>
<td>Configuration file rules</td>
<td>259</td>
</tr>
<tr>
<td>Writing configurations with sam deploy --guided</td>
<td>260</td>
</tr>
<tr>
<td>AWS SAM policy templates</td>
<td>261</td>
</tr>
<tr>
<td>Syntax</td>
<td>261</td>
</tr>
<tr>
<td>Examples</td>
<td>261</td>
</tr>
<tr>
<td>Policy template table</td>
<td>262</td>
</tr>
<tr>
<td>Troubleshooting</td>
<td>266</td>
</tr>
<tr>
<td>Policy template list</td>
<td>267</td>
</tr>
<tr>
<td>AWS SAM CLI telemetry</td>
<td>302</td>
</tr>
<tr>
<td>Disabling telemetry for a session</td>
<td>302</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Disabling telemetry for your profile in all sessions</td>
<td>302</td>
</tr>
<tr>
<td>Types of information collected</td>
<td>303</td>
</tr>
<tr>
<td>Learn more</td>
<td>303</td>
</tr>
<tr>
<td>Permissions</td>
<td>303</td>
</tr>
<tr>
<td>Grant administrator permissions</td>
<td>304</td>
</tr>
<tr>
<td>Attach necessary AWS managed policies</td>
<td>304</td>
</tr>
<tr>
<td>Grant specific IAM permissions</td>
<td>304</td>
</tr>
<tr>
<td>Important notes</td>
<td>307</td>
</tr>
<tr>
<td>Installing AWS SAM CLI on 32-bit Windows</td>
<td>307</td>
</tr>
<tr>
<td>Document history</td>
<td>308</td>
</tr>
</tbody>
</table>
What is the AWS Serverless Application Model (AWS SAM)?

The AWS Serverless Application Model (AWS SAM) is an open-source framework that you can use to build serverless applications on AWS.

A serverless application is a combination of Lambda functions, event sources, and other resources that work together to perform tasks. Note that a serverless application is more than just a Lambda function—it can include additional resources such as APIs, databases, and event source mappings.

You can use AWS SAM to define your serverless applications. AWS SAM consists of the following components:

- **AWS SAM template specification**. You use this specification to define your serverless application. It provides you with a simple and clean syntax to describe the functions, APIs, permissions, configurations, and events that make up a serverless application. You use an AWS SAM template file to operate on a single, deployable, versioned entity that’s your serverless application. For the full AWS SAM template specification, see AWS Serverless Application Model ( AWS SAM) specification (p. 26).

- **AWS SAM command line interface (AWS SAM CLI)**. You use this tool to build serverless applications that are defined by AWS SAM templates. The CLI provides commands that enable you to verify that AWS SAM template files are written according to the specification, invoke Lambda functions locally, step-through debug Lambda functions, package and deploy serverless applications to the AWS Cloud, and so on. For details about how to use the AWS SAM CLI, including the full AWS SAM CLI Command Reference, see AWS SAM CLI command reference (p. 236).

This guide shows you how to use AWS SAM to define, test, and deploy a simple serverless application. It also provides an example application (p. 14) that you can download, test locally, and deploy to the AWS Cloud. You can use this example application as a starting point for developing your own serverless applications.

Benefits of using AWS SAM

Because AWS SAM integrates with other AWS services, creating serverless applications with AWS SAM provides the following benefits:

- **Single-deployment configuration**. AWS SAM makes it easy to organize related components and resources, and operate on a single stack. You can use AWS SAM to share configuration (such as memory and timeouts) between resources, and deploy all related resources together as a single, versioned entity.

- **Extension of AWS CloudFormation**. Because AWS SAM is an extension of AWS CloudFormation, you get the reliable deployment capabilities of AWS CloudFormation. You can define resources by using AWS CloudFormation in your AWS SAM template. Also, you can use the full suite of resources, intrinsic functions, and other template features that are available in AWS CloudFormation.
• **Built-in best practices.** You can use AWS SAM to define and deploy your infrastructure as config. This makes it possible for you to use and enforce best practices such as code reviews. Also, with a few lines of configuration, you can enable safe deployments through CodeDeploy, and can enable tracing by using AWS X-Ray.

• **Local debugging and testing.** The AWS SAM CLI lets you locally build, test, and debug serverless applications that are defined by AWS SAM templates. The CLI provides a Lambda-like execution environment locally. It helps you catch issues upfront by providing parity with the actual Lambda execution environment. To step through and debug your code to understand what the code is doing, you can use AWS SAM with AWS toolkits like the AWS Toolkit for JetBrains, AWS Toolkit for PyCharm, AWS Toolkit for IntelliJ, and AWS Toolkit for Visual Studio Code. This tightens the feedback loop by making it possible for you to find and troubleshoot issues that you might run into in the cloud.

• **Deep integration with development tools.** You can use AWS SAM with a suite of AWS tools for building serverless applications. You can discover new applications in the AWS Serverless Application Repository. For authoring, testing, and debugging AWS SAM–based serverless applications, you can use the AWS Cloud9 IDE. To build a deployment pipeline for your serverless applications, you can use CodeBuild, CodeDeploy, and CodePipeline. You can also use AWS CodeStar to get started with a project structure, code repository, and a CI/CD pipeline that's automatically configured for you. To deploy your serverless application, you can use the Jenkins plugin. You can use the Stackery.io toolkit to build production-ready applications.

**Next step**

Getting started with AWS SAM (p. 3)
Getting started with AWS SAM

To get started with AWS SAM, use the AWS SAM CLI to create a serverless application that you can package and deploy in the AWS Cloud. You can run the application both in the AWS Cloud or locally on your development host.

To install the AWS SAM CLI, including everything that needs to be installed or configured to use the AWS SAM CLI, see Installing the AWS SAM CLI (p. 3). After the AWS SAM CLI is installed, you can run through the following tutorial.

Topics
• Installing the AWS SAM CLI (p. 3)
• Setting up AWS credentials (p. 13)
• Tutorial: Deploying a Hello World application (p. 14)

Installing the AWS SAM CLI

AWS SAM provides you with a command line tool, the AWS SAM CLI, that makes it easy for you to create and manage serverless applications. You need to install and configure a few things in order to use the AWS SAM CLI.

To install the AWS SAM CLI, see the following instructions for your development host:

Topics
• Installing the AWS SAM CLI on Linux (p. 3)
• Installing the AWS SAM CLI on Windows (p. 9)
• Installing the AWS SAM CLI on macOS (p. 11)

Installing the AWS SAM CLI on Linux

The AWS SAM command line interface (CLI) is supported on 64-bit versions of recent distributions of CentOS, Fedora, Ubuntu, and Amazon Linux 2. To install the AWS SAM CLI, you must extract or "unzip" the downloaded package. If your operating system doesn't have the built-in unzip command, use an equivalent.

To install and configure the prerequisites for using the AWS SAM CLI on your Linux host, follow these steps:

1. Create an AWS account.
2. Configure AWS Identity and Access Management (IAM) permissions and AWS credentials.
3. Install Docker. Note: Docker is a prerequisite only for testing your application locally or using the --use-container option.
4. Install the AWS SAM CLI.

Step 1: Create an AWS account

If you don't already have an AWS account, see aws.amazon.com and choose Create an AWS Account. For detailed instructions, see How do I create and activate a new AWS account?
Step 2: Configure IAM permissions and AWS credentials

The IAM user that you use with AWS SAM must have sufficient permissions to make necessary AWS service calls and manage AWS resources. The simplest way to ensure that a user has sufficient permissions is to grant administrator privileges to them. For more information, see Creating your first IAM admin user and group in the IAM User Guide.

Note
If you don't want to grant administrator privileges to users who use the AWS Command Line Interface (AWS CLI), you can grant restricted sets of permissions to them. For more information, see Permissions (p. 303).

In addition, to enable the AWS SAM CLI to make AWS service calls, you must set up AWS credentials. For more information, see Setting up AWS credentials (p. 13).

Step 3: Install Docker (optional)

Note
Docker is a prerequisite only for testing your application locally and for building deployment packages using the --use-container option. If you don't plan to use these features initially, you can skip this section or install Docker at a later time.

Docker is an application that runs containers on your Linux machines. AWS SAM provides a local environment that's similar to AWS Lambda to use as a Docker container. You can use this container to build, test, and debug your serverless applications.

To run serverless projects and functions locally with the AWS SAM CLI, you must have Docker installed and working. The AWS SAM CLI uses the DOCKER_HOST environment variable to contact the Docker daemon. The following steps describe how to install, configure, and verify a Docker installation to work with the AWS SAM CLI.

Docker is available on many different operating systems, including most modern Linux distributions, for example, CentOS, Debian, and Ubuntu. For information about installing Docker on your particular operating system, see Get Docker on the Docker Docs website.

If you're using Amazon Linux 2, follow these steps to install Docker:

1. Update the installed packages and package cache on your instance.

   ```bash
   sudo yum update -y
   ```

2. Install the most recent Docker Community Edition package.

   ```bash
   sudo amazon-linux-extras install docker
   ```

3. Start the Docker service.

   ```bash
   sudo service docker start
   ```

4. Add the ec2-user to the docker group so that you can run Docker commands without using sudo.

   ```bash
   sudo usermod -a -G docker ec2-user
   ```

5. Pick up the new docker group permissions by logging out and logging back in again. To do this, close your current SSH terminal window and reconnect to your instance in a new one. Your new SSH session should have the appropriate docker group permissions.

6. Verify that the ec2-user can run Docker commands without using sudo.
You should see the following output, confirming that Docker is installed and running:

<table>
<thead>
<tr>
<th>CONTAINER ID</th>
<th>IMAGE</th>
<th>COMMAND</th>
<th>CREATED</th>
<th>STATUS</th>
</tr>
</thead>
</table>

If you run into issues installing Docker, see the Troubleshooting (p. 6) section later in this guide. Or, see the Troubleshooting section of Post-installation steps for Linux on the Docker Docs website.

**Step 4: Install the AWS SAM CLI**

To install the AWS SAM CLI, follow these steps:

1. Download the AWS SAM CLI zip file to a directory of your choice.
2. Verify the integrity and authenticity of the downloaded installer files by generating a hash value using the following command:

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

   The output should look like the following example:

   ```
   5b9c61df9a92f0e9f7d335a124ae5d94cc10a11b424fa2635e0cfae56b6d3e3  aws-sam-cli-linux-x86_64.zip
   ```

   Compare the hash value with the one for your desired AWS SAM CLI version in the AWS SAM CLI release notes on GitHub.

3. Unzip the installation files into the `sam-installation/` subdirectory.

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

4. Install the AWS SAM CLI.

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

5. Verify the installation.

   ```
   sam --version
   ```

   On successful installation, you should see output like the following:

   ```
   SAM CLI, version 1.18.0
   ```

You're now ready to start development.

**Upgrading**

To upgrade the AWS SAM CLI, perform the same steps as in the previous Install the AWS SAM CLI section, but add the `--update` option to the install command, as follows:
Uninstalling

To uninstall the AWS SAM CLI, you must delete the symlink and installation directory by running the following commands:

1. Locate the symlink and install paths.
   - Find the symlink using the `which` command:
     ```bash
     which sam
     ```
     The output shows the path where the AWS SAM binaries are located, for example:
     ```bash
     /usr/local/bin/sam
     ```
   - Find the directory that the symlink points to using the `ls` command:
     ```bash
     ls -l /usr/local/bin/sam
     ```
     In the following example, the installation directory is `/usr/local/aws-sam-cli`.

2. Delete the symlink.
   ```bash
   sudo rm /usr/local/bin/sam
   ```

3. Delete the installation directory.
   ```bash
   sudo rm -rf /usr/local/aws-sam-cli
   ```

Troubleshooting

Docker error: "Cannot connect to the Docker daemon. Is the docker daemon running on this host?"

In some cases, to provide permissions for the `ec2-user` to access the Docker daemon, you might have to reboot your instance. If you receive this error, try rebooting your instance.

Shell error: "command not found"

If you receive this error, your shell can't locate the AWS SAM CLI executable in the path. Verify the location of the directory where you installed the AWS SAM CLI executable, and then verify that the directory is on your path.

AWS SAM CLI error: "'/lib64/libc.so.6: version `GLIBC_2.14' not found (required by /usr/local/aws-sam-cli/dist/libz.so.1)"

If you receive this error, you're using an unsupported version of Linux, and the built-in glibc version is out of date. Try either of the following:
Upgrade your Linux host to the 64-bit version of a recent distribution of CentOS, Fedora, Ubuntu, or Amazon Linux 2.

Follow the instructions for Installing the AWS SAM CLI on Linux using Homebrew (p. 7).

Next steps

You're now ready to begin building your own serverless applications using AWS SAM. To start with a sample serverless application, choose one of the following links:

- Tutorial: Deploying a Hello World application (p. 14) – Step-by-step instructions to download, build, and deploy a simple serverless application.
- AWS SAM example applications in GitHub – Sample applications in the AWS SAM GitHub repository that you can further experiment with.

Installing the AWS SAM CLI on Linux using Homebrew

To install the AWS SAM CLI on Linux, you can use the Homebrew package manager. For more information about Homebrew, see Homebrew on Linux on the Homebrew Documentation website.

**Note**

Installing Homebrew changes your environment's default Python version to the one that Homebrew installs.

To install Homebrew, you must first install Git. Git is available on many different operating systems, including most modern Linux distributions. For instructions about installing Git on your particular operating system, see Installing Git on the Git website.

Install Homebrew

After successfully installing Git, to install Homebrew, run the following command:

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"
```

Next, add Homebrew to your PATH by running the following commands. These commands work on all major flavors of Linux by adding either `~/.profile` on Debian and Ubuntu, or `~/.bash_profile` on CentOS, Fedora, and Red Hat.

```
test -d ~/.linuxbrew && eval $(~/.linuxbrew/bin/brew shellenv)
test -d /home/linuxbrew/.linuxbrew && eval $(/home/linuxbrew/.linuxbrew/bin/brew shellenv)
test -x ~/.bash_profile && echo "eval \$(brew --prefix)/bin/brew shellenv)"
 >> ~/.bash_profile
echo "eval \$(brew --prefix)/bin/brew shellenv)" >>~/.profile
```

Verify that Homebrew is installed.

```
brew --version
```

On successful installation of Homebrew, you should see output like the following:

```
Homebrew 2.1.6
Homebrew/homebrew-core (git revision ef21; last commit 2019-06-19)
```
Install the AWS SAM CLI using Homebrew

To install the AWS SAM CLI using Homebrew, run the following commands:

```
brew tap aws/tap
brew install aws-sam-cli
```

Verify the installation.

```
sam --version
```

On successful installation of the AWS SAM CLI, you should see output like the following:

```
SAM CLI, version 1.15.0
```

Upgrading the AWS SAM CLI using Homebrew

To upgrade the AWS SAM CLI using Homebrew, replace `install` with `upgrade` as follows:

```
brew upgrade aws-sam-cli
```

Troubleshooting

Installing Homebrew message: "Enter your password to install to /home/linuxbrew/.linuxbrew"

During the Install Homebrew step, by default you're prompted to provide a password. However, you might not want to set up a password for the current user, for example, when you're setting up a non-interactive environment like CI/CD systems.

If you don't want to set up a password for the current user, you can install Homebrew in non-interactive mode by setting the environment variable `CI=1`. For example:

```
CI=1 /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"
```

Installing AWS SAM CLI error: "The following formulae cannot be installed from bottles and must be built from source. pkg-config, gdbm, openssl@1.1, ncurses, xz and python@3.8"

If you see this error while installing the AWS SAM CLI, you don't have the gcc module installed. Install the gcc module for your Linux distribution.

```
# for Amazon Linux, Amazon Linux 2, CentOS and Red Hat:
sudo yum install gcc
# for Debian and Ubuntu:
sudo apt-get update
sudo apt-get install gcc
```

After installing the gcc module, run the commands in the Install the AWS SAM CLI using Homebrew section again.
Shell error: "command not found"

If you receive this error, your shell can't locate the AWS SAM CLI executable in the PATH. Verify the location of the directory where you installed the AWS SAM CLI executable, and then verify that the directory is on your PATH.

For example, if you used the instructions in this topic to both install Homebrew and use Homebrew to install the AWS SAM CLI, then the AWS SAM CLI executable is installed to the following location:

```
/home/linuxbrew/.linuxbrew/bin/sam
```

## Installing the AWS SAM CLI on Windows

Follow these steps to install and configure the prerequisites for using the AWS SAM command line interface (CLI) on your Windows host:

1. Create an AWS Identity and Access Management (AWS) account.
2. Configure IAM permissions and AWS credentials.
3. Install Docker. **Note:** Docker is a prerequisite only for testing your application locally or using the `--use-container` option.
4. Install the AWS SAM CLI.

### Step 1: Create an AWS account

If you don't already have an AWS account, see [aws.amazon.com](http://aws.amazon.com) and choose **Create an AWS Account**. For detailed instructions, see **Create and Activate an AWS Account**.

### Step 2: Configure IAM permissions and AWS credentials

The IAM user that you use with AWS SAM must have sufficient permissions to make necessary AWS service calls and manage AWS resources. The simplest way to ensure that a user has sufficient permissions is to grant administrator privileges to them. For more information, see **Creating your first IAM admin user and group** in the [IAM User Guide](https://docs.aws.amazon.com/IAM/latest/UserGuide/).  

**Note**

If you don't want to grant administrator privileges to users who use the AWS Command Line Interface (AWS CLI), you can grant restricted sets of permissions to them. For more information, see **Permissions** (p. 303).

In addition, to enable the AWS SAM CLI to make AWS service calls, you must set up AWS credentials. For more information, see **Setting up AWS credentials** (p. 13).

### Step 3: Install Docker (optional)

**Note**

Docker is a prerequisite only for testing your application locally and for building deployment packages using the `--use-container` option. If you don't plan to use these features initially, you can skip this section or install Docker at a later time.

Docker is an application that runs containers on your Linux machines. AWS SAM provides a local environment that's similar to AWS Lambda to use as a Docker container. You can use this container to build, test, and debug your serverless applications.
To run serverless projects and functions locally with the AWS SAM CLI, you must have Docker installed and working. The AWS SAM CLI uses the DOCKER_HOST environment variable to contact the Docker daemon. The following steps describe how to install, configure, and verify a Docker installation to work with the AWS SAM CLI.

1. **Install Docker.**

   Docker Desktop supports the most recent Windows operating system. For legacy versions of Windows, the Docker Toolbox is available. Choose your version of Windows for the correct Docker installation steps:
   - To install Docker for Windows 10, see Install Docker Desktop for Windows.
   - To install Docker for older versions of Windows, see Install Docker Toolbox on Windows.

2. **Configure your shared drives.**

   The AWS SAM CLI requires that the project directory, or any parent directory, is listed in a shared drive. In some cases you must share your drive in order for Docker to function properly.
   - If you’re using Windows 10 in Hyper-V mode, see Docker File Sharing.
   - To share drives on older versions of Windows, see Add Shared Directories.

3. **Verify the installation.**

   After Docker is installed, verify that it’s working. Also confirm that you can run Docker commands from the command line (for example, `docker ps`). You don’t need to install, fetch, or pull any containers—the AWS SAM CLI does this automatically as required.

   If you run into issues installing Docker, see the Logs and troubleshooting section of the Docker installation guide for additional troubleshooting tips.

**Step 4: Install the AWS SAM CLI**

Windows Installer (MSI) files are the package installer files for the Windows operating system.

Follow these steps to install the AWS SAM CLI using the MSI file.

1. **Install the AWS SAM CLI 64-bit.**

   **Note**
   - If you operate on 32-bit system, see Installing AWS SAM CLI on 32-bit Windows (p. 307).

2. **Verify the installation.**

   After completing the installation, verify it by opening a new command prompt or PowerShell prompt. You should be able to invoke `sam` from the command line.

   ```bash
sam --version
   ```

   You should see output like the following after successful installation of the AWS SAM CLI:

   ```
   SAM CLI, version 1.15.0
   ```

3. **Install Git.**

   To download sample applications using the `sam init` command, you must also install Git. For instructions, see Installing Git.
You're now ready to start development.

Uninstalling

To uninstall the AWS SAM CLI using Windows Settings, follow these steps:

1. From the Start menu, search for "Add or remove programs".
2. Select the entry named **AWS SAM Command Line Interface** and choose **Uninstall** to launch the uninstaller.
3. Confirm that you want to uninstall the AWS SAM CLI.

Next steps

You're now ready to begin building your own serverless applications using AWS SAM! If you want to start with sample serverless applications, choose one of the following links:

- **Tutorial: Deploying a Hello World application (p. 14)** – Step-by-step instructions to download, build, and deploy a simple serverless application.
- **AWS SAM example applications in GitHub** – Sample applications in the AWS SAM GitHub repository that you can further experiment with.

Installing the AWS SAM CLI on macOS

Follow these steps to install and configure the prerequisites for using the AWS SAM command line interface (CLI) on your macOS host:

1. Create an AWS account.
2. Configure AWS Identity and Access Management (IAM) permissions and AWS credentials.
3. Install Docker. **Note:** Docker is a prerequisite only for testing your application locally or using the --use-container option.
4. Install Homebrew.
5. Install the AWS SAM CLI.

Step 1: Create an AWS account

If you don't already have an AWS account, see [aws.amazon.com](http://aws.amazon.com) and choose **Create an AWS Account**. For detailed instructions, see [How do I create and activate a new AWS account?](http://aws.amazon.com)

Step 2: Configure IAM permissions and AWS credentials

The IAM user that you use with AWS SAM must have sufficient permissions to make necessary AWS service calls and manage AWS resources. The simplest way to ensure that a user has sufficient permissions is to grant administrator privileges to them. For more information, see [Creating your first IAM admin user and group](http://docs.aws.amazon.com/IAM/latest/userguide/) in the **IAM User Guide**.

**Note**

If you don't want to grant administrator privileges to users who use the AWS Command Line Interface (AWS CLI), you can grant restricted sets of permissions to them. For more information, see [Permissions](http://docs.aws.amazon.com/IAM/latest/userguide/)

In addition, to enable the AWS SAM CLI to make AWS service calls, you must set up AWS credentials. For more information, see [Setting up AWS credentials](http://docs.aws.amazon.com/).
Step 3: Install Docker (optional)

Note
Docker is a prerequisite only for testing your application locally and for building deployment packages using the `--use-container` option. If you don’t plan to use these features initially, you can skip this section or install Docker at a later time.

Docker is an application that runs containers on your macOS machines. AWS SAM provides a local environment that’s similar to AWS Lambda to use as a Docker container. You can use this container to build, test, and debug your serverless applications.

To run serverless projects and functions locally with the AWS SAM CLI, you must have Docker installed and working. The AWS SAM CLI uses the `DOCKER_HOST` environment variable to contact the Docker daemon. The following steps describe how to install, configure, and verify a Docker installation to work with the AWS SAM CLI.

1. Install Docker

   The AWS SAM CLI supports Docker running on macOS Sierra 10.12 or above. To install Docker see Install Docker Desktop for Mac.

2. Configure your shared drives

   The AWS SAM CLI requires that the project directory, or any parent directory, is listed in a shared drive. To share drives on macOS, see File sharing.

3. Verify the installation

   After Docker is installed, verify that it’s working. Also confirm that you can run Docker commands from the command line (for example, `docker ps`). You don’t need to install, fetch, or pull any containers—the AWS SAM CLI does this automatically as required.

If you run into issues installing Docker, see the Logs and troubleshooting section of the Docker installation guide for additional troubleshooting tips.

Step 4: Install Homebrew

The recommended approach for installing the AWS SAM CLI on macOS is to use the Homebrew package manager. For more information about Homebrew, see Homebrew Documentation.

To install Homebrew, you must first install Git. For more information about Git, see Git Documentation. Git is available on many different operating systems, including macOS. For instructions about installing Git on your particular operating system, see Installing Git.

Once you have successfully installed Git, run the following to install Homebrew, making sure to follow the prompts:

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"
```

Verify that Homebrew is installed:

```
brew --version
```

You should see output like the following on successful installation of Homebrew:

```
Homebrew 2.5.7
```
Step 5: Install the AWS SAM CLI

Follow these steps to install the AWS SAM CLI using Homebrew:

```bash
brew tap aws/tap
brew install aws-sam-cli
```

Verify the installation:

```bash
sam --version
```

You should see output like the following after successful installation of the AWS SAM CLI:

```
SAM CLI, version 1.15.0
```

You're now ready to start development.

Upgrading

To upgrade the AWS SAM CLI, using Homebrew, run the following command:

```bash
brew upgrade aws-sam-cli
```

Uninstalling

To uninstall the AWS SAM CLI, using Homebrew, run the following command:

```bash
brew uninstall aws-sam-cli
```

Next steps

You're now ready to begin building your own serverless applications using AWS SAM! If you want to start with sample serverless applications, choose one of the following links:

- Tutorial: Deploying a Hello World application (p. 14) – Step-by-step instructions to download, build, and deploy a simple serverless application.
- AWS SAM example applications in GitHub – Sample applications in the AWS SAM GitHub repository that you can further experiment with.

Setting up AWS credentials

The AWS SAM command line interface (CLI) requires you to set AWS credentials so that it can make calls to AWS services on your behalf. For example, the AWS SAM CLI makes calls to Amazon S3 and AWS CloudFormation.

You might have already set AWS credentials to work with AWS tools, like one of the AWS SDKs or the AWS CLI. If you haven't, this topic shows you the recommended approaches for setting AWS credentials.
To set AWS credentials, you must have the **access key ID** and your **secret access key** for the IAM user you want to configure. For information about access key IDs and secret access keys, see [Managing Access Keys for IAM Users in the IAM User Guide](https://docs.aws.amazon.com/iam/latest/userguide/iam-create-user.html).

Next, determine whether you have the AWS CLI installed. Then follow the instructions in one of the following sections:

### Using the AWS CLI

If you have the AWS CLI installed, use the `aws configure` command and follow the prompts:

```
$ aws configure
AWS Access Key ID [None]: your_access_key_id
AWS Secret Access Key [None]: your_secret_access_key
Default region name [None]:
Default output format [None]:
```

For information about the `aws configure` command, see [Quickly Configuring the AWS CLI in the AWS Command Line Interface User Guide](https://docs.aws.amazon.com/cli/latest/reference/configure.html).

### Not using the AWS CLI

If you don’t have the AWS CLI installed, you can either create a credentials file or set environment variables:

- **Credentials file** – You can set credentials in the AWS credentials file on your local system. This file must be located in one of the following locations:
  - `~/.aws/credentials` on Linux or macOS
  - `C:\Users\USERNAME\.aws\credentials` on Windows

  This file should contain lines in the following format:

  ```
  [default]
  aws_access_key_id = your_access_key_id
  aws_secret_access_key = your_secret_access_key
  ```

- **Environment variables** – You can set the `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY` environment variables.

  To set these variables on Linux or macOS, use the `export` command:

  ```
  export AWS_ACCESS_KEY_ID=your_access_key_id
  export AWS_SECRET_ACCESS_KEY=your_secret_access_key
  ```

  To set these variables on Windows, use the `set` command:

  ```
  set AWS_ACCESS_KEY_ID=your_access_key_id
  set AWS_SECRET_ACCESS_KEY=your_secret_access_key
  ```

### Tutorial: Deploying a Hello World application

In this guide, you download, build, and deploy a sample Hello World application using AWS SAM. You then test the application in the AWS Cloud, and optionally test it locally on your development host.
This application implements a basic API backend. It consists of an Amazon API Gateway endpoint and an AWS Lambda function. When you send a GET request to the API Gateway endpoint, the Lambda function is invoked. This function returns a *hello world* message.

The following diagram shows the components of this application:

When you initialize your sample application, you have the option to choose a Lambda deployment package type, either **Zip** or **Image**. For more information about package types, see Lambda deployment packages in the AWS Lambda Developer Guide.

The following is a preview of commands that you run to create your Hello World application. For more information about each of these commands, see the sections later in this tutorial.

```
#Step 1 - Download a sample application
sam init

#Step 2 - Build your application
cd sam-app
sam build

#Step 3 - Deploy your application
sam deploy --guided
```

**Prerequisites**

This guide assumes that you’ve completed the steps for your operating system in Installing the AWS SAM CLI (p. 3), including:

1. Creating an AWS account.
2. Configuring AWS Identity and Access Management (IAM) permissions.
3. Installing Docker. **Note:** Docker is a prerequisite only for testing your application locally.
4. Installing Homebrew. **Note:** Homebrew is a prerequisite only for Linux and macOS.
5. Installing the AWS SAM command line interface (CLI). **Note:** Make sure that you have version 1.13.0 or later. Check the version by running the `sam --version` command.
6. If you select the **Image** package type, having an Amazon Elastic Container Registry (Amazon ECR) repository URI to perform a deployment.
Step 1: Download a sample AWS SAM application

Command to run:

```bash
sam init
```

Follow the on-screen prompts. For this tutorial, we recommend that you choose AWS Quick Start Templates, the Zip package type, the runtime of your choice, and the Hello World Example.

Example output:

```
-----------------------
Generating application:
-----------------------
Name: sam-app
Runtime: python3.7
Dependency Manager: pip
Application Template: hello-world
Output Directory: .

Next steps can be found in the README file at ./sam-app/README.md
```

What AWS SAM is doing:

This command creates a directory with the name that you provided as the project name. The contents of the project directory are similar to the following:

```
sam-app/
  ### README.md
  ### events/
  #   ### event.json
  ### hello_world/
  #   ### __init__.py
  #   ### app.py
  #   ### requirements.txt
  ### template.yaml

  ### tests/
  ###    __init__.py
  ###    test_handler.py
```

**Note**

These project directory contents are created when you choose one of the Python runtimes and the Hello World Example.

There are three especially important files:

- `template.yaml`: Contains the AWS SAM template that defines your application's AWS resources.
- `hello_world/app.py`: Contains your actual Lambda handler logic.
- `hello_world/requirements.txt`: Contains any Python dependencies that the application requires, and is used for `sam build`.
Step 2: Build your application

Command to run:

First, change into the project directory, where the template.yaml file for the sample application is located. (By default, this directory is sam-app.) Then run this command:

```
sam build
```

Example output:

```
Build Succeeded
Built Artifacts : .aws-sam/build
Built Template : .aws-sam/build/template.yaml

Commands you can use next
=================================
[*] Invoke Function: sam local invoke
[*] Deploy: sam deploy --guided
```

What AWS SAM is doing:

The AWS SAM CLI comes with abstractions for a number of Lambda runtimes to build your dependencies, and copies the source code into staging folders so that everything is ready to be packaged and deployed. The `sam build` command builds any dependencies that your application has, and copies your application source code to folders under .aws-sam/build to be zipped and uploaded to Lambda.

You can see the following top-level tree under .aws-sam:

```
.aws_sam/
    build/
    HelloWorldFunction/
    template.yaml
```

HelloWorldFunction is a directory that contains your app.py file, as well as third-party dependencies that your application uses.

Step 3: Deploy your application to the AWS Cloud

Command to run:

```
sam deploy --guided
```

Follow the on-screen prompts. To accept the default options provided in the interactive experience, respond with Enter. If you selected the Image package type when you downloaded your sample application, you are prompted for an Amazon ECR repository. To deploy your serverless application, provide a valid Amazon ECR repository URI.

Note

For the prompt HelloWorldFunction may not have authorization defined, Is this okay? [y/N], AWS SAM is informing you that the sample application configures an API
Gateway API without authorization. When you deploy the sample application, AWS SAM creates a publicly available URL.

You can acknowledge this notification by answering "Y" to the prompt. For information about configuring authorization, see Controlling access to API Gateway APIs (p. 186).

**Example output:**

```plaintext
Deploying with following values
================================
Stack name : sam-app
Region     : us-east-1
Confirm changeset : False
Deployment s3 bucket : sam-bucket
Capabilities : ["CAPABILITY_IAM"]
Parameter overrides : {}

Initiating deployment
=====================

Waiting for changeset to be created..

CloudFormation stack changeset
---------------------------------
<table>
<thead>
<tr>
<th>Operation</th>
<th>ResourceType</th>
<th>LogicalResourceId</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ Add</td>
<td>HelloWorldFunction</td>
<td>HelloWorldFunctionProd</td>
</tr>
<tr>
<td>+ Add</td>
<td>AWS::Lambda::Permission</td>
<td>AWS::Lambda::Permission</td>
</tr>
<tr>
<td>+ Add</td>
<td>ServerlessRestApiDeployment47fc2d5f9d</td>
<td>ServerlessRestApiDeployment47fc2d5f9d</td>
</tr>
<tr>
<td>+ Add</td>
<td>AWS::ApiGateway::Deployment</td>
<td>AWS::ApiGateway::Deployment</td>
</tr>
<tr>
<td>+ Add</td>
<td>ServerlessRestApiProdStage</td>
<td>ServerlessRestApiProdStage</td>
</tr>
<tr>
<td>+ Add</td>
<td>AWS::ApiGateway::Stage</td>
<td>AWS::ApiGateway::Stage</td>
</tr>
<tr>
<td>+ Add</td>
<td>ServerlessRestApi</td>
<td>ServerlessRestApi</td>
</tr>
<tr>
<td>* Modify</td>
<td>AWS::IAM::Role</td>
<td>HelloWorldFunctionRole</td>
</tr>
<tr>
<td>* Modify</td>
<td>AWS::Lambda::Function</td>
<td>HelloWorldFunction</td>
</tr>
</tbody>
</table>

2019-11-21 14:33:24 - Waiting for stack create/update to complete

CloudFormation events from changeset
-------------------------------------
<table>
<thead>
<tr>
<th>ResourceType</th>
<th>LogicalResourceId</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPDATE_IN_PROGRESS</td>
<td>AWS::IAM::Role</td>
</tr>
<tr>
<td>HelloWorldFunctionRole</td>
<td>-</td>
</tr>
<tr>
<td>UPDATE_COMPLETE</td>
<td>AWS::IAM::Role</td>
</tr>
<tr>
<td>HelloWorldFunctionRole</td>
<td>-</td>
</tr>
<tr>
<td>UPDATE_IN_PROGRESS</td>
<td>AWS::Lambda::Function</td>
</tr>
<tr>
<td>HelloWorldFunction</td>
<td>-</td>
</tr>
<tr>
<td>UPDATE COMPLETE</td>
<td>AWS::Lambda::Function</td>
</tr>
<tr>
<td>HelloWorldFunction</td>
<td>-</td>
</tr>
<tr>
<td>CREATE IN_PROGRESS</td>
<td>AWS::ApiGateway::RestApi</td>
</tr>
<tr>
<td>ServerlessRestApi</td>
<td>-</td>
</tr>
<tr>
<td>CREATE COMPLETE</td>
<td>AWS::ApiGateway::RestApi</td>
</tr>
<tr>
<td>ServerlessRestApi</td>
<td>-</td>
</tr>
<tr>
<td>CREATE IN_PROGRESS</td>
<td>AWS::ApiGateway::RestApi</td>
</tr>
<tr>
<td>ServerlessRestApi</td>
<td>-</td>
</tr>
<tr>
<td>CREATE IN_PROGRESS</td>
<td>AWS::ApiGateway::Deployment</td>
</tr>
<tr>
<td>ServerlessRestApiDeployment47fc2d5f9d</td>
<td>-</td>
</tr>
</tbody>
</table>
```

---

18
CREATE_IN_PROGRESS         AWS::Lambda::Permission  HelloWorldFunctionHelloWorldPerm
sionProd
CREATE_IN_PROGRESS         AWS::Lambda::Permission  HelloWorldFunctionHelloWorldPerm
CREATE_IN_PROGRESS         AWS::ApiGateway::Deployment  ServerlessRestApiDeployment47fc2d5
CREATE_COMPLETE           AWS::ApiGateway::Deployment  ServerlessRestApiDeployment47fc2d5
CREATE_IN_PROGRESS         AWS::ApiGateway::Stage    ServerlessRestApiProdStage
CREATE_COMPLETE           AWS::ApiGateway::Stage    ServerlessRestApiProdStage
CREATE_COMPLETE           AWS::Lambda::Permission  HelloWorldFunctionHelloWorldPerm
UPDATE_COMPLETE_CLEANUP_IN_PROGRESS  AWS::CloudFormation::Stack  sam-app
UPDATE_COMPLETE           AWS::CloudFormation::Stack  sam-app

Stack sam-app outputs:

<table>
<thead>
<tr>
<th>OutputKey-Description</th>
<th>OutputValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>HelloWorldFunctionIamRole - Implicit IAM Role created for Hello World</td>
<td>arn:aws:iam::123456789012:role/sam-app-function</td>
</tr>
<tr>
<td>HelloWorldFunctionRole-104VTJ0TST7M0</td>
<td>HelloWorldApi - API Gateway endpoint URL for Prod stage for Hello World</td>
</tr>
<tr>
<td><a href="https://0ks2zue0zh.execute-api.us-east-1.amazonaws.com/Prod/hello/">https://0ks2zue0zh.execute-api.us-east-1.amazonaws.com/Prod/hello/</a></td>
<td><a href="https://0ks2zue0zh.execute-api.us-east-1.amazonaws.com/Prod/hello/">https://0ks2zue0zh.execute-api.us-east-1.amazonaws.com/Prod/hello/</a></td>
</tr>
<tr>
<td>HelloWorldFunction - Hello World Lambda Function ARN</td>
<td>arn:aws:lambda:us-east-1:123456789012:function:sam-app-</td>
</tr>
<tr>
<td>HelloWorldFunction-1TY92MJX0BXU5</td>
<td></td>
</tr>
</tbody>
</table>

Successfully created/updated stack - sam-app in us-east-1

What AWS SAM is doing:

This command deploys your application to the AWS Cloud. It takes the deployment artifacts that you build with the `sam build` command, packages and uploads them to an Amazon Simple Storage Service (Amazon S3) bucket that the AWS SAM CLI creates, and deploys the application using AWS CloudFormation. In the output of the `sam deploy` command, you can see the changes being made to your AWS CloudFormation stack.

If your application created an HTTP endpoint, the outputs that `sam deploy` generates also show you the endpoint URL for your test application. You can use `curl` to send a request to your application using that endpoint URL. For example:

```
curl https://<restapiid>.execute-api.us-east-1.amazonaws.com/Prod/hello/
```

After successfully deploying your application, you see output like the following:
Step 4: (Optional) Test your application locally

If you see \{"message": "hello world"\} after executing the curl command, you've successfully deployed your serverless application to AWS, and you're calling your live Lambda function. Otherwise, see the Troubleshooting (p. 22) section later in this tutorial.

Step 4: (Optional) Test your application locally

When you're developing your application, you might find it useful to test locally. The AWS SAM CLI provides the `sam local` command to run your application using Docker containers that simulate the execution environment of Lambda. There are two options to do this:

- Host your API locally
- Invoke your Lambda function directly

This step describes both options.

Host your API locally

Command to run:

```
sam local start-api
```

Example output:

```
2019-07-12 15:27:58 You can now browse to the above endpoints to invoke your functions.
You do not need to restart/reload SAM CLI while working on your functions, changes will be reflected instantly/automatically. You only need to restart SAM CLI if you update your AWS SAM template
START RequestId: 52fdfc07-2182-154f-163f-5f0f9a621d72 Version: $LATEST
END RequestId: 52fdfc07-2182-154f-163f-5f0f9a621d72
REPORT RequestId: 52fdfc07-2182-154f-163f-5f0f9a621d72  Duration: 4.42 ms  Billed Duration: 100 ms  Memory Size: 128 MB  Max Memory Used: 22 MB
2019-07-12 15:28:58 No Content-Type given. Defaulting to 'application/json'.
```

It can take a while for the Docker image to load. After it's loaded, you can use curl to send a request to your application that's running on your local host:

```
curl http://127.0.0.1:3000/hello
```

Example output:
2019-07-12 15:29:57 Invoking app.lambda_handler (python3.7)

Fetching lambci/lambda:python3.7 Docker container image......
HelloWorldFunction as /var/task:ro,delegated inside runtime container
START RequestId: 52fdfc07-2182-154f-163f-5f0f9a621d72 Version: $LATEST
END RequestId: 52fdfc07-2182-154f-163f-5f0f9a621d72
REPORT RequestId: 52fdfc07-2182-154f-163f-5f0f9a621d72 Duration: 7.92 ms Billed
Duration: 100 ms Memory Size: 128 MB Max Memory Used: 22 MB
{"statusCode":200,"body":"{"message": "hello world"}"

What AWS SAM is doing:
The start-api command starts up a local endpoint that replicates your REST API endpoint. It downloads an execution container that you can run your function in locally. The end result is the same output that you saw when you called your function in the AWS Cloud.

Invoke your Lambda function directly

Command to run:
sam local invoke "HelloWorldFunction" -e events/event.json

Example output:

Communication container image downloading
2019-07-01 14:08:42 Found credentials in shared credentials file: ~/.aws/credentials
2019-07-01 14:08:42 Invoking app.lambda_handler (python3.7)

Fetching lambci/lambda:python3.7 Docker container image..................................................................................................................... ...
HelloWorldFunction as /var/task:ro,delegated inside runtime container
START RequestId: 52fdfc07-2182-154f-163f-5f0f9a621d72 Version: $LATEST
END RequestId: 52fdfc07-2182-154f-163f-5f0f9a621d72
REPORT RequestId: 52fdfc07-2182-154f-163f-5f0f9a621d72 Duration: 3.51 ms Billed
Duration: 100 ms Memory Size: 128 MB Max Memory Used: 22 MB
{"statusCode":200,"body": "{"message": "hello world"}"}

What AWS SAM is doing:
The invoke command directly invokes your Lambda functions, and can pass input event payloads that you provide. With this command, you pass the event payload in the file event.json that the sample application provides.

Your initialized application comes with a default aws-proxy event for API Gateway. A number of values are pre-populated for you. In this case, the HelloWorldFunction doesn't care about the particular values, so a stubbed request is OK. You can specify a number of values to substitute in to the request to simulate what you would expect from an actual request. The following is an example of generating your own input event and comparing the output with the default event.json object:
sam local generate-event apigateway aws-proxy --body "" --path "hello" --method GET > api-event.json
diff api-event.json event.json

Example output:
Troubleshooting

AWS SAM CLI error: "Security Constraints Not Satisfied"

When executing `sam deploy --guided`, you are prompted with the question `HelloWorldFunction` may not have authorization defined, Is this okay? [y/N]. If you respond to this prompt with "N" (the default response), you see the following error:

```
Error: Security Constraints Not Satisfied
```

The prompt is informing you that the application you’re about to deploy may have an API Gateway API configured without authorization. By responding "N" to this prompt (the default), you are saying that this is not OK.

To fix this, you have the following options:

- Configure your application with authorization. For information about configuring authorization, see Controlling access to API Gateway APIs (p. 186).
- Respond to this question with "Y" to indicate that you are OK with deploying an application that has an API Gateway API configured without authorization.

AWS SAM CLI error: "no such option: --app-template"

When executing `sam init`, you see the following error:

```
Error: no such option: --app-template
```
This means that you are using an older version of the AWS SAM CLI that does not support the **--app-template** parameter. To fix this, you can either update your version of AWS SAM CLI to 0.33.0 or later, or omit the **--app-template** parameter from the `sam init` command.

**AWS SAM CLI error: "no such option: --guided"**

When executing `sam deploy`, you see the following error:

```
Error: no such option: --guided
```

This means that you are using an older version of the AWS SAM CLI that does not support the **--guided** parameter. To fix this, you can either update your version of AWS SAM CLI to 0.33.0 or later, or omit the **--guided** parameter from the `sam deploy` command.

**AWS SAM CLI error: "Failed to create managed resources: Unable to locate credentials"**

When executing `sam deploy`, you see the following error:

```
Error: Failed to create managed resources: Unable to locate credentials
```

This means that you have not set up AWS credentials to enable the AWS SAM CLI to make AWS service calls. To fix this, you must set up AWS credentials. For more information, see [Setting up AWS credentials](#) (p. 13).

**AWS SAM CLI error: "Running AWS SAM projects locally requires Docker. Have you got it installed?"**

When executing `sam local start-api`, you see the following error:

```
Error: Running AWS SAM projects locally requires Docker. Have you got it installed?
```

This means that you do not have Docker properly installed. Docker is required to test your application locally. To fix this, follow the instructions for installing Docker for your development host. Go to [Installing the AWS SAM CLI](#) (p. 3), choose the appropriate platform, and then follow the instructions in the section titled **Install Docker**.

**Curl error: "Missing Authentication Token"**

When trying to invoke the API Gateway endpoint, you see the following error:

```
{"message":"Missing Authentication Token"}
```

This means that you've attempted to send a request to the correct domain, but the URI isn't recognizable. To fix this, verify the full URL, and update the `curl` command with the correct URL.
Curl error: "curl: (6) Could not resolve: ...

When trying to invoke the API Gateway endpoint, you see the following error:

curl: (6) Could not resolve: endpointdomain (Domain name not found)

This means that you've attempted to send a request to an invalid domain. This can happen if your serverless application failed to deploy successfully, or if you have a typo in your curl command. Verify that the application deployed successfully by using the AWS CloudFormation console or the AWS CLI, and verify that your curl command is correct.

Clean up

If you no longer need the AWS resources that you created by running this tutorial, you can remove them by deleting the AWS CloudFormation stack that you deployed.

To delete the AWS CloudFormation stack using the AWS Management Console, follow these steps:

2. In the left navigation pane, choose Stacks.
3. In the list of stacks, choose sam-app (or the name of the stack that you created).
4. Choose Delete.

When done, the status of the stack changes to DELETE_COMPLETE.

Alternatively, you can delete the AWS CloudFormation stack by running the following AWS CLI command:

```bash
aws cloudformation delete-stack --stack-name sam-app --region region
```

Verify the deleted stack

For both methods of deleting the AWS CloudFormation stack, you can verify that it was deleted by going to the AWS CloudFormation console. In the left navigation pane, choose Stacks, and then in the dropdown list next to the search box, choose Deleted. You should see your stack's name in the list of deleted stacks.

Conclusion

In this tutorial, you've done the following:

1. Created, built, and deployed a serverless application to AWS using AWS SAM.
2. Tested your application locally using the AWS SAM CLI and Docker.
3. Deleted the AWS resources that you no longer need.

Next steps

You're now ready to start building your own applications using the AWS SAM CLI.
To help you get started, you can download any of the example applications from the AWS Serverless Application Repository Examples repository on GitHub.
AWS Serverless Application Model (AWS SAM) specification

You use the AWS SAM specification to define your serverless application. This section provides details for the AWS SAM template sections, resources types, resource properties, data types, resource attributes, intrinsic functions, and API Gateway extensions that you can use in AWS SAM templates.

AWS SAM templates are an extension of AWS CloudFormation templates, with some additional components that make them easier to work with. For the full reference for AWS CloudFormation templates, see AWS CloudFormation Template Reference in the AWS CloudFormation User Guide.

Topics
- AWS SAM template anatomy (p. 26)
- AWS SAM resource and property reference (p. 32)
- Resource attributes (p. 173)
- Intrinsic functions (p. 174)
- Generated AWS CloudFormation resources (p. 174)
- API Gateway extensions (p. 180)

AWS SAM template anatomy

An AWS SAM template file closely follows the format of an AWS CloudFormation template file, which is described in Template anatomy in the AWS CloudFormation User Guide. The primary differences between AWS SAM template files and AWS CloudFormation template files are the following:

- **Transform declaration.** The declaration Transform: AWS::Serverless-2016-10-31 is required for AWS SAM template files. This declaration identifies an AWS CloudFormation template file as an AWS SAM template file. For more information about transforms, see Transform in the AWS CloudFormation User Guide.

- **Globals section.** The Globals section is unique to AWS SAM. It defines properties that are common to all your serverless functions and APIs. All the AWS::Serverless::Function, AWS::Serverless::Api, and AWS::Serverless::SimpleTable resources inherit the properties that are defined in the Globals section. For more information about this section, see Globals section of the AWS SAM template (p. 28).

- **Resources section.** In AWS SAM templates the Resources section can contain a combination of AWS CloudFormation resources and AWS SAM resources. For more information about AWS CloudFormation resources, see AWS resource and property types reference in the AWS CloudFormation User Guide. For more information about AWS SAM resources, see AWS SAM resource and property reference (p. 32).

- **Parameters section.** Objects that are declared in the Parameters section cause the sam deploy --guided command to present additional prompts to the user. For examples of declared objects and the corresponding prompts, see sam deploy (p. 241) in the AWS SAM CLI command reference.

All other sections of an AWS SAM template file correspond to the AWS CloudFormation template file section of the same name.

YAML

The following example shows a YAML-formatted template fragment.
Template sections

AWS SAM templates can include several major sections. Only the Transform and Resources sections are required.

You can include template sections in any order. However, as you build your template, it can be helpful to use the logical order that's shown in the following list. This is because the values in one section might refer to values from a previous section.

Transform (required)

For AWS SAM templates, you must include this section with a value of AWS::Serverless-2016-10-31.

Additional transforms are optional. For more information about transforms, see Transform in the AWS CloudFormation User Guide.

Globals (optional) (p. 28)

Properties that are common to all your serverless functions, APIs, and simple tables. All the AWS::Serverless::Function, AWS::Serverless::Api, and AWS::Serverless::SimpleTable resources inherit the properties that are defined in theGlobals section.

This section is unique to AWS SAM. There isn't a corresponding section in AWS CloudFormation templates.

Description (optional)

A text string that describes the template.

This section corresponds directly with the Description section of AWS CloudFormation templates.

Metadata (optional)

Objects that provide additional information about the template.
This section corresponds directly with the Metadata section of AWS CloudFormation templates.

**Parameters (optional)**

Values to pass to your template at runtime (when you create or update a stack). You can refer to parameters from the Resources and Outputs sections of the template.

Values that are passed in using the `--parameter-overflowes` parameter of the `sam deploy` command—and entries in the configuration file—take precedence over entries in the AWS SAM template file. For more information about the `sam deploy` command, see `sam deploy (p. 241)` in the AWS SAM CLI command reference. For more information about the configuration file, see AWS SAM CLI configuration file (p. 259).

**Mappings (optional)**

A mapping of keys and associated values that you can use to specify conditional parameter values, similar to a lookup table. You can match a key to a corresponding value by using the `Fn::FindInMap` intrinsic function in the Resources and Outputs sections.

This section corresponds directly with the Mappings section of AWS CloudFormation templates.

**Conditions (optional)**

Conditions that control whether certain resources are created or whether certain resource properties are assigned a value during stack creation or update. For example, you could conditionally create a resource that depends on whether the stack is for a production or test environment.

This section corresponds directly with the Conditions section of AWS CloudFormation templates.

**Resources (required)**

The stack resources and their properties, such as an Amazon Elastic Compute Cloud (Amazon EC2) instance or an Amazon Simple Storage Service (Amazon S3) bucket. You can refer to resources in the Resources and Outputs sections of the template.

This section is similar to the Resources section of AWS CloudFormation templates. In AWS SAM templates, this section can contain AWS SAM resources in addition to AWS CloudFormation resources.

**Outputs (optional)**

The values that are returned whenever you view your stack's properties. For example, you can declare an output for an S3 bucket name, and then call the `aws cloudformation describe-stacks` AWS Command Line Interface (AWS CLI) command to view the name.

This section corresponds directly with the Outputs section of AWS CloudFormation templates.

**Next steps**

To download and deploy a sample serverless application that contains an AWS SAM template file, see Getting started with AWS SAM (p. 3) and follow the instructions in Tutorial: Deploying a Hello World application (p. 14).

**Globals section of the AWS SAM template**

Sometimes resources that you declare in an AWS SAM template have common configurations. For example, you might have an application with multiple `AWS::Serverless::Function` resources that have identical Runtime, Memory, VPCConfig, Environment, and Cors configurations. Instead of duplicating this information in every resource, you can declare them once in theGlobals section and let your resources inherit them.
The **Globals** section is supported by the `AWS::Serverless::Function`, `AWS::Serverless::Api`, `AWS::Serverless::HttpApi`, and `AWS::Serverless::SimpleTable` resources.

**Example:**

```yaml
Globals:
  Function:
    Runtime: nodejs12.x
    Timeout: 180
    Handler: index.handler
  Environment:
    Variables:
      TABLE_NAME: data-table

Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      Environment:
        Variables:
          MESSAGE: "Hello From SAM"

  ThumbnailFunction:
    Type: AWS::Serverless::Function
    Properties:
      Events:
        Thumbnail:
          Type: Api
          Properties:
            Path: /thumbnail
            Method: POST
```

In this example, both `HelloWorldFunction` and `ThumbnailFunction` use "nodejs12.x" for Runtime, "180" seconds for Timeout, and "index.handler" for Handler. `HelloWorldFunction` adds the MESSAGE environment variable, in addition to the inherited `TABLE_NAME`. `ThumbnailFunction` inherits all the `Globals` properties and adds an API event source.

**Supported resources and properties**

AWS SAM supports the following resources and properties.
EventInvokeConfig:

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

HttpApi:
Auth:
AccessLogSettings:
StageVariables:
Tags:

SimpleTable:
SSESpecification:

---

**Note**

Any resources and properties that are not included in the previous list are not supported. Some reasons for not supporting them include: 1) They open potential security issues, or 2) They make the template hard to understand.

### Implicit APIs

AWS SAM creates *implicit APIs* when you declare an API in the `Events` section. You can use `Globals` to override all properties of implicit APIs.

### Overridable properties

Resources can override the properties that you declare in the `Globals` section. For example, you can add new variables to an environment variable map, or you can override globally declared variables. But the resource cannot remove a property that's specified in the `Globals` section.

More generally, the `Globals` section declares properties that all your resources share. Some resources can provide new values for globally declared properties, but they can't remove them. If some resources use a property but others don't, then you must not declare them in the `Globals` section.

The following sections describe how overriding works for different data types.

#### Primitive data types are replaced

Primitive data types include strings, numbers, Booleans, and so on.

The value specified in the `Resources` section replaces the value in the `Globals` section.

**Example:**

```text
Globals:
```
Maps are merged

Maps are also known as dictionaries or collections of key-value pairs.

Map entries in the Resources section are merged with global map entries. If there are duplicates, the Resource section entry overrides the Globals section entry.

Example:

```yaml
Globals:
  Function:
    Environment:
      Variables:
        STAGE: Production
        TABLE_NAME: global-table

Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      Environment:
        Variables:
          TABLE_NAME: resource-table
          NEW_VAR: hello

The environment variables of MyFunction are set to the following:

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

Lists are additive

Lists are also known as arrays.

List entries in the Globals section are prepended to the list in the Resources section.

Example:

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

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

The SecurityGroupIds for MyFunction’s VpcConfig are set to the following:

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

AWS SAM resource and property reference

This section contains reference information for the AWS SAM resource and property types.

Topics
- AWS::Serverless::Api (p. 32)
- AWS::Serverless::Application (p. 63)
- AWS::Serverless::Function (p. 67)
- AWS::Serverless::HttpApi (p. 128)
- AWS::Serverless::LayerVersion (p. 147)
- AWS::Serverless::SimpleTable (p. 151)
- AWS::Serverless::StateMachine (p. 154)

AWS::Serverless::Api

Creates a collection of Amazon API Gateway resources and methods that can be invoked through HTTPS endpoints.

An AWS::Serverless::Api (p. 32) resource need not be explicitly added to a AWS Serverless Application Definition template. A resource of this type is implicitly created from the union of Api events defined on AWS::Serverless::Function (p. 67) resources defined in the template that do not refer to an AWS::Serverless::Api (p. 32) resource.

An AWS::Serverless::Api (p. 32) resource should be used to define and document the API using OpenApi, which provides more ability to configure the underlying Amazon API Gateway resources.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

Type: AWS::Serverless::Api
Properties:
  AccessLogSetting: AccessLogSetting
  Auth: ApiAuth (p. 40)
  BinaryMediaTypes: List
  CacheClusterEnabled: Boolean
  CacheClusterSize: String
  CanarySetting: CanarySetting
  Cors: String | CorsConfiguration (p. 57)
Properties

AccessLogSetting

Configures Access Log Setting for a stage.

_Type_: AccessLogSetting

_Required_: No

.AWS CloudFormation compatibility: This property is passed directly to the AccessLogSetting property of an AWS::ApiGateway::Stage resource.

Auth

Configure authorization to control access to your API Gateway API.

For more information about configuring access using AWS SAM see Controlling access to API Gateway APIs (p. 186).

_Type_: ApiAuth (p. 40)

_Required_: No

.AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

BinaryMediaTypes

List of MIME types that your API could return. Use this to enable binary support for APIs. Use ~1 instead of / in the mime types.

_Type_: List

_Required_: No

.AWS CloudFormation compatibility: This property is similar to the BinaryMediaTypes property of an AWS::ApiGateway::RestApi resource. The list of BinaryMediaTypes is added to both the AWS CloudFormation resource and the OpenAPI document.

CacheClusterEnabled

Indicates whether cache clustering is enabled for the stage.

_Type_: Boolean

_Required_: No
AWS CloudFormation compatibility: This property is passed directly to the CacheClusterEnabled property of an AWS::ApiGateway::Stage resource.

CacheClusterSize

The stage’s cache cluster size.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the CacheClusterSize property of an AWS::ApiGateway::Stage resource.

CanarySetting

Configure a canary setting to a stage of a regular deployment.

Type: CanarySetting

Required: No

AWS CloudFormation compatibility: This property is passed directly to the CanarySetting property of an AWS::ApiGateway::Stage resource.

Cors

Manage Cross-origin resource sharing (CORS) for all your API Gateway APIs. Specify the domain to allow as a string or specify a dictionary with additional Cors configuration. NOTE: CORS requires AWS SAM to modify your OpenAPI definition. So, it works only if inline OpenApi is defined with DefinitionBody.

For more information about CORS, see Enable CORS for an API Gateway REST API Resource in the API Gateway Developer Guide.

Type: String | CorsConfiguration (p. 57)

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn’t have an AWS CloudFormation equivalent.

DefinitionBody

OpenAPI specification that describes your API. If neither DefinitionUri nor DefinitionBody are specified, SAM will generate a DefinitionBody for you based on your template configuration.

Type: String

Required: No

AWS CloudFormation compatibility: This property is similar to the Body property of an AWS::ApiGateway::RestApi resource. If certain properties are provided, content may be inserted or modified into the DefinitionBody before being passed to CloudFormation. Properties include Auth, BinaryMediaTypes, Cors, GatewayResponses, Models, and an EventSource of type Api on for a corresponding AWS::Serverless::Function.

DefinitionUri

AWS S3 Uri, local file path, or location object of the the OpenAPI document defining the API. The AWS S3 object this property references must be a valid OpenAPI file. If neither DefinitionUri nor DefinitionBody are specified, SAM will generate a DefinitionBody for you based on your template configuration.
If a local file path is provided, the template must go through the workflow that includes the `sam deploy` or `sam package` command, in order for the definition to be transformed properly.

Intrinsic functions are not supported in external OpenApi files referenced by `DefinitionUri`. Use instead the `DefinitionBody` property with the `Include Transform` to import an OpenApi definition into the template.

**Type:** String | [ApiDefinition (p. 56)]

**Required:** No

**AWS CloudFormation compatibility:** This property is similar to the `BodyS3Location` property of an `AWS::ApiGateway::RestApi` resource. The nested Amazon S3 properties are named differently.

**Description**
A description of the Api resource.

**Type:** String

**Required:** No

**AWS CloudFormation compatibility:** This property is passed directly to the `Description` property of an `AWS::ApiGateway::RestApi` resource.

**Domain**
Configures a custom domain for this API Gateway API.

**Type:** [DomainConfiguration (p. 58)]

**Required:** No

**AWS CloudFormation compatibility:** This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**EndpointConfiguration**
The endpoint type of a REST API.

**Type:** [EndpointConfiguration (p. 62)]

**Required:** No

**AWS CloudFormation compatibility:** This property is similar to the `EndpointConfiguration` property of an `AWS::ApiGateway::RestApi` resource. The nested configuration properties are named differently.

**GatewayResponses**
Configures Gateway Responses for an API. Gateway Responses are responses returned by API Gateway, either directly or through the use of Lambda Authorizers. For more information, see the documentation for the [Api Gateway OpenApi extension for Gateway Responses](#).

**Type:** Map

**Required:** No

**AWS CloudFormation compatibility:** This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**MethodSettings**
Configures all settings for API stage including Logging, Metrics, CacheTTL, Throttling.
Type: MethodSettings

Required: No

AWS CloudFormation compatibility: This property is passed directly to the MethodSettings property of an AWS::ApiGateway::Stage resource.

MinimumCompressionSize

Allow compression of response bodies based on client's Accept-Encoding header. Compression is triggered when response body size is greater than or equal to your configured threshold. The maximum body size threshold is 10 MB (10,485,760 Bytes). - The following compression types are supported: gzip, deflate, and identity.

Type: Integer

Required: No

AWS CloudFormation compatibility: This property is passed directly to the MinimumCompressionSize property of an AWS::ApiGateway::RestApi resource.

Models

The schemas to be used by your API methods. These schemas can be described using JSON or YAML. See the Examples section at the bottom of this page for example models.

Type: Map

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

Name

A name for the API Gateway RestApi resource

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the Name property of an AWS::ApiGateway::RestApi resource.

OpenApiVersion

Version of OpenApi to use. This can either be 2.0 for the Swagger specification, or one of the OpenApi 3.0 versions, like 3.0.1. For more information about OpenAPI, see the OpenAPI Specification.

Note: Setting this property to any valid value will also remove the stage Stage that SAM creates.

Type: String

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

StageName

The name of the stage, which API Gateway uses as the first path segment in the invoke Uniform Resource Identifier (URI).
To reference the stage resource, use `<api-logical-id>.Stage`. For more information about referencing resources generated when an AWS::Serverless::Api resource is specified, see AWS CloudFormation resources generated when AWS::Serverless::Api is specified (p. 176). For general information about generated AWS CloudFormation resources, see Generated AWS CloudFormation resources (p. 174).

**Type:** String

**Required:** Yes

**AWS CloudFormation compatibility:** This property is similar to the `StageName` property of an AWS::ApiGateway::Stage resource. It is required in SAM, but not required in API Gateway.

**Additional notes:** The Implicit API has a stage name of "Prod".

**Tags**

A map (string to string) that specifies the tags to be added to this API Gateway stage. Keys and values are limited to alphanumeric characters. Keys can be 1 to 127 Unicode characters in length and cannot be prefixed with `aws:`. Values can be 1 to 255 Unicode characters in length.

**Type:** Map

**Required:** No

**AWS CloudFormation compatibility:** This property is similar to the `Tags` property of an AWS::ApiGateway::Stage resource. The Tags property in SAM consists of Key:Value pairs; in CloudFormation it consists of a list of Tag objects.

**TracingEnabled**

Indicates whether active tracing with X-Ray is enabled for the stage. For more information about X-Ray, see Tracing user requests to REST APIs using X-Ray in the API Gateway Developer Guide.

**Type:** Boolean

**Required:** No

**AWS CloudFormation compatibility:** This property is passed directly to the `TracingEnabled` property of an AWS::ApiGateway::Stage resource.

**Variables**

A map (string to string) that defines the stage variables, where the variable name is the key and the variable value is the value. Variable names are limited to alphanumeric characters. Values must match the following regular expression: `[A-Za-z0-9._~:/?#&=,-]+`.

**Type:** Map

**Required:** No

**AWS CloudFormation compatibility:** This property is passed directly to the `Variables` property of an AWS::ApiGateway::Stage resource.

**Return Values**

**Ref**

When the logical ID of this resource is provided to the Ref intrinsic function, it returns the ID of the underlying API Gateway API.
For more information about using the `Ref` function, see `Ref` in the *AWS CloudFormation User Guide*.

**Fn::GetAtt**

`Fn::GetAtt` returns a value for a specified attribute of this type. The following are the available attributes and sample return values.

For more information about using `Fn::GetAtt`, see `Fn::GetAtt` in the *AWS CloudFormation User Guide*.

**RootResourceId**

The root resource ID for a `RestApi` resource, such as `a0bc123d4e`.

**Examples**

**SimpleApiExample**

A Hello World AWS SAM template file that contains a Lambda Function with an API endpoint. This is a full AWS SAM template file for a working serverless application.

**YAML**

```yaml
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: AWS SAM template with a simple API definition
Resources:
  ApiGatewayApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: prod
  ApiFunction: # Adds a GET api endpoint at "/" to the ApiGatewayApi via an Api event
    Type: AWS::Serverless::Function
    Properties:
      Events:
        ApiEvent:
          Type: Api
          Properties:
            Path: /
            Method: get
            RestApiId:
              Ref: ApiGatewayApi
          Runtime: python3.7
          Handler: index.handler
          InlineCode: |
            def handler(event, context):
              return {'body': 'Hello World!', 'statusCode': 200}
```

**ApiCorsExample**

An AWS SAM template snippet with an API defined in an external Swagger file along with Lambda integrations and CORS configurations. This is just a portion of an AWS SAM template file showing an `AWS::Serverless::Api` definition.

**YAML**

```yaml
Resources:
  ApiGatewayApi:
```
Type: AWS::Serverless::Api
Properties:
  StageName: Prod
  # Allows www.example.com to call these APIs
  # SAM will automatically add AllowMethods with a list of methods for this API
  Cors: "www.example.com"
  DefinitionBody: # Pull in an OpenApi definition from S3
    'Fn::Transform':
      Name: 'AWS::Include'
      # Replace "bucket" with your bucket name
      Parameters:
        Location: s3://bucket/swagger.yaml

ApiCognitoAuthExample

An AWS SAM template snippet with an API that uses AWS Cognito to authorize requests against the API. This is just a portion of an AWS SAM template file showing an AWS::Serverless::Api definition.

YAML

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

ApiModelsExample

An AWS SAM template snippet with an API that includes a Models schema. This is just a portion of an AWS SAM template file, showing an AWS::Serverless::Api definition with two model schemas.

YAML

Resources:
  ApiGatewayApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      Models:
        User:
          type: object
          required:
            - username
            - employee_id
          properties:
            username:
              type: string
            employee_id:
              type: integer
            department:
              type: string
        Item:
          type: object
          properties:
count:
  type: integer
category:
  type: string
price:
  type: integer

### ApiAuth

Configure authorization to control access to your API Gateway API.

For more information and examples for configuring access using AWS SAM see Controlling access to API Gateway APIs (p. 186).

### Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

**YAML**

```yaml
AddDefaultAuthorizerToCorsPreflight: Boolean
ApiKeyRequired: Boolean
Authorizers: CognitoAuthorizer (p. 44) | LambdaTokenAuthorizer (p. 50) | LambdaRequestAuthorizer (p. 47)
DefaultAuthorizer: String
InvokeRole: String
UsagePlan: ApiUsagePlan (p. 42)
```

### Properties

**AddDefaultAuthorizerToCorsPreflight**

If the DefaultAuthorizer and Cors properties are set, then setting AddDefaultAuthorizerToCorsPreflight will cause the default authorizer to be added to the Options property in the OpenAPI section.

*Type:* Boolean

*Required:* No

*Default:* True

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**ApiKeyRequired**

If set to true then an API key is required for all API events. For more information about API keys see Create and Use Usage Plans with API Keys in the API Gateway Developer Guide.

*Type:* Boolean

*Required:* No

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.
Authorizers

The authorizer used to control access to your API Gateway API.

For more information, see Controlling access to API Gateway APIs (p. 186).

Type: CognitoAuthorizer (p. 44) | LambdaTokenAuthorizer (p. 50) | LambdaRequestAuthorizer (p. 47)

Required: No
Default: None

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn’t have an AWS CloudFormation equivalent.

Additional notes: SAM adds the Authorizers to the OpenApi definition of an Api.

DefaultAuthorizer

Specify a default authorizer for an API Gateway API, which will be used for authorizing API calls by default.

Note: If the Api EventSource for the function associated with this API is configured to use IAM Permissions, then this property must be set to AWS_IAM, otherwise an error will result.

Type: String

Required: No
Default: None

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn’t have an AWS CloudFormation equivalent.

InvokeRole

Sets integration credentials for all resources and methods to this value.

CALLER_CREDENTIALS maps to arn:aws:iam::*:user/*, which uses the caller credentials to invoke the endpoint.

Valid values: CALLER_CREDENTIALS, NONE, IAMRoleArn

Type: String

Required: No
Default: CALLER_CREDENTIALS

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn’t have an AWS CloudFormation equivalent.

ResourcePolicy

Configure Resource Policy for all methods and paths on an API.

Type: ResourcePolicyStatement (p. 53)

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn’t have an AWS CloudFormation equivalent.
Additional notes: This setting can also be defined on individual AWS::Serverless::Function using the ApiFunctionAuth (p. 89). This is required for APIs with EndpointConfiguration: PRIVATE.

UsagePlan

Configures a usage plan associated with this API. For more information about usage plans see Create and Use Usage Plans with API Keys in the API Gateway Developer Guide.

This AWS SAM property generates three additional AWS CloudFormation resources when this property is set: an AWS::ApiGateway::UsagePlan, an AWS::ApiGateway::UsagePlanKey, and an AWS::ApiGateway::ApiKey. For information about this scenario, see UsagePlan property is specified (p. 177). For general information about generated AWS CloudFormation resources, see Generated AWS CloudFormation resources (p. 174).

Type: ApiUsagePlan (p. 42)

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

Examples

CognitoAuth

Cognito Auth Example

YAML

```yaml
Auth:
  Authorizers:
    MyCognitoAuth:
      UserPoolArn:
        Fn::GetAtt:
        - MyUserPool
        - Arn
      AuthType: "COGNITO_USER_POOLS"
  DefaultAuthorizer: MyCognitoAuth
  InvokeRole: CALLER_CREDENTIALS
  AddDefaultAuthorizerToCorsPreflight: false
  ApiKeyRequired: false
  ResourcePolicy:
    CustomStatements: [{
      "Effect": "Allow",
      "Principal": "*",
      "Action": "execute-api:Invoke",
      "Resource": "execute-api:/Prod/GET/pets",
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": "1.2.3.4"
        }
      }
    }]
  IpRangeBlacklist:
    - "10.20.30.40"
```

ApiUsagePlan

Configures a usage plan for an API Gateway API. For more information about usage plans, see Create and Use Usage Plans with API Keys in the API Gateway Developer Guide.
Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

**YAML**

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

**Properties**

CreateUsagePlan

Determines how this usage plan is configured. Valid values are PER_API, SHARED, and NONE.

PER_API creates AWS::ApiGateway::UsagePlan, AWS::ApiGateway::ApiKey, and AWS::ApiGateway::UsagePlanKey resources that are specific to this API. These resources have logical IDs of <api-logical-id>UsagePlan, <api-logical-id>ApiKey, and <api-logical-id>UsagePlanKey, respectively.

SHARED creates AWS::ApiGateway::UsagePlan, AWS::ApiGateway::ApiKey, and AWS::ApiGateway::UsagePlanKey resources that are shared across any API that also has CreateUsagePlan: SHARED in the same AWS SAM template. These resources have logical IDs of ServerlessUsagePlan, ServerlessApiKey, and ServerlessUsagePlanKey, respectively. If you use this option, we recommend that you add additional configuration for this usage plan on only one API resource to avoid conflicting definitions and an uncertain state.

NONE disables the creation or association of a usage plan with this API. This is only necessary if SHARED or PER_API is specified in the Globals section of the AWS SAM template (p. 28).

**Valid values:** PER_API, SHARED, and NONE

**Type:** String

**Required:** Yes

**AWS CloudFormation compatibility:** This property is unique to AWS SAM and doesn’t have an AWS CloudFormation equivalent.

Description

A description of the usage plan.

**Type:** String

**Required:** No

**AWS CloudFormation compatibility:** This property is passed directly to the Description property of an AWS::ApiGateway::UsagePlan resource.

Quota

Configures the number of requests that users can make within a given interval.

**Type:** QuotaSettings

**Required:** No

**Required:** No
AWS CloudFormation compatibility: This property is passed directly to the Quota property of an AWS::ApiGateway::UsagePlan resource.

Tags
An array of arbitrary tags (key-value pairs) to associate with the usage plan.
This property uses the CloudFormation Tag Type.

Type: List

Required: No

AWS CloudFormation compatibility: This property is passed directly to the Tags property of an AWS::ApiGateway::UsagePlan resource.

Throttle
Configures the overall request rate (average requests per second) and burst capacity.

Type: ThrottleSettings

Required: No

AWS CloudFormation compatibility: This property is passed directly to the Throttle property of an AWS::ApiGateway::UsagePlan resource.

UsagePlanName
A name for the usage plan.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the UsagePlanName property of an AWS::ApiGateway::UsagePlan resource.

Examples

UsagePlan
The following is a usage plan example.

YAML

Auth:
UsagePlan:
  CreateUsagePlan: PER_API
  Description: Usage plan for this API
  Quota:
    Limit: 500
    Period: MONTH
    Throttle:
      BurstLimit: 100
      RateLimit: 50
  Tags:
    - Key: TagName
      Value: TagValue

CognitoAuthorizer

Define a Amazon Cognito User Pool authorizer.
For more information and examples, see Controlling access to API Gateway APIs (p. 186).

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

**YAML**

```yaml
AuthorizationScopes: List
Identity: CognitoAuthorizationIdentity (p. 46)
UserPoolArn: String
```

**Properties**

**AuthorizationScopes**

List of authorization scopes for this authorizer.

- **Type**: List
- **Required**: No

*AWS CloudFormation compatibility*: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**Identity**

This property can be used to specify an IdentitySource in an incoming request for an authorizer

- **Type**: CognitoAuthorizationIdentity (p. 46)
- **Required**: No

*AWS CloudFormation compatibility*: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**UserPoolArn**

Can refer to a user pool/specify a userpool arn to which you want to add this cognito authorizer

- **Type**: String
- **Required**: Yes

*AWS CloudFormation compatibility*: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**Examples**

**CognitoAuth**

Cognito Auth Example

**YAML**

```yaml
Auth:
  Authorizers:
    MyCognitoAuth:
      AuthorizationScopes:
```
CognitoAuthorizationIdentity

This property can be used to specify an IdentitySource in an incoming request for an authorizer. For more information about IdentitySource see the ApiGateway Authorizer OpenApi extension.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

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

Properties

Header

- Specify the header name for Authorization in the OpenApi definition.
  - Type: String
  - Required: No
  - Default: Authorization

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

ReauthorizeEvery

- The time-to-live (TTL) period, in seconds, that specifies how long API Gateway caches authorizer results. If you specify a value greater than 0, API Gateway caches the authorizer responses. By default, API Gateway sets this property to 300. The maximum value is 3600, or 1 hour.
  - Type: Integer
  - Required: No
  - Default: 300

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

ValidationExpression

- Specify a validation expression for validating the incoming Identity
  - Type: String
**AWS::Serverless::Api**

*Required:* No

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

### Examples

**CognitoAuthIdentity**

**YAML**

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

**LambdaRequestAuthorizer**

Configure a Lambda Authorizer to control access to your API with a Lambda function.

For more information and examples, see Controlling access to API Gateway APIs (p. 186).

**Syntax**

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

**YAML**

```yaml
AuthorizationScopes: List
FunctionArn: String
FunctionInvokeRole: String
FunctionPayloadType: String
Identity: LambdaRequestAuthorizationIdentity (p. 48)
```

**Properties**

**AuthorizationScopes**

List of authorization scopes for this authorizer.

*Type:* List

*Required:* No

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**FunctionArn**

Specify the function arn of the Lambda function which provides authorization for the API.

*Type:* String

*Required:* Yes

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.
FunctionInvokeRole

Adds authorizer credentials to the OpenApi definition of the Lambda authorizer.

*Type:* String

*Required:* No

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

FunctionPayloadType

This property can be used to define the type of Lambda Authorizer for an API.

*Valid values:* TOKEN or REQUEST

*Type:* String

*Required:* No

*Default:* TOKEN

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

Identity

This property can be used to specify an IdentitySource in an incoming request for an authorizer

*Type:* LambdaRequestAuthorizationIdentity (p. 48)

*Required:* No

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**Examples**

**LambdaRequestAuth**

**YAML**

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

**LambdaRequestAuthorizationIdentity**

This property can be used to specify an IdentitySource in an incoming request for an authorizer. For more information about IdentitySource see the ApiGateway Authorizer OpenApi extension.
Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

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

Properties

Context

Converts the given context strings to the mapping expressions of format `context.contextString`.

Type: List

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

Headers

Converts the headers to comma-separated string of mapping expressions of format `method.request.header.name`.

Type: List

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

QueryStrings

Converts the given query strings to comma-separated string of mapping expressions of format `method.request.querystring.queryString`.

Type: List

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

ReauthorizeEvery

The time-to-live (TTL) period, in seconds, that specifies how long API Gateway caches authorizer results. If you specify a value greater than 0, API Gateway caches the authorizer responses. By default, API Gateway sets this property to 300. The maximum value is 3600, or 1 hour.

Type: Integer

Required: No

Default: 300
AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn’t have an AWS CloudFormation equivalent.

StageVariables

Converts the given stage variables to comma-separated string of mapping expressions of format stageVariables.stageVariable.

Type: List

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn’t have an AWS CloudFormation equivalent.

Examples

LambdaRequestIdentity

YAML

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

LambdaTokenAuthorizer

Configure a Lambda Authorizer to control access to your API with a Lambda function.

For more information and examples, see Controlling access to API Gateway APIs (p. 186).

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

```yaml
AuthorizationScopes: List
FunctionArn: String
FunctionInvokeRole: String
FunctionPayloadType: String
Identity: LambdaTokenAuthorizationIdentity (p. 52)
```

Properties

AuthorizationScopes

List of authorization scopes for this authorizer.

Type: List

Required: No
AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

FunctionArn

Specify the function arn of the Lambda function which provides authorization for the API.

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

FunctionInvokeRole

Adds authorizer credentials to the OpenApi definition of the Lambda authorizer.

Type: String

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

FunctionPayloadType

This property can be used to define the type of Lambda Authorizer for an Api.

Valid values: TOKEN or REQUEST

Type: String

Required: No

Default: TOKEN

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

Identity

This property can be used to specify an IdentitySource in an incoming request for an authorizer.

Type: LambdaTokenAuthorizationIdentity (p. 52)

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

Examples

LambdaTokenAuth

YAML

```
Authorizers:
  MyLambdaTokenAuth:
    FunctionArn:
      Fn::GetAtt:
        - MyAuthFunction
```
- Arn
  Identity:
    Header: MyCustomAuthHeader # OPTIONAL; Default: 'Authorization'
    ValidationExpression: mycustomauthexpression # OPTIONAL
    ReauthorizeEvery: 20 # OPTIONAL; Service Default: 300

BasicLambdaTokenAuth

YAML

Authorizers:
  MyLambdaTokenAuth:
    FunctionArn:
      Fn::GetAtt:
      - MyAuthFunction
      - Arn

LambdaTokenAuthorizationIdentity

This property can be used to specify an IdentitySource in an incoming request for an authorizer. For more information about IdentitySource see the ApiGateway Authorizer OpenApi extension.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

| ReauthorizeEvery: Integer | ValidationExpression: String |

Properties

ReauthorizeEvery

The time-to-live (TTL) period, in seconds, that specifies how long API Gateway caches authorizer results. If you specify a value greater than 0, API Gateway caches the authorizer responses. By default, API Gateway sets this property to 300. The maximum value is 3600, or 1 hour.

Type: Integer

Required: No

Default: 300

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

ValidationExpression

Specify a validation expression for validating the incoming Identity.

Type: String

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.
Examples

LambdaTokenIdentity

YAML

```yaml
Identity:
  Header: Auth
  ValidationExpression: Bearer.*
  ReauthorizeEvery: 30
```

ResourcePolicyStatement

Configures a resource policy for all methods and paths of an API. For more information about resource policies, see Controlling access to an API with API Gateway resource policies in the API Gateway Developer Guide.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

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

Properties

AwsAccountBlacklist

The AWS accounts to block.

*Type*: List

*Required*: No

*AWS CloudFormation compatibility*: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

AwsAccountWhitelist

The AWS accounts to allow. For an example use of this property, see the Examples section at the bottom of this page.

*Type*: List

*Required*: No

*AWS CloudFormation compatibility*: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.
CustomStatements

A list of custom resource policy statements to apply to this API. For an example use of this property, see the Examples section at the bottom of this page.

_Type: List_

_Required: No_

_AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent._

IntrinsicVpcBlacklist

The list of virtual private clouds (VPCs) to block, where each VPC is specified as a reference such as a _dynamic reference_ or the _Ref_ _intrinsic function_. For an example use of this property, see the Examples section at the bottom of this page.

_Type: List_

_Required: No_

_AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent._

IntrinsicVpcWhitelist

The list of VPCs to allow, where each VPC is specified as a reference such as a _dynamic reference_ or the _Ref_ _intrinsic function_.

_Type: List_

_Required: No_

_AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent._

IntrinsicVpceBlacklist

The list of VPC endpoints to block, where each VPC endpoint is specified as a reference such as a _dynamic reference_ or the _Ref_ _intrinsic function_.

_Type: List_

_Required: No_

_AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent._

IntrinsicVpceWhitelist

The list of VPC endpoints to allow, where each VPC endpoint is specified as a reference such as a _dynamic reference_ or the _Ref_ _intrinsic function_. For an example use of this property, see the Examples section at the bottom of this page.

_Type: List_

_Required: No_

_AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent._

IpRangeBlacklist

The IP addresses or address ranges to block. For an example use of this property, see the Examples section at the bottom of this page.
Type: List

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

IpRangeWhitelist

The IP addresses or address ranges to allow.

Type: List

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

SourceVpcBlacklist

The source VPC or VPC endpoints to block. Source VPC names must start with "vpc-" and source VPC endpoint names must start with "vpce-". For an example use of this property, see the Examples section at the bottom of this page.

Type: List

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

SourceVpcWhitelist

The source VPC or VPC endpoints to allow. Source VPC names must start with "vpc-" and source VPC endpoint names must start with "vpce-".

Type: List

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

Examples

Resource Policy Example

The following example blocks two IP addresses and a source VPC, and allows an AWS account.

YAML

```yaml
Auth:
  ResourcePolicy:
    CustomStatements: [{
      "Effect": "Allow",
      "Principal": "*",
      "Action": "execute-api:Invoke",
      "Resource": "execute-api:/Prod/GET/pets",
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": "1.2.3.4"
        }
      }
    }
```
IpRangeBlacklist:
  - "10.20.30.40"
  - "1.2.3.4"
SourceVpcBlacklist:
  - "vpce-1a2b3c4d"
AwsAccountWhitelist:
  - "111122223333"
IntrinsicVpcBlacklist:
  - "{{resolve:ssm:SomeVPCReference:1}}"
  - !Ref MyVPC
IntrinsicVpceWhitelist:
  - "{{resolve:ssm:SomeVPCEReference:1}}"
  - !Ref MyVPCE

## ApiDefinition

An OpenAPI document defining the API.

### Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

**YAML**

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

### Properties

**Bucket**

The name of the Amazon S3 bucket where the OpenAPI file is stored.

*Type: String*

*Required: Yes*

*AWS CloudFormation compatibility:* This property is passed directly to the `Bucket` property of the `AWS::ApiGateway::RestApi S3Location` data type.

**Key**

The Amazon S3 key of the OpenAPI file.

*Type: String*

*Required: Yes*

*AWS CloudFormation compatibility:* This property is passed directly to the `Key` property of the `AWS::ApiGateway::RestApi S3Location` data type.

**Version**

For versioned objects, the version of the OpenAPI file.

*Type: String*

*Required: No*
AWS CloudFormation compatibility: This property is passed directly to the `Version` property of the `AWS::ApiGateway::RestApi` S3Location data type.

**Examples**

**Definition Uri example**

API Definition example

**YAML**

```yaml
DefinitionUri:
  Bucket: mybucket-name
  Key: mykey-name
  Version: 121212
```

**CorsConfiguration**

Manage cross-origin resource sharing (CORS) for your API Gateway APIs. Specify the domain to allow as a string or specify a dictionary with additional Cors configuration. NOTE: Cors requires SAM to modify your OpenAPI definition, so it only works with inline OpenApi defined in the `DefinitionBody` property.

For more information about CORS, see Enable CORS for an API Gateway REST API Resource in the API Gateway Developer Guide.

Note: If CorsConfiguration is set both in OpenAPI and at the property level, AWS SAM merges them, with the properties taking precedence.

**Syntax**

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

**YAML**

```yaml
AllowCredentials: Boolean
AllowHeaders: String
AllowMethods: String
AllowOrigin: String
MaxAge: String
```

**Properties**

**AllowCredentials**

Boolean indicating whether request is allowed to contain credentials.

_Type_: Boolean

_Required_: No

/aws CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**AllowHeaders**

String of headers to allow.

_Type_: String
**AWS::Serverless::Api**

**Required:** No

**AWS CloudFormation compatibility:** This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**AllowMethods**

String containing the HTTP methods to allow.

**Type:** String

**Required:** No

**AWS CloudFormation compatibility:** This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**AllowOrigin**

String of origin to allow.

**Type:** String

**Required:** Yes

**AWS CloudFormation compatibility:** This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**MaxAge**

String containing the number of seconds to cache CORS Preflight request.

**Type:** String

**Required:** No

**AWS CloudFormation compatibility:** This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**Examples**

**CorsConfiguration**

Cors Configuration example. This is just a portion of an AWS SAM template file showing an AWS::Serverless::Api definition with Cors configured.

**YAML**

```yaml
Resources:
  ApiGatewayApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      Cors:
        AllowMethods: "'POST, GET'"
        AllowHeaders: "'X-Forwarded-For'"
        AllowOrigin: "'www.example.com'"
        MaxAge: "'600'"
        AllowCredentials: True
```

**DomainConfiguration**

Configures a custom domain for an API.
Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

**YAML**

```yaml
BasePath: List
CertificateArn: String
DomainName: String
EndpointConfiguration: String
MutualTlsAuthentication: MutualTlsAuthentication
Route53: Route53Configuration (p. 61)
SecurityPolicy: String
```

**Properties**

**BasePath**

A list of the basepaths to configure with the Amazon API Gateway domain name.

*Type*: List

*Required*: No

*Default*: /

*AWS CloudFormation compatibility*: This property is similar to the `BasePath` property of an AWS::ApiGateway::BasePathMapping resource. AWS SAM creates multiple AWS::ApiGateway::BasePathMapping resources, one per `BasePath` specified in this property.

**CertificateArn**

The Amazon Resource Name (ARN) of an AWS managed certificate this domain name's endpoint. AWS Certificate Manager is the only supported source.

*Type*: String

*Required*: Yes

*AWS CloudFormation compatibility*: This property is similar to the `CertificateArn` property of an AWS::ApiGateway::DomainName resource. If `EndpointConfiguration` is set to `REGIONAL` (the default value), `CertificateArn` maps to `RegionalCertificateArn` in AWS::ApiGateway::DomainName. If the `EndpointConfiguration` is set to `EDGE`, `CertificateArn` maps to `CertificateArn` in AWS::ApiGateway::DomainName.

*Additional notes*: For an `EDGE` endpoint, you must create the certificate in the `us-east-1` AWS Region.

**DomainName**

The custom domain name for your API Gateway API. Uppercase letters are not supported.

AWS SAM generates an AWS::ApiGateway::DomainName resource when this property is set. For information about this scenario, see DomainName property is specified (p. 177). For information about generated AWS CloudFormation resources, see Generated AWS CloudFormation resources (p. 174).

*Type*: String

*Required*: Yes
AWS CloudFormation compatibility: This property is passed directly to the DomainName property of an AWS::ApiGateway::DomainName resource.

EndpointConfiguration

Defines the type of API Gateway endpoint to map to the custom domain. The value of this property determines how the CertificateArn property is mapped in AWS CloudFormation.

**Valid values:** REGIONAL or EDGE

**Type:** String

**Required:** No

**Default:** REGIONAL

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

MutualTlsAuthentication

The mutual Transport Layer Security (TLS) authentication configuration for a custom domain name.

**Type:** MutualTlsAuthentication

**Required:** No

AWS CloudFormation compatibility: This property is passed directly to the MutualTlsAuthentication property of an AWS::ApiGateway::DomainName resource.

Route53

Defines an Amazon Route 53 configuration.

**Type:** Route53Configuration (p. 61)

**Required:** No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

SecurityPolicy

The TLS version plus cipher suite for this domain name.

**Type:** String

**Required:** No

AWS CloudFormation compatibility: This property is passed directly to the SecurityPolicy property of an AWS::ApiGateway::DomainName resource.

Examples

DomainName

DomainName example

**YAML**

```
Domain:
  DomainName: www.example.com
  CertificateArn: arn-example
  EndpointConfiguration: EDGE
```
Route53Configuration

Configures the Route53 record sets for an API.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

**YAML**

```yaml
DistributionDomainName: String
EvaluateTargetHealth: Boolean
HostedZoneId: String
HostedZoneName: String
IpV6: Boolean
```

Properties

**DistributionDomainName**

Configures a custom distribution of the API custom domain name.

*Type: String*

*Required: No*

*Default: Use the API Gateway distribution.*

*AWS CloudFormation compatibility: This property is passed directly to the `DNSName` property of an `AWS::Route53::RecordSetGroup AliasTarget` resource.*

*Additional notes: The domain name of a CloudFront distribution.*

**EvaluateTargetHealth**

When EvaluateTargetHealth is true, an alias record inherits the health of the referenced AWS resource, such as an Elastic Load Balancing load balancer or another record in the hosted zone.

*Type: Boolean*

*Required: No*

*AWS CloudFormation compatibility: This property is passed directly to the `EvaluateTargetHealth` property of an `AWS::Route53::RecordSetGroup AliasTarget` resource.*

*Additional notes: You can't set EvaluateTargetHealth to true when the alias target is a CloudFront distribution.*

**HostedZoneId**

The ID of the hosted zone that you want to create records in.

Specify either `HostedZoneName` or `HostedZoneId`, but not both. If you have multiple hosted zones with the same domain name, you must specify the hosted zone using `HostedZoneId`.

*Type: String*
Required: No

AWS CloudFormation compatibility: This property is passed directly to the HostedZoneId property of an AWS::Route53::RecordSetGroup RecordSet resource.

HostedZoneName

The name of the hosted zone that you want to create records in.

Specify either HostedZoneName or HostedZoneId, but not both. If you have multiple hosted zones with the same domain name, you must specify the hosted zone using HostedZoneId.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the HostedZoneName property of an AWS::Route53::RecordSetGroup RecordSet resource.

IpV6

When this property is set, AWS SAM creates a AWS::Route53::RecordSet resource and sets Type to AAAA for the provided HostedZone.

Type: Boolean

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

Examples

Route 53 Configuration Example

This example shows how to configure Route 53.

YAML

<table>
<thead>
<tr>
<th>Domain:</th>
</tr>
</thead>
<tbody>
<tr>
<td>DomainName: <a href="http://www.example.com">www.example.com</a></td>
</tr>
<tr>
<td>CertificateArn: arn-example</td>
</tr>
<tr>
<td>EndpointConfiguration: EDGE</td>
</tr>
<tr>
<td>Route53:</td>
</tr>
<tr>
<td>HostedZoneId: Z1PA6795UKNMB9</td>
</tr>
<tr>
<td>EvaluateTargetHealth: true</td>
</tr>
<tr>
<td>DistributionDomainName: xyz</td>
</tr>
</tbody>
</table>

EndpointConfiguration

The endpoint type of a REST API.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

| Type: String |
| VPC EndpointIds: List |
Properties

Type

The endpoint type of a REST API.

Valid values: EDGE or REGIONAL or PRIVATE

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the Types property of the AWS::ApiGateway::RestApi EndpointConfiguration data type.

VPCEndpointIds

A list of VPC endpoint IDs of a REST API against which to create Route53 aliases.

Type: List

Required: No

AWS CloudFormation compatibility: This property is passed directly to the VpcEndpointIds property of the AWS::ApiGateway::RestApi EndpointConfiguration data type.

Examples

EndpointConfiguration

Endpoint Configuration example

YAML

```
EndpointConfiguration:
  Type: EDGE
  VPCEndpointIds:
    - vpce-123a123a
    - vpce-321a321a
```

AWS::Serverless::Application

Embeds a serverless application from the AWS Serverless Application Repository or from an Amazon S3 bucket as a nested application. Nested applications are deployed as nested AWS::CloudFormation::Stack resources, which can contain multiple other resources including other AWS::Serverless::Application (p. 63) resources.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

```
Type: AWS::Serverless::Application
Properties:
  Location: String | ApplicationLocationObject (p. 66)
  NotificationARNs: List
```
Parameters: Map
Tags: Map
TimeoutInMinutes: Integer

Properties

Location

Template URL, file path, or location object of a nested application.

If a template URL is provided, it must follow the format specified in the CloudFormation TemplateUrl documentation and contain a valid CloudFormation or SAM template. An ApplicationLocationObject (p. 66) can be used to specify an application that has been published to the AWS Serverless Application Repository.

If a local file path is provided, the template must go through the workflow that includes the `sam deploy` or `sam package` command, in order for the application to be transformed properly.

Type: String | ApplicationLocationObject (p. 66)

Required: Yes

AWS CloudFormation compatibility: This property is similar to the TemplateURL property of an AWS::CloudFormation::Stack resource. The CloudFormation version does not take an ApplicationLocationObject (p. 66) to retrieve an application from the AWS Serverless Application Repository.

NotificationARNs

A list of existing Amazon SNS topics where notifications about stack events are sent.

Type: List

Required: No

AWS CloudFormation compatibility: This property is passed directly to the NotificationARNs property of an AWS::CloudFormation::Stack resource.

Parameters

Application parameter values.

Type: Map

Required: No

AWS CloudFormation compatibility: This property is passed directly to the Parameters property of an AWS::CloudFormation::Stack resource.

Tags

A map (string to string) that specifies the tags to be added to this application. Keys and values are limited to alphanumeric characters. Keys can be 1 to 127 Unicode characters in length and cannot be prefixed with aws:. Values can be 1 to 255 Unicode characters in length.

Type: Map

Required: No

AWS CloudFormation compatibility: This property is similar to the Tags property of an AWS::CloudFormation::Stack resource. The Tags property in SAM consists of Key:Value pairs; in CloudFormation it consists of a list of Tag objects. When the stack is created, SAM will automatically add a lambda:createdBy:SAM tag to this application. In addition,
if this application is from the AWS Serverless Application Repository, then SAM will also automatically the two additional tags serverlessrepo:applicationId:ApplicationId and serverlessrepo:semanticVersion:SemanticVersion.

TimeoutInMinutes

The length of time, in minutes, that AWS CloudFormation waits for the nested stack to reach the CREATE_COMPLETE state. The default is no timeout. When AWS CloudFormation detects that the nested stack has reached the CREATE_COMPLETE state, it marks the nested stack resource as CREATE_COMPLETE in the parent stack and resumes creating the parent stack. If the timeout period expires before the nested stack reaches CREATE_COMPLETE, AWS CloudFormation marks the nested stack as failed and rolls back both the nested stack and parent stack.

Type: Integer

Required: No

AWS CloudFormation compatibility: This property is passed directly to the TimeoutInMinutes property of an AWS::CloudFormation::Stack resource.

Return Values

Ref

When the logical ID of this resource is provided to the Ref intrinsic function, it returns the resource name of the underlying AWS::CloudFormation::Stack resource.

For more information about using the Ref function, see Ref in the AWS CloudFormation User Guide.

Fn::GetAtt

Fn::GetAtt returns a value for a specified attribute of this type. The following are the available attributes and sample return values.

For more information about using Fn::GetAtt, see Fn::GetAtt in the AWS CloudFormation User Guide.

Outputs.ApplicationOutputName

The value of the stack output with name ApplicationOutputName.

Examples

SAR Application

Application that uses a template from the Serverless Application Repository

YAML

```yaml
Type: AWS::Serverless::Application
Properties:
  Location:
    ApplicationId: 'arn:aws:serverlessrepo:us-east-1:1012345678901:applications/my-application'
    SemanticVersion: 1.0.0
  Parameters:
    StringParameter: parameter-value
    IntegerParameter: 2
```
Normal-Application
Application from an S3 url

YAML

```yaml
Type: AWS::Serverless::Application
Properties:
  Location: https://s3.amazonaws.com/demo-bucket/template.yaml
```

ApplicationLocationObject
An application that has been published to the AWS Serverless Application Repository.

Syntax
To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

```yaml
ApplicationId: String
SemanticVersion: String
```

Properties

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

Type: String
Required: Yes

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

SemanticVersion
The semantic version of the application.

Type: String
Required: Yes

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

Examples

my-application
Example application location object

YAML

```yaml
Location:
  ApplicationId: 'arn:aws:serverlessrepo:us-east-1:012345678901:applications/my-application'
```
SemanticVersion: 1.0.0

## AWS::Serverless::Function

Creates an AWS Lambda function, an AWS Identity and Access Management (IAM) execution role, and event source mappings that trigger the function.

The AWS::Serverless::Function (p. 67) resource also supports the Metadata resource attribute, so you can instruct AWS SAM to build custom runtimes that your application requires. For more information about building custom runtimes, see Building custom runtimes (p. 204).

### Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

### YAML

```
Type: AWS::Serverless::Function
Properties:
  AssumeRolePolicyDocument: JSON
  AutoPublishAlias: String
  AutoPublishCodeSha256: String
  CodeSigningConfigArn: String
  CodeUri: String | FunctionCode (p. 127)
  DeadLetterQueue: Map | DeadLetterQueue (p. 77)
  DeploymentPreference: DeploymentPreference (p. 78)
  Description: String
  Environment: Environment
  EventInvokeConfig: EventInvokeConfiguration (p. 80)
  Events: EventSource (p. 85)
  FileSystemConfig: List
  FunctionName: String
  Handler: String
  ImageConfig: ImageConfig
  ImageUri: String
  InlineCode: String
  KmsKeyArn: String
  Layers: List
  MemorySize: Integer
  PackageType: String
  PermissionsBoundary: String
  Policies: String | List | Map
  ProvisionedConcurrencyConfig: ProvisionedConcurrencyConfig
  ReservedConcurrentExecutions: Integer
  Role: String
  Runtime: String
  Tags: Map
  Timeout: Integer
  Tracing: String
  VersionDescription: String
  VpcConfig: VpcConfig
```

### Properties

**AssumeRolePolicyDocument**

Adds an AssumeRolePolicyDocument for the default created Role for this function. If this property isn't specified, AWS SAM adds a default assume role for this function.
Type: JSON

Required: No

**AWS CloudFormation compatibility:** This property is similar to the `AssumeRolePolicyDocument` property of an `AWS::IAM::Role` resource. AWS SAM adds this property to the generated IAM role for this function. If a role's Amazon Resource Name (ARN) is provided for this function, this property does nothing.

**AutoPublishAlias**

The name of the Lambda alias. For more information about Lambda aliases, see Lambda function aliases in the AWS Lambda Developer Guide. For examples that use this property, see Deploying serverless applications gradually (p. 220).

AWS SAM generates `AWS::Lambda::Version` and `AWS::Lambda::Alias` resources when this property is set. For information about this scenario, see AutoPublishAlias property is specified (p. 178). For general information about generated AWS CloudFormation resources, see Generated AWS CloudFormation resources (p. 174).

Type: String

Required: No

**AWS CloudFormation compatibility:** This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**AutoPublishCodeSha256**

The string value that is used, along with the value in `CodeUri`, to determine whether a new Lambda version should be published.

This property addresses a problem that occurs when an AWS SAM template has the following characteristics: the `DeploymentPreference` object is configured for gradual deployments (as described in Deploying serverless applications gradually (p. 220)), the `AutoPublishAlias` property is set and doesn't change between deployments, and the `CodeUri` property is set and doesn't change between deployments.

This scenario can occur when the deployment package stored in an Amazon Simple Storage Service (Amazon S3) location is replaced by a new deployment package that contains updated Lambda function code, but the `CodeUri` property remains unchanged (as opposed to the new deployment package being uploaded to a new Amazon S3 location and the `CodeUri` being changed to the new location).

In this scenario, to trigger the gradual deployment successfully, you must provide a unique value for `AutoPublishCodeSha256`.

Type: String

Required: No

**AWS CloudFormation compatibility:** This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**CodeSigningConfigArn**

The ARN of the `AWS::Lambda::CodeSigningConfig` resource, used to enable code signing for this function. For more information about code signing, see Configuring code signing for AWS SAM applications (p. 196).

Type: String
Required: No

AWS CloudFormation compatibility: This property is passed directly to the CodeSigningConfigArn property of an AWS::Lambda::Function resource.

CodeUri

The function code's Amazon S3 URI, local file path, or FunctionCode (p. 127) object. This property only applies if the PackageType property is set to Zip, otherwise it is ignored.

Notes:
1. If the PackageType property is set to Zip (default), then one of CodeUri or InlineCode is required.
2. If an Amazon S3 URI or FunctionCode (p. 127) object is provided, the Amazon S3 object referenced must be a valid Lambda deployment package.
3. If a local file path is provided, for the code to be transformed properly the template must go through the workflow that includes the sam deploy or sam package command.

Type: String | FunctionCode (p. 127)

Required: Conditional

AWS CloudFormation compatibility: This property is similar to the Code property of an AWS::Lambda::Function resource. The nested Amazon S3 properties are named differently.

DeadLetterQueue

Configures an Amazon Simple Notification Service (Amazon SNS) topic or Amazon Simple Queue Service (Amazon SQS) queue where Lambda sends events that it can't process. For more information about dead-letter queue functionality, see AWS Lambda function dead letter queues in the AWS Lambda Developer Guide.

Type: Map | DeadLetterQueue (p. 77)

Required: No

AWS CloudFormation compatibility: This property is similar to the DeadLetterConfig property of an AWS::Lambda::Function resource. In AWS CloudFormation the type is derived from the TargetArn, whereas in AWS SAM you must pass the type along with the TargetArn.

DeploymentPreference

The settings to enable gradual Lambda deployments.

If a DeploymentPreference object is specified, AWS SAM creates an AWS::CodeDeploy::Application called ServerlessDeploymentApplication (one per stack), an AWS::CodeDeploy::DeploymentGroup called <function-logical-id>DeploymentGroup, and an AWS::IAM::Role called CodeDeployServiceRole.

Type: DeploymentPreference (p. 78)

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

See also: For more information about this property, see Deploying serverless applications gradually (p. 220).
Description

A description of the function.

*Type:* String

*Required:* No

*AWS CloudFormation compatibility:* This property is passed directly to the `Description` property of an `AWS::Lambda::Function` resource.

Environment

The configuration for the runtime environment.

*Type:* Environment

*Required:* No

*AWS CloudFormation compatibility:* This property is passed directly to the `Environment` property of an `AWS::Lambda::Function` resource.

EventInvokeConfig

The object that describes event invoke configuration on a Lambda function.

*Type:* `EventInvokeConfiguration` (p. 80)

*Required:* No

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

Events

Specifies the events that trigger this function. Events consist of a type and a set of properties that depend on the type.

*Type:* `EventSource` (p. 85)

*Required:* No

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

FileSystemConfigs

List of `FileSystemConfig` objects that specify the connection settings for an Amazon Elastic File System (Amazon EFS) file system.

If your template contains an `AWS::EFS::MountTarget` resource, you must also specify a `DependsOn` resource attribute to ensure that the mount target is created or updated before the function.

*Type:* List

*Required:* No

*AWS CloudFormation compatibility:* This property is passed directly to the `FileSystemConfigs` property of an `AWS::Lambda::Function` resource.

FunctionName

A name for the function. If you don't specify a name, a unique name is generated for you.
**Type**: String

**Required**: No

*AWS CloudFormation compatibility*: This property is passed directly to the `FunctionName` property of an `AWS::Lambda::Function` resource.

**Handler**

The function within your code that is called to begin execution. This property is only required if the `PackageType` property is set to `Zip`.

**Type**: String

**Required**: Conditional

*AWS CloudFormation compatibility*: This property is passed directly to the `Handler` property of an `AWS::Lambda::Function` resource.

**ImageConfig**

The object used to configure Lambda container image settings. For more information, see Using container images with Lambda in the *AWS Lambda Developer Guide*.

**Type**: `ImageConfig`

**Required**: No

*AWS CloudFormation compatibility*: This property is passed directly to the `ImageConfig` property of an `AWS::Lambda::Function` resource.

**ImageUri**

The URI of the Amazon Elastic Container Registry (Amazon ECR) repository for the Lambda function’s container image. This property only applies if the `PackageType` property is set to `Image`, otherwise it is ignored. For more information, see Using container images with Lambda in the *AWS Lambda Developer Guide*.

**Note**: If the `PackageType` property is set to `Image`, then either `ImageUri` is required, or you must or you must build your application with necessary `Metadata` entries in the AWS SAM template file. For more information, see Building applications (p. 199).

**Type**: String

**Required**: No

*AWS CloudFormation compatibility*: This property is passed directly to the `ImageUri` property of the `AWS::Lambda::Function Code` data type.

**InlineCode**

The Lambda function code that is written directly in the template. This property only applies if the `PackageType` property is set to `Zip`, otherwise it is ignored.

**Note**: If the `PackageType` property is set to `Zip` (default), then one of `CodeUri` or `InlineCode` is required.

**Type**: String

**Required**: Conditional

*AWS CloudFormation compatibility*: This property is passed directly to the `ZipFile` property of the `AWS::Lambda::Function Code` data type.
KmsKeyArn

The ARN of an AWS Key Management Service (AWS KMS) key that Lambda uses to encrypt and decrypt your function's environment variables.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the KmsKeyArn property of an AWS::Lambda::Function resource.

Layers

The list of LayerVersion ARNs that this function should use. The order specified here is the order in which they will be imported when running the Lambda function.

Type: List

Required: No

AWS CloudFormation compatibility: This property is passed directly to the Layers property of an AWS::Lambda::Function resource.

MemorySize

The size of the memory in MB allocated per invocation of the function.

Type: Integer

Required: No

AWS CloudFormation compatibility: This property is passed directly to the MemorySize property of an AWS::Lambda::Function resource.

PackageType

The deployment package type of the Lambda function. For more information, see Lambda deployment packages in the AWS Lambda Developer Guide.

Notes:

1. If this property is set to Zip (default), then either CodeUri or InlineCode applies, and ImageUri is ignored.
2. If this property is set to Image, then only ImageUri applies, and both CodeUri and InlineCode are ignored.

Valid values: Zip or Image

Type: String

Required: No

Default: Zip

AWS CloudFormation compatibility: This property is passed directly to the PackageType property of an AWS::Lambda::Function resource.

PermissionsBoundary

The ARN of a permissions boundary to use for this function's execution role. This property works only if the role is generated for you.
Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the PermissionsBoundary property of an AWS::IAM::Role resource.

Policies

One or more policies that this function needs. They will be appended to the default role for this function.

This property accepts a single string or a list of strings, and can be the name of AWS managed policies or AWS SAM policy templates, or inline IAM policy documents formatted in YAML.

For more information about AWS managed policies, see AWS managed policies in the IAM User Guide. For more information about AWS SAM policy templates, see AWS SAM policy templates (p. 261) in the AWS Serverless Application Model Developer Guide. For more information about inline policies, see Inline policies in the IAM User Guide.

Note: If the Role property is set, this property is ignored.

Type: String | List | Map

Required: No

AWS CloudFormation compatibility: This property is similar to the Policies property of an AWS::IAM::Role resource. AWS SAM supports AWS managed policy names and AWS SAM policy templates, in addition to JSON policy documents. AWS CloudFormation accepts only JSON policy documents.

ProvisionedConcurrencyConfig

The provisioned concurrency configuration of a function’s alias.

Note: ProvisionedConcurrencyConfig can be specified only if the AutoPublishAlias is set. Otherwise, an error results.

Type: ProvisionedConcurrencyConfig

Required: No

AWS CloudFormation compatibility: This property is passed directly to the ProvisionedConcurrencyConfig property of an AWS::Lambda::Alias resource.

ReservedConcurrentExecutions

The maximum number of concurrent executions that you want to reserve for the function.

For more information about this property, see AWS Lambda Function Scaling in the AWS Lambda Developer Guide.

Type: Integer

Required: No

AWS CloudFormation compatibility: This property is passed directly to the ReservedConcurrentExecutions property of an AWS::Lambda::Function resource.

Role

The ARN of an IAM role to use as this function’s execution role.
**Type:** String

**Required:** No

**AWS CloudFormation compatibility:** This property is similar to the `Role` property of an AWS::Lambda::Function resource. This is required in AWS CloudFormation but not in AWS SAM. If a role isn't specified, one is created for you with a logical ID of `<function-logical-id>Role`.

**Runtime**

The identifier of the function's runtime. This property is only required if the PackageType property is set to `Zip`.

**Note:** If you specify the provided identifier for this property, you can use the Metadata resource attribute to instruct AWS SAM to build the custom runtime that this function requires. For more information about building custom runtimes, see Building custom runtimes (p. 204).

**Type:** String

**Required:** Conditional

**AWS CloudFormation compatibility:** This property is passed directly to the `Runtime` property of an AWS::Lambda::Function resource.

**Tags**

A map (string to string) that specifies the tags added to the Lambda function and the corresponding execution role. Keys and values are limited to alphanumeric characters. Keys can be 1 to 127 Unicode characters in length and cannot be prefixed with `aws:`. Values can be 1 to 255 Unicode characters in length.

**Type:** Map

**Required:** No

**AWS CloudFormation compatibility:** This property is similar to the `Tags` property of an AWS::Lambda::Function resource. The Tags property in AWS SAM consists of key-value pairs. In AWS CloudFormation it consists of a list of `Tag` objects. When the stack is created, AWS SAM automatically adds a `lambda:createdBy:SAM` tag to this Lambda function and the corresponding execution role.

**Timeout**

The maximum time in seconds that the function can run before it is stopped.

**Type:** Integer

**Required:** No

**Default:** 3

**AWS CloudFormation compatibility:** This property is passed directly to the `Timeout` property of an AWS::Lambda::Function resource.

**Tracing**

The string that specifies the function's X-Ray tracing mode. For more information about X-Ray, see Using AWS Lambda with AWS X-Ray in the AWS Lambda Developer Guide.

**Valid values:** Active or PassThrough

**Type:** String
**Required:** No

**AWS CloudFormation compatibility:** This property is similar to the `TracingConfig` property of an `AWS::Lambda::Function` resource. If the `Tracing` property is set to `Active` and the `Role` property is not specified, then AWS SAM adds the `arn:aws:iam::aws:policy/AWSXrayWriteOnlyAccess` policy to the Lambda execution role that it creates for you.

**VersionDescription**

Specifies the `Description` field that is added on the new Lambda version resource.

**Type:** String

**Required:** No

**AWS CloudFormation compatibility:** This property is passed directly to the `Description` property of an `AWS::Lambda::Version` resource.

**VpcConfig**

The configuration that enables this function to access private resources within your virtual private cloud (VPC).

**Type:** VpcConfig

**Required:** No

**AWS CloudFormation compatibility:** This property is passed directly to the `VpcConfig` property of an `AWS::Lambda::Function` resource.

**Return Values**

**Ref**

When the logical ID of this resource is provided to the `Ref` intrinsic function, it returns the resource name of the underlying Lambda function.

For more information about using the `Ref` function, see `Ref` in the *AWS CloudFormation User Guide*.

**Fn::GetAtt**

`Fn::GetAtt` returns a value for a specified attribute of this type. The following are the available attributes and sample return values.

For more information about using `Fn::GetAtt`, see `Fn::GetAtt` in the *AWS CloudFormation User Guide*.

**Arn**

The ARN of the underlying Lambda function.

**Examples**

**Simple function**

The following is a basic example of an `AWS::Serverless::Function (p. 67)` resource of package type `zip` (default) and function code in an Amazon S3 bucket.
YAML

```yaml
Type: AWS::Serverless::Function
Properties:
  Handler: index.handler
  Runtime: python3.6
  CodeUri: s3://bucket-name/key-name
```

Function properties example

The following is an example of an `AWS::Serverless::Function` (p. 67) of package type `Zip` (default) that uses `InlineCode`, `Layers`, `Tracing`, `Policies`, `Amazon EFS`, and an `Api` event source.

YAML

```yaml
Type: AWS::Serverless::Function
DependsOn: MyMountTarget        # This is needed if an AWS::EFS::MountTarget resource is declared for EFS
Properties:
  Handler: index.handler
  Runtime: python3.6
  InlineCode: |
    def handler(event, context):
      print("Hello, world!")
  ReservedConcurrentExecutions: 30
  Layers:
    - Ref: MyLayer
  Tracing: Active
  Timeout: 120
  FileSystemConfigs:
    - Arn: !Ref MyEfsFileSystem
      LocalMountPath: /mnt/EFS
  Policies:
    - AWSLambdaExecute
      Version: '2012-10-17'
      Statement:
        - Effect: Allow
          Action:
            - s3:GetObject
            - s3:GetObjectACL
          Resource: 'arn:aws:s3:::my-bucket/*'
  Events:
    ApiEvent:
      Type: Api
      Properties:
        Path: /path
        Method: get
```

ImageConfig example

The following is an example of an `ImageConfig` for a Lambda function of package type `Image`.

YAML

```yaml
HelloWorldFunction:
  Type: AWS::Serverless::Function
  Properties:
    PackageType: Image
    ImageUri: account-id.dkr.ecr.region.amazonaws.com/ecr-repo-name:image-name
    ImageConfig:
      Command: ["app.lambda_handler"]
```
### EntryPoint

- "entrypoint1"

### WorkingDirectory

"workDir"

---

### DeadLetterQueue

Specifies an SQS queue or SNS topic that AWS Lambda (Lambda) sends events to when it can’t process them. For more information about dead letter queue functionality, see [AWS Lambda Function Dead Letter Queues](#).

SAM will automatically add appropriate permission to your Lambda function execution role to give Lambda service access to the resource. sqs:SendMessage will be added for SQS queues and sns:Publish for SNS topics.

#### Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

**YAML**

```yaml
general:
  DeadLetterQueue:
    TargetArn: String
    Type: String
```

#### Properties

**TargetArn**

The Amazon Resource Name (ARN) of an Amazon SQS queue or Amazon SNS topic.

*Type: String*

*Required: Yes*

*AWS CloudFormation compatibility:* This property is passed directly to the `TargetArn` property of the `AWS::Lambda::Function` DeadLetterConfig data type.

**Type**

The type of dead letter queue.

*Valid values: SNS, SQS*

*Type: String*

*Required: Yes*

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn’t have an AWS CloudFormation equivalent.

#### Examples

**DeadLetterQueue**

Dead Letter Queue example for an SNS topic.

**YAML**

```yaml
deadLetterQueue:
```

---
**DeploymentPreference**

Specifies the configurations to enable gradual Lambda deployments. For more information about configuring gradual Lambda deployments, see Deploying serverless applications gradually (p. 220).

**Note:** You must specify an AutoPublishAlias in your AWS::Serverless::Function (p. 67) to use a DeploymentPreference object, otherwise an error will result.

**Syntax**

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

**YAML**

```
Alarms: List
Enabled: Boolean
Hooks: Hooks (p. 79)
Role: String
TriggerConfigurations: List
Type: String
```

**Properties**

**Alarms**

A list of CloudWatch alarms that you want to be triggered by any errors raised by the deployment.

*Type: List*

*Required: No*

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**Enabled**

Whether this deployment preference is enabled.

*Type: Boolean*

*Required: No*

*Default: True*

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**Hooks**

Validation Lambda functions that are run before and after traffic shifting.

*Type: Hooks (p. 79)*

*Required: No*

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.
Role

An IAM role ARN that CodeDeploy will use for traffic shifting. An IAM role will not be created if this is provided.

_Type_: String

_Required_: No

_AWS CloudFormation compatibility_: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**TriggerConfigurations**

A list of trigger configurations you want to associate with the deployment group. Used to notify an SNS topic on lifecycle events.

_Type_: List

_Required_: No

_AWS CloudFormation compatibility_: This property is passed directly to the `TriggerConfigurations` property of an `AWS::CodeDeploy::DeploymentGroup` resource.

**Type**

There are two categories of deployment types at the moment: Linear and Canary. For more information about available deployment types see _Deploying serverless applications gradually_ (p. 220).

_Type_: String

_Required_: Yes

_AWS CloudFormation compatibility_: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**Examples**

**DeploymentPreference**

Example deployment preference

**YAML**

```
DeploymentPreference:
  Enabled: True
Type: Canary10Percent10Minutes
Alarms:
  - Ref: AliasErrorMetricGreaterThanZeroAlarm
  - Ref: LatestVersionErrorMetricGreaterThanZeroAlarm
Hooks:
  PreTraffic:
    Ref: PreTrafficLambdaFunction
  PostTraffic:
    Ref: PostTrafficLambdaFunction
```

**Hooks**

Validation Lambda functions that are run before and after traffic shifting.
**Note**: The Lambda functions referenced in this property configure the `CodeDeployLambdaAliasUpdate` object of the resulting `AWS::Lambda::Alias` resource. For more information, see `CodeDeployLambdaAliasUpdate Policy` in the *AWS CloudFormation User Guide*.

**Syntax**

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

**YAML**

```yaml
PostTraffic: String
PreTraffic: String
```

**Properties**

**PostTraffic**

Lambda function that is run after traffic shifting.

*Type*: String

*Required*: No

*AWS CloudFormation compatibility*: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**PreTraffic**

Lambda function that is run before traffic shifting.

*Type*: String

*Required*: No

*AWS CloudFormation compatibility*: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**Examples**

**Hooks**

Example hook functions

**YAML**

```yaml
Hooks:
  PreTraffic:
    Ref: PreTrafficLambdaFunction
  PostTraffic:
    Ref: PostTrafficLambdaFunction
```

**EventInvokeConfiguration**

Configuration options for asynchronous Lambda Alias or Version invocations.

**Syntax**

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.
YAML

```yaml
DestinationConfig: EventInvokeDestinationConfiguration (p. 82)
MaximumEventAgeInSeconds: Integer
MaximumRetryAttempts: Integer
```

Properties

DestinationConfig

A configuration object that specifies the destination of an event after Lambda processes it.

*Type:* EventInvokeDestinationConfiguration (p. 82)

*Required:* No

.AWS CloudFormation compatibility:* This property is similar to the DestinationConfig property of an AWS::Lambda::EventInvokeConfig resource. SAM requires an extra parameter, "Type", that does not exist in CloudFormation.

MaximumEventAgeInSeconds

The maximum age of a request that Lambda sends to a function for processing.

*Type:* Integer

*Required:* No

.AWS CloudFormation compatibility:* This property is passed directly to the MaximumEventAgeInSeconds property of an AWS::Lambda::EventInvokeConfig resource.

MaximumRetryAttempts

The maximum number of times to retry before the function returns an error.

*Type:* Integer

*Required:* No

.AWS CloudFormation compatibility:* This property is passed directly to the MaximumRetryAttempts property of an AWS::Lambda::EventInvokeConfig resource.

Examples

MaximumEventAgeInSeconds

MaximumEventAgeInSeconds example

YAML

```yaml
EventInvokeConfig:
  MaximumEventAgeInSeconds: 60
  MaximumRetryAttempts: 2
  DestinationConfig:
    OnSuccess:
      Type: SQS
    OnFailure:
      Type: Lambda
      Destination: !GetAtt DestinationLambda.Arn
```
EventInvokeDestinationConfiguration

A configuration object that specifies the destination of an event after Lambda processes it.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

```
OnFailure: OnFailure (p. 82)
OnSuccess: OnSuccess (p. 84)
```

Properties

OnFailure

A destination for events that failed processing.

Type: OnFailure (p. 82)

Required: No

AWS CloudFormation compatibility: This property is similar to the OnFailure property of an AWS::Lambda::EventInvokeConfig resource. Requires Type, an additional SAM-only property.

OnSuccess

A destination for events that were processed successfully.

Type: OnSuccess (p. 84)

Required: No

AWS CloudFormation compatibility: This property is similar to the OnSuccess property of an AWS::Lambda::EventInvokeConfig resource. Requires Type, an additional SAM-only property.

Examples

OnSuccess

OnSuccess example

YAML

```
EventInvokeConfig:
  DestinationConfig:
    OnSuccess:
      Type: SQS
    OnFailure:
      Type: Lambda
      Destination: !GetAtt DestinationLambda.Arn
```

OnFailure

A destination for events that failed processing.
Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

**YAML**

```
Destination: String
Type: String
```

Properties

**Destination**

The Amazon Resource Name (ARN) of the destination resource.

*Type: String*

*Required: Conditional*

**AWS CloudFormation compatibility:** This property is similar to the `OnFailure` property of an `AWS::Lambda::EventInvokeConfig` resource. SAM will add any necessary permissions to the auto-generated IAM Role associated with this function to access the resource referenced in this property.

**Additional notes:** If the type is Lambda/EventBridge, Destination is required.

**Type**

Type of the resource referenced in the destination. Supported types are SQS, SNS, Lambda, and EventBridge.

*Type: String*

*Required: No*

**AWS CloudFormation compatibility:** This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**Additional notes:** If the type is SQS/SNS and the Destination property is left blank, then the SQS/SNS resource is auto generated by SAM. To reference the resource, use `<function-logical-id>.DestinationQueue` for SQS or `<function-logical-id>.DestinationTopic` for SNS. If the type is Lambda/EventBridge, Destination is required.

Examples

**EventInvoke Configuration Example with SQS and Lambda destinations**

In this example no Destination is given for the SQS OnSuccess configuration, so SAM implicitly creates a SQS queue and adds any necessary permissions. Also for this example, a Destination for a Lambda resource declared in the template file is specified in the OnFailure configuration, so SAM adds the necessary permissions to this Lambda function to call the destination Lambda function.

**YAML**

```
EventInvokeConfig:
  DestinationConfig:
    OnSuccess:
      Type: SQS
    OnFailure:
```


EventInvoke Configuration Example with SNS destination

In this example a Destination is given for an SNS topic declared in the template file for the OnSuccess configuration.

YAML

```
EventInvokeConfig:
  DestinationConfig:
    OnSuccess:
      Type: SNS
      Destination:
        Ref: DestinationSNS       # Arn of an SNS topic declared in the template file
```

OnSuccess

A destination for events that were processed successfully.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

```
Destination: String
Type: String
```

Properties

Destination

The Amazon Resource Name (ARN) of the destination resource.

_Type: String

Required: Conditional

_AWS CloudFormation compatibility:_ This property is similar to the OnSuccess property of an AWS::Lambda::EventInvokeConfig resource. SAM will add any necessary permissions to the auto-generated IAM Role associated with this function to access the resource referenced in this property.

_Additional notes:_ If the type is Lambda/EventBridge, Destination is required.

Type

Type of the resource referenced in the destination. Supported types are SQS, SNS, Lambda, and EventBridge.

_Type: String

Required: No

_AWS CloudFormation compatibility:_ This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.
Additional notes: If the type is SQS/SNS and the Destination property is left blank, then the SQS/SNS resource is auto generated by SAM. To reference the resource, use `<function-logical-id>.DestinationQueue` for SQS or `<function-logical-id>.DestinationTopic` for SNS. If the type is Lambda/EventBridge, Destination is required.

Examples

EventInvoke Configuration Example with SQS and Lambda destinations

In this example no Destination is given for the SQS OnSuccess configuration, so SAM implicitly creates a SQS queue and adds any necessary permissions. Also for this example, a Destination for a Lambda resource declared in the template file is specified in the OnFailure configuration, so SAM adds the necessary permissions to this Lambda function to call the destination Lambda function.

YAML

```
EventInvokeConfig:
  DestinationConfig:
    OnSuccess:
      Type: SQS
    OnFailure:
      Type: Lambda
```

EventInvoke Configuration Example with SNS destination

In this example a Destination is given for an SNS topic declared in the template file for the OnSuccess configuration.

YAML

```
EventInvokeConfig:
  DestinationConfig:
    OnSuccess:
      Type: SNS
    OnFailure:
      Destination:
        Ref: DestinationSNS       # Arn of an SNS topic declared in the template file
```

EventSource

The object describing the source of events which trigger the function. Each event consists of a type and a set of properties that depend on that type. For more information about the properties of each event source, see the topic corresponding to that type.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

```
Properties:
  S3 (p. 118) | SNS (p. 123) | Kinesis (p. 111) | DynamoDB (p. 99)
  | SQS (p. 126) | Api (p. 87) | Schedule (p. 119) | CloudWatchEvent (p. 95)
  | EventBridgeRule (p. 102) | CloudWatchLogs (p. 97) | IoTRule (p. 110)
  | AlexaSkill (p. 86) | Cognito (p. 98) | HttpApi (p. 106) | MSK (p. 116) | MQ (p. 114)
Type: String
```
Properties

Properties

Object describing properties of this event mapping. The set of properties must conform to the defined Type.

**Type:** S3 (p. 118) | SNS (p. 123) | Kinesis (p. 111) | DynamoDB (p. 99) | SQS (p. 126) | Api (p. 87) | Schedule (p. 119) | CloudWatchEvent (p. 95) | EventBridgeRule (p. 102) | CloudWatchLogs (p. 97) | IoTRule (p. 110) | AlexaSkill (p. 86) | Cognito (p. 98) | HttpApi (p. 106) | MSK (p. 116) | MQ (p. 114)

**Required:** Yes

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn’t have an AWS CloudFormation equivalent.

**Type**

The event type.

*Valid values:* S3, SNS, Kinesis, DynamoDB, SQS, Api, Schedule, CloudWatchEvent, CloudWatchLogs, IoTRule, AlexaSkill, Cognito, EventBridgeRule, HttpApi, MSK, MQ

**Type:** String

**Required:** Yes

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn’t have an AWS CloudFormation equivalent.

**Examples**

**APIEvent**

Example of using an API event

**YAML**

```yaml
ApiEvent:
  Type: Api
  Properties:
    Method: get
    Path: /group/{user}
    RestApiId:
      Ref: MyApi
```

**AlexaSkill**

The object describing an AlexaSkill event source type.

**Syntax**

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

**YAML**

```yaml
SkillId: String
```
Properties

SkillId

The Alexa Skill ID for your Alexa Skill. For more information about Skill ID see Configure the trigger for a Lambda function in the Alexa Skills Kit documentation.

*Type:* String

*Required:* No

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

Examples

AlexaSkillTrigger

Alexa Skill Event Example

YAML

AlexaSkillEvent:
  Type: AlexaSkill

Api

The object describing an Api event source type. If an AWS::Serverless::Api resource is defined, the path and method values must correspond to an operation in the OpenAPI definition of the API.

If no AWS::Serverless::Api resource is defined, the function input and output are a representation of the HTTP request and HTTP response.

For example, using the JavaScript API, the status code and body of the response can be controlled by returning an object with the keys statusCode and body.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

```
Auth: ApiFunctionAuth (p. 89)
Method: String
Path: String
RequestModel: RequestModel (p. 94)
RequestParameters: String | RequestParameter (p. 94)
RestApiId: String
```

Properties

Auth

Auth configuration for this specific Api+Path+Method.

Useful for overriding the API's DefaultAuthorizer setting auth config on an individual path when no DefaultAuthorizer is specified or overriding the default ApiKeyRequired setting.
Type: ApiFunctionAuth (p. 89)

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

Method

HTTP method for which this function is invoked.

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

Path

Uri path for which this function is invoked. Must start with /.

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

RequestModel

Request model to use for this specific Api+Path+Method. This should reference the name of a model specified in the Models section of an AWS::Serverless::Api (p. 32) resource.

Type: RequestModel (p. 94)

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

RequestParameters

Request parameters configuration for this specific Api+Path+Method. All parameter names must start with method.request and must be limited to method.request.header, method.request.querystring, or method.request.path.

If a parameter is a string and not a Function Request Parameter Object, then Required and Caching will default to False.

Type: String | RequestParameter (p. 94)

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

RestApiId

Identifier of a RestApi resource, which must contain an operation with the given path and method. Typically, this is set to reference an AWS::Serverless::Api (p. 32) resource defined in this template.

If you don't define this property, AWS SAM creates a default AWS::Serverless::Api (p. 32) resource using a generated OpenApi document. That resource contains a union of all paths and methods defined by Api events in the same template that do not specify a RestApiId.
This cannot reference an AWS::Serverless::Api resource defined in another template.

**Type**: String

**Required**: No

**AWS CloudFormation compatibility**: This property is unique to AWS SAM and doesn’t have an AWS CloudFormation equivalent.

### Examples

#### ApiEvent

An example of Api Event

**YAML**

```
Events:
  ApiEvent:
    Type: Api
    Properties:
      Path: /path
      Method: get
      RequestParameters:
        - method.request.header.Authorization
```

#### ApiFunctionAuth

Configures authorization at the event level, for a specific API, path, and method.

**Syntax**

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

**YAML**

```
ApiKeyRequired: Boolean
AuthorizationScopes: List
Authorizer: String
InvokeRole: String
ResourcePolicy: ResourcePolicyStatement (p. 91)
```

**Properties**

**ApiKeyRequired**

Requires an API key for this API, path, and method.

**Type**: Boolean

**Required**: No

**AWS CloudFormation compatibility**: This property is unique to AWS SAM and doesn’t have an AWS CloudFormation equivalent.

**AuthorizationScopes**

The authorization scopes to apply to this API, path, and method.
The scopes that you specify will override any scopes applied by the DefaultAuthorizer property if you have specified it.

*Type*: List

*Required*: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**Authorizer**

The Authorizer for a specific Function

If you have specified a Global Authorizer on the API and want to make a specific Function public, override by setting Authorizer to NONE.

*Type*: String

*Required*: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**InvokeRole**

Specifies the InvokeRole to use for AWS_IAM authorization.

*Type*: String

*Required*: No

*Default*: CALLER_CREDENTIALS

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

Additional notes: CALLER_CREDENTIALS maps to arn:aws:iam::*:user/*, which uses the caller credentials to invoke the endpoint.

**ResourcePolicy**

Configure Resource Policy for this path on an API.

*Type*: ResourcePolicyStatement (p. 91)

*Required*: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**Examples**

**Function-Auth**

The following example specifies authorization at the function level.

**YAML**

```yaml
Auth:
  ApiKeyRequired: true
  Authorizer: NONE
```
ResourcePolicyStatement

Configures a resource policy for all methods and paths of an API. For more information about resource policies, see Controlling access to an API with API Gateway resource policies in the API Gateway Developer Guide.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

```yaml
AwsAccountBlacklist: List
AWSAccountWhitelist: List
CustomStatements: List
IntrinsicVpcBlacklist: List
IntrinsicVpcWhitelist: List
IntrinsicVpceBlacklist: List
IntrinsicVpceWhitelist: List
IpRangeBlacklist: List
IpRangeWhiteList: List
SourceVpcBlacklist: List
SourceVpcWhiteList: List
```

Properties

**AwsAccountBlacklist**

The AWS accounts to block.

*Type*: List

*Required*: No

*AWS CloudFormation compatibility*: This property is unique to AWS SAM and doesn’t have an AWS CloudFormation equivalent.

**AwsAccountWhiteList**

The AWS accounts to allow. For an example use of this property, see the Examples section at the bottom of this page.

*Type*: List

*Required*: No

*AWS CloudFormation compatibility*: This property is unique to AWS SAM and doesn’t have an AWS CloudFormation equivalent.

**CustomStatements**

A list of custom resource policy statements to apply to this API. For an example use of this property, see the Examples section at the bottom of this page.

*Type*: List

*Required*: No

*AWS CloudFormation compatibility*: This property is unique to AWS SAM and doesn’t have an AWS CloudFormation equivalent.
**IntrinsicVpcBlacklist**

The list of virtual private clouds (VPCs) to block, where each VPC is specified as a reference such as a dynamic reference or the `Ref` intrinsic function. For an example use of this property, see the Examples section at the bottom of this page.

*Type:* List

*Required:* No

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**IntrinsicVpcWhitelist**

The list of VPCs to allow, where each VPC is specified as a reference such as a dynamic reference or the `Ref` intrinsic function.

*Type:* List

*Required:* No

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**IntrinsicVpceBlacklist**

The list of VPC endpoints to block, where each VPC endpoint is specified as a reference such as a dynamic reference or the `Ref` intrinsic function.

*Type:* List

*Required:* No

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**IntrinsicVpceWhitelist**

The list of VPC endpoints to allow, where each VPC endpoint is specified as a reference such as a dynamic reference or the `Ref` intrinsic function. For an example use of this property, see the Examples section at the bottom of this page.

*Type:* List

*Required:* No

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**IpRangeBlacklist**

The IP addresses or address ranges to block. For an example use of this property, see the Examples section at the bottom of this page.

*Type:* List

*Required:* No

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**IpRangeWhitelist**

The IP addresses or address ranges to allow.
Type: List

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

SourceVpcBlacklist

The source VPC or VPC endpoints to block. Source VPC names must start with "vpc-" and source VPC endpoint names must start with "vpce-". For an example use of this property, see the Examples section at the bottom of this page.

Type: List

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

SourceVpcWhitelist

The source VPC or VPC endpoints to allow. Source VPC names must start with "vpc-" and source VPC endpoint names must start with "vpce-".

Type: List

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

Examples

Resource Policy Example

The following Policy example blocks two IP addresses and a source VPC, and allows an AWS account.

YAML

```
Auth:
  ResourcePolicy:
    CustomStatements: [{
      "Effect": "Allow",
      "Principal": "*",
      "Action": "execute-api:Invoke",
      "Resource": "execute-api:/Prod/GET/pets",
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": "1.2.3.4"
        }
      }
    }]
  IpRangeBlacklist:
    - "10.20.30.40"
    - "1.2.3.4"
  SourceVpcBlacklist:
    - "vpce-1a2b3c4d"
  AwsAccountWhitelist:
    - "111122223333"
  IntrinsicVpcBlacklist:
    - "{{resolve:ssm:SomeVPCReference:1}}"
    - !Ref MyVPC
```
IntrinsicVpceWhitelist:
- "{{resolve:ssm:SomeVPCEReference:1}}"
- !Ref MyVPC

RequestModel

Configure Request Model for a specific Api+Path+Method.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

| Model: String |
| Required: Boolean |

Properties

Model

Name of a model defined in the Models property of the AWS::Serverless::Api (p. 32).

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

Required

adds a required property in the parameters section of OpenApi definition for given API endpoint

Type: Boolean

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

Examples

Request Model

Request Model Example

YAML

RequestModel:
  Model: User
  Required: true

RequestParameter

Configure Request Parameter for a specific Api+Path+Method.

Either Required or Caching property needs to be specified for request parameter
Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

**YAML**

```
Caching: Boolean
Required: Boolean
```

Properties

**Caching**

Adds `cacheKeyParameters` section to the API Gateway OpenApi definition

*Type:* Boolean

*Required:* Conditional

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn’t have an AWS CloudFormation equivalent.

**Required**

This field specifies whether a parameter is required

*Type:* Boolean

*Required:* Conditional

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn’t have an AWS CloudFormation equivalent.

Examples

**Request Parameter**

Example of setting Request Parameters

**YAML**

```
RequestParameters:
  - method.request.header.Authorization:
    Required: true
    Caching: true
```

**CloudWatchEvent**

The object describing a CloudWatchEvent event source type.

AWS Serverless Application Model (AWS SAM) generates an `AWS::Events::Rule` resource when this event type is set.

*Important Note:* `EventBridgeRule (p. 102)` is the preferred event source type to use, instead of CloudWatchEvent. EventBridgeRule and CloudWatchEvent use the same underlying service, API, and AWS CloudFormation resources. However, AWS SAM will add support for new features only to EventBridgeRule.
Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

```
EventBusName: String
Input: String
InputPath: String
Pattern: EventPattern
```

Properties

EventBusName

The event bus to associate with this rule. If you omit this property, AWS SAM uses the default event bus.

Type: String

Required: No

Default: Default event bus

AWS CloudFormation compatibility: This property is passed directly to the EventBusName property of an AWS::Events::Rule resource.

Input

Valid JSON text passed to the target. If you use this property, nothing from the event text itself is passed to the target.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the Input property of an AWS::Events::Rule Target resource.

InputPath

When you don't want to pass the entire matched event to the target, use the InputPath property to describe which part of the event to pass.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the InputPath property of an AWS::Events::Rule Target resource.

Pattern

Describes which events are routed to the specified target. For more information, see Events and Event Patterns in EventBridge in the Amazon EventBridge User Guide.

Type: EventPattern

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the EventPattern property of an AWS::Events::Rule resource.
Examples

CloudWatchEvent

The following is an example of a CloudWatchEvent event source type.

YAML

```yaml
CWEvent:
  Type: CloudWatchEvent
  Properties:
    Input: '{"Key": "Value"}'
    Pattern:
      detail:
        state:
        - terminated
```

CloudWatchLogs

The object describing a CloudWatchLogs event source type.

This event generates a AWS::Logs::SubscriptionFilter resource and specifies a subscription filter and associates it with the specified log group.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

```yaml
FilterPattern: String
LogGroupName: String
```

Properties

FilterPattern

The filtering expressions that restrict what gets delivered to the destination AWS resource. For more information about the filter pattern syntax, see Filter and Pattern Syntax.

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the FilterPattern property of an AWS::Logs::SubscriptionFilter resource.

LogGroupName

The log group to associate with the subscription filter. All log events that are uploaded to this log group are filtered and delivered to the specified AWS resource if the filter pattern matches the log events.

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the LogGroupName property of an AWS::Logs::SubscriptionFilter resource.
Examples

Cloudwatchlogs Subscription filter

Cloudwatchlogs Subscription filter Example

YAML

```yaml
CWLog:
  Type: CloudWatchLogs
  Properties:
    LogGroupName:
      Ref: CloudWatchLambdaLogsGroup
    FilterPattern: My pattern
```

Cognito

The object describing a Cognito event source type.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

```yaml
Trigger: List
UserPool: String
```

Properties

Trigger

The Lambda trigger configuration information for the new user pool.

Type: List

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the LambdaConfig property of an AWS::Cognito::UserPool resource.

UserPool

Reference to UserPool defined in the same template

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

Examples

Cognito Event

Cognito Event Example
YAML

CognitoUserPoolPreSignup:
  Type: Cognito
  Properties:
    UserPool:
      Ref: MyCognitoUserPool
    Trigger: PreSignUp

DynamoDB

The object describing a DynamoDB event source type. For more information, see Using AWS Lambda with Amazon DynamoDB in the AWS Lambda Developer Guide.

AWS SAM generates an AWS::Lambda::EventSourceMapping resource when this event type is set.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

BatchSize: Integer
BisectBatchOnFunctionError: Boolean
DestinationConfig: DestinationConfig
Enabled: Boolean
FunctionResponseTypes: List
MaximumBatchingWindowInSeconds: Integer
MaximumRecordAgeInSeconds: Integer
MaximumRetryAttempts: Integer
ParallelizationFactor: Integer
StartingPosition: String
Stream: String
TumblingWindowInSeconds: Integer

Properties

BatchSize

The maximum number of items to retrieve in a single batch.

Type: Integer
Required: No
Default: 100

AWS CloudFormation compatibility: This property is passed directly to the BatchSize property of an AWS::Lambda::EventSourceMapping resource.

Minimum: 1
Maximum: 1000

BisectBatchOnFunctionError

If the function returns an error, split the batch in two and retry.

Type: Boolean
Required: No

AWS CloudFormation compatibility: This property is passed directly to the BisectBatchOnFunctionError property of an AWS::Lambda::EventSourceMapping resource.

DestinationConfig

An Amazon Simple Queue Service (Amazon SQS) queue or Amazon Simple Notification Service (Amazon SNS) topic destination for discarded records.

Type: DestinationConfig

Required: No

AWS CloudFormation compatibility: This property is passed directly to the DestinationConfig property of an AWS::Lambda::EventSourceMapping resource.

Enabled

Disables the event source mapping to pause polling and invocation.

Type: Boolean

Required: No

AWS CloudFormation compatibility: This property is passed directly to the Enabled property of an AWS::Lambda::EventSourceMapping resource.

FunctionResponseTypes

A list of the response types currently applied to the event source mapping. For more information, see Reporting batch item failures in the AWS Lambda Developer Guide.

Valid values: ReportBatchItemFailures

Type: List

Required: No

AWS CloudFormation compatibility: This property is passed directly to the FunctionResponseTypes property of an AWS::Lambda::EventSourceMapping resource.

MaximumBatchingWindowInSeconds

The maximum amount of time to gather records before invoking the function, in seconds.

Type: Integer

Required: No

AWS CloudFormation compatibility: This property is passed directly to the MaximumBatchingWindowInSeconds property of an AWS::Lambda::EventSourceMapping resource.

MaximumRecordAgeInSeconds

The maximum age of a record that Lambda sends to a function for processing.

Type: Integer

Required: No

AWS CloudFormation compatibility: This property is passed directly to the MaximumRecordAgeInSeconds property of an AWS::Lambda::EventSourceMapping resource.
MaximumRetryAttempts

The maximum number of times to retry when the function returns an error.

_Type:_ Integer

_Required:_ No

_AWS CloudFormation compatibility:_ This property is passed directly to the `MaximumRetryAttempts` property of an `AWS::Lambda::EventSourceMapping` resource.

ParallelizationFactor

The number of batches to process from each shard concurrently.

_Type:_ Integer

_Required:_ No

_AWS CloudFormation compatibility:_ This property is passed directly to the `ParallelizationFactor` property of an `AWS::Lambda::EventSourceMapping` resource.

StartingPosition

The position in a stream from which to start reading.

_Valid values:_ TRIM_HORIZON or LATEST

_Type:_ String

_Required:_ Yes

_AWS CloudFormation compatibility:_ This property is passed directly to the `StartingPosition` property of an `AWS::Lambda::EventSourceMapping` resource.

Stream

The Amazon Resource Name (ARN) of the DynamoDB stream.

_Type:_ String

_Required:_ Yes

_AWS CloudFormation compatibility:_ This property is passed directly to the `EventSourceArn` property of an `AWS::Lambda::EventSourceMapping` resource.

TumblingWindowInSeconds

The duration, in seconds, of a processing window. The valid range is 1 to 900 (15 minutes).

For more information, see Tumbling windows in the _AWS Lambda Developer Guide_.

_Type:_ Integer

_Required:_ No

_AWS CloudFormation compatibility:_ This property is passed directly to the `TumblingWindowInSeconds` property of an `AWS::Lambda::EventSourceMapping` resource.

_Examples_

_DynamoDB event source for existing DynamoDB table_

DynamoDB event source for a DynamoDB table that already exists in an AWS account.
YAML

```yaml
Events:
  DDBEvent:
    Type: DynamoDB
    Properties:
      StartingPosition: TRIM_HORIZON
      BatchSize: 10
      Enabled: false
```

DynamoDB Event for DynamoDB Table Declared in Template

DynamoDB Event for a DynamoDB table that is declared in the same template file.

YAML

```yaml
Events:
  DDBEvent:
    Type: DynamoDB
    Properties:
      Stream: !GetAtt MyDynamoDBTable.StreamArn  # This must be the name of a DynamoDB table declared in the same template file
      StartingPosition: TRIM_HORIZON
      BatchSize: 10
      Enabled: false
```

EventBridgeRule

The object describing an EventBridgeRule event source type, which sets your serverless function as the target of an Amazon EventBridge rule. For more information, see What Is Amazon EventBridge? in the Amazon EventBridge User Guide.

AWS SAM generates an AWS::Events::Rule resource when this event type is set.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

```yaml
DeadLetterConfig: DeadLetterConfig (p. 104)
EventBusName: String
Input: String
InputPath: String
Pattern: EventPattern
RetryPolicy: RetryPolicy
Target: Target (p. 106)
```

Properties

DeadLetterConfig

Configure the Amazon Simple Queue Service (Amazon SQS) queue where EventBridge sends events after a failed target invocation. Invocation can fail, for example, when sending an event to a Lambda...
function that doesn't exist, or when EventBridge has insufficient permissions to invoke the Lambda function. For more information, see Event retry policy and using dead-letter queues in the Amazon EventBridge User Guide.

Note: The AWS::Serverless::Function resource type has a similar data type, DeadLetterQueue, which handles failures that occur after successful invocation of the target Lambda function. Examples of these types of failures include Lambda throttling, or errors returned by the Lambda target function. For more information about the function DeadLetterQueue property, see AWS Lambda function dead-letter queues in the AWS Lambda Developer Guide.

Type: DeadLetterConfig (p. 104)

Required: No

AWS CloudFormation compatibility: This property is similar to the DeadLetterConfig property of the AWS::Events::Rule Target data type. The AWS SAM version of this property includes additional subproperties, in case you want AWS SAM to create the dead-letter queue for you.

EventBusName

The event bus to associate with this rule. If you omit this property, AWS SAM uses the default event bus.

Type: String

Required: No

Default: Default event bus

AWS CloudFormation compatibility: This property is passed directly to the EventBusName property of an AWS::Events::Rule resource.

Input

Valid JSON text passed to the target. If you use this property, nothing from the event text itself is passed to the target.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the Input property of an AWS::Events::Rule Target resource.

InputPath

When you don't want to pass the entire matched event to the target, use the InputPath property to describe which part of the event to pass.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the InputPath property of an AWS::Events::Rule Target resource.

Pattern

Describes which events are routed to the specified target. For more information, see Events and Event Patterns in EventBridge in the Amazon EventBridge User Guide.

Type: EventPattern

Required: Yes
AWS CloudFormation compatibility: This property is passed directly to the EventPattern property of an AWS::Events::Rule resource.

RetryPolicy

A RetryPolicy object that includes information about the retry policy settings. For more information, see Event retry policy and using dead-letter queues in the Amazon EventBridge User Guide.

Type: RetryPolicy

Required: No

AWS CloudFormation compatibility: This property is passed directly to the RetryPolicy property of the AWS::Events::Rule Target data type.

Target

The AWS resource that EventBridge invokes when a rule is triggered. You can use this property to specify the logical ID of the target. If this property is not specified, then AWS SAM generates the logical ID of the target.

Type: Target (p. 106)

Required: No

AWS CloudFormation compatibility: This property is similar to the Targets property of an AWS::Events::Rule resource. The AWS SAM version of this property only allows you to specify the logical ID of a single target.

Examples

EventBridgeRule

The following is an example of an EventBridgeRule event source type.

YAML

```
EBRule:
  Type: EventBridgeRule
  Properties:
    Input: '{"Key": "Value"}
    Pattern:
      detail:
        state:
        - terminated
    RetryPolicy:
      MaximumRetryAttempts: 5
      MaximumEventAgeInSeconds: 900
    DeadLetterConfig:
      Type: SQS
      QueueLogicalId: EBRuleDLQ
    Target:
      Id: MyTarget
```

DeadLetterConfig

The object used to specify the Amazon Simple Queue Service (Amazon SQS) queue where EventBridge sends events after a failed target invocation. Invocation can fail, for example, when sending an event to a Lambda function that doesn't exist, or insufficient permissions to invoke the Lambda function. For more information, see Event retry policy and using dead-letter queues in the Amazon EventBridge User Guide.
Note: The AWS::Serverless::Function (p. 67) resource type has a similar data type, DeadLetterQueue which handles failures that occur after successful invocation of the target Lambda function. Examples of this type of failure include Lambda throttling, or errors returned by the Lambda target function. For more information about the function DeadLetterQueue property, see AWS Lambda function dead letter queues in the AWS Lambda Developer Guide.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

```
Arn: String
QueueLogicalId: String
Type: String
```

Properties

Arn

The Amazon Resource Name (ARN) of the Amazon SQS queue specified as the target for the dead-letter queue.

Note: Specify either the Type property or Arn property, but not both.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the Arn property of the AWS::Events::Rule DeadLetterConfig data type.

QueueLogicalId

The custom name of the dead letter queue that AWS SAM creates if Type is specified.

Note: If the Type property is not set, this property is ignored.

Type: String

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

Type

The type of the queue. When this property is set, AWS SAM automatically creates a dead-letter queue and attaches necessary resource-based policy to grant permission to rule resource to send events to the queue.

Note: Specify either the Type property or Arn property, but not both.

Valid values: SQS

Type: String

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.
Examples

DeadLetterConfig

YAML

```
DeadLetterConfig:
  Type: SQS
  QueueLogicalId: MyDLQ
```

Target

Configures the AWS resource that EventBridge invokes when a rule is triggered.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

```
Id: String
```

Properties

Id

The logical ID of the target.

The value of `Id` can include alphanumeric characters, periods (.), hyphens (-), and underscores (_).

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the `Id` property of the `AWS::Events::Rule Target` data type.

Examples

Target

Target

YAML

```
EBRule:
  Type: EventBridgeRule
  Properties:
    Target:
      Id: MyTarget
```

HttpApi

The object describing an event source with type HttpApi.
If an OpenApi definition for the specified path and method exists on the API, SAM will add the Lambda integration and security section (if applicable) for you.

If no OpenApi definition for the specified path and method exists on the API, SAM will create this definition for you.

**Syntax**

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

**YAML**

```yaml
ApiId: String
Auth: HttpApiFunctionAuth (p. 109)
Method: String
Path: String
PayloadFormatVersion: String
RouteSettings: RouteSettings
TimeoutInMilliseconds: Integer
```

**Properties**

**ApiId**

Identifier of an `AWS::Serverless::HttpApi (p. 128)` resource defined in this template.

If not defined, a default `AWS::Serverless::HttpApi (p. 128)` resource is created called `ServerlessHttpApi` using a generated OpenApi document containing a union of all paths and methods defined by Api events defined in this template that do not specify an `ApiId`.

This cannot reference an `AWS::Serverless::HttpApi (p. 128)` resource defined in another template.

*Type: String*

*Required: No*

*AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.*

**Auth**

Auth configuration for this specific Api+Path+Method.

Useful for overriding the API's `DefaultAuthorizer` or setting auth config on an individual path when no `DefaultAuthorizer` is specified.

*Type: HttpApiFunctionAuth (p. 109)*

*Required: No*

*AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.*

**Method**

HTTP method for which this function is invoked.

If no `Path` and `Method` are specified, SAM will create a default API path that routes any request that doesn't map to a different endpoint to this Lambda function. Only one of these default paths can exist per API.
**Type**: String  

**Required**: No  

**AWS CloudFormation compatibility**: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**Path**  

Uri path for which this function is invoked. Must start with `/`.  

If no `Path` and `Method` are specified, SAM will create a default API path that routes any request that doesn't map to a different endpoint to this Lambda function. Only one of these default paths can exist per API.  

**Type**: String  

**Required**: No  

**AWS CloudFormation compatibility**: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**PayloadFormatVersion**  

Specifies the format of the payload sent to an integration.  

NOTE: `PayloadFormatVersion` requires SAM to modify your OpenAPI definition, so it only works with inline OpenApi defined in the `DefinitionBody` property.  

**Type**: String  

**Required**: No  

**Default**: 2.0  

**AWS CloudFormation compatibility**: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**RouteSettings**  

The per-route route settings for this HTTP API. For more information about route settings, see `AWS::ApiGatewayV2::Stage RouteSettings` in the [API Gateway Developer Guide](https://docs.aws.amazon.com/serveless/latest/userguide/).  

Note: If `RouteSettings` are specified in both the `HttpApi` resource and event source, AWS SAM merges them with the event source properties taking precedence.  

**Type**: `RouteSettings`  

**Required**: No  

**AWS CloudFormation compatibility**: This property is passed directly to the `RouteSettings` property of an `AWS::ApiGatewayV2::Stage` resource.

**TimeoutInMillis**  

Custom timeout between 50 and 29,000 milliseconds.  

NOTE: `TimeoutInMillis` requires SAM to modify your OpenAPI definition, so it only works with inline OpenApi defined in the `DefinitionBody` property.  

**Type**: Integer  

**Required**: No
**Default**: 5000

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**Examples**

**Default HttpApi Event**

HttpApi Event that uses the default path. All unmapped paths and methods on this API will route to this endpoint.

**YAML**

```
Events:
  HttpApiEvent:
    Type: HttpApi
```

**HttpApi**

HttpApi Event that uses a specific path and method.

**YAML**

```
Events:
  HttpApiEvent:
    Type: HttpApi
    Properties:
      Path: /
      Method: GET
```

**HttpApi Authorization**

HttpApi Event that uses an Authorizer.

**YAML**

```
Events:
  HttpApiEvent:
    Type: HttpApi
    Properties:
      Path: /authenticated
      Method: GET
      Auth:
        Authorizer: OpenIdAuth
        AuthorizationScopes:
          - scope1
          - scope2
```

**HttpApiFunctionAuth**

Configures authorization at the event level.

Configure Auth for a specific API + Path + Method

**Syntax**

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.
YAML

```
AuthorizationScopes: List
Authorizer: String
```

Properties

AuthorizationScopes

The authorization scopes to apply to this API, path, and method.

Scopes listed here will override any scopes applied by the DefaultAuthorizer if one exists.

*Type: List
*Required: No

*AWS CloudFormation compatibility*: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

Authorizer

The Authorizer for a specific Function

If you have specified a Global Authorizer on the API and want to make a specific Function public, override by setting Authorizer to NONE.

*Type: String
*Required: No

*AWS CloudFormation compatibility*: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

Examples

Function-Auth

Specifying Authorization at Function level

YAML

```
Auth:
  Authorizer: OpenIdAuth
  AuthorizationScopes:
    - scope1
    - scope2
```

IoTRule

The object describing an IoTRule event source type.

*Creates an AWS::IoT::TopicRule resource to declare an AWS IoT rule. For more information see AWS CloudFormation documentation*

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.
**YAML**

```
AwsIotSqlVersion: String
Sql: String
```

**Properties**

**AwsIotSqlVersion**

The version of the SQL rules engine to use when evaluating the rule.

*Type*: String

*Required*: No

*AWS CloudFormation compatibility*: This property is passed directly to the `AwsIotSqlVersion` property of an `AWS::IoT::TopicRule TopicRulePayload` resource.

**Sql**

The SQL statement used to query the topic. For more information, see [AWS IoT SQL Reference](https://docs.aws.amazon.com/iot/latest/dg/sql-reference.html) in the [AWS IoT Developer Guide](https://docs.aws.amazon.com/iot/latest/dg).

*Type*: String

*Required*: Yes

*AWS CloudFormation compatibility*: This property is passed directly to the `Sql` property of an `AWS::IoT::TopicRule TopicRulePayload` resource.

**Examples**

**IOT Rule**

**IOT Rule Example**

**YAML**

```
IoTRule:
  Type: IoTRule
  Properties:
    Sql: SELECT * FROM 'topic/test'
```

**Kinesis**

The object describing a Kinesis event source type. For more information, see [Using AWS Lambda with Amazon Kinesis](https://docs.aws.amazon.com/lambda/latest/dg/lambda-kinesis.html) in the [AWS Lambda Developer Guide](https://docs.aws.amazon.com/lambda/latest/dg).

AWS SAM generates an `AWS::Lambda::EventSourceMapping` resource when this event type is set.

**Syntax**

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

**YAML**

```
BatchSize: Integer
```
**Properties**

**BatchSize**

The maximum number of items to retrieve in a single batch.

*Type:* Integer

*Required:* No

*Default:* 100

*AWS CloudFormation compatibility:* This property is passed directly to the `BatchSize` property of an `AWS::Lambda::EventSourceMapping` resource.

*Minimum:* 1

*Maximum:* 10000

**BisectBatchOnFunctionError**

If the function returns an error, split the batch in two and retry.

*Type:* Boolean

*Required:* No

*AWS CloudFormation compatibility:* This property is passed directly to the `BisectBatchOnFunctionError` property of an `AWS::Lambda::EventSourceMapping` resource.

**DestinationConfig**

An Amazon Simple Queue Service (Amazon SQS) queue or Amazon Simple Notification Service (Amazon SNS) topic destination for discarded records.

*Type:* `DestinationConfig`

*Required:* No

*AWS CloudFormation compatibility:* This property is passed directly to the `DestinationConfig` property of an `AWS::Lambda::EventSourceMapping` resource.

**Enabled**

Disables the event source mapping to pause polling and invocation.

*Type:* Boolean

*Required:* No

*AWS CloudFormation compatibility:* This property is passed directly to the `Enabled` property of an `AWS::Lambda::EventSourceMapping` resource.
FunctionResponseTypes

A list of the response types currently applied to the event source mapping. For more information, see Reporting batch item failures in the AWS Lambda Developer Guide.

Valid values: ReportBatchItemFailures

Type: List

Required: No

AWS CloudFormation compatibility: This property is passed directly to the FunctionResponseTypes property of an AWS::Lambda::EventSourceMapping resource.

MaximumBatchingWindowInSeconds

The maximum amount of time to gather records before invoking the function, in seconds.

Type: Integer

Required: No

AWS CloudFormation compatibility: This property is passed directly to the MaximumBatchingWindowInSeconds property of an AWS::Lambda::EventSourceMapping resource.

MaximumRecordAgeInSeconds

The maximum age of a record that Lambda sends to a function for processing.

Type: Integer

Required: No

AWS CloudFormation compatibility: This property is passed directly to the MaximumRecordAgeInSeconds property of an AWS::Lambda::EventSourceMapping resource.

MaximumRetryAttempts

The maximum number of times to retry when the function returns an error.

Type: Integer

Required: No

AWS CloudFormation compatibility: This property is passed directly to the MaximumRetryAttempts property of an AWS::Lambda::EventSourceMapping resource.

ParallelizationFactor

The number of batches to process from each shard concurrently.

Type: Integer

Required: No

AWS CloudFormation compatibility: This property is passed directly to the ParallelizationFactor property of an AWS::Lambda::EventSourceMapping resource.

StartingPosition

The position in a stream from which to start reading.

Valid values: TRIM_HORIZON or LATEST
Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the StartingPosition property of an AWS::Lambda::EventSourceMapping resource.

Stream

The Amazon Resource Name (ARN) of the data stream or a stream consumer.

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the EventSourceArn property of an AWS::Lambda::EventSourceMapping resource.

TumblingWindowInSeconds

The duration, in seconds, of a processing window. The valid range is 1 to 900 (15 minutes).

For more information, see Tumbling windows in the AWS Lambda Developer Guide.

Type: Integer

Required: No

AWS CloudFormation compatibility: This property is passed directly to the TumblingWindowInSeconds property of an AWS::Lambda::EventSourceMapping resource.

Examples

Kinesis event source

The following is an example of a Kinesis event source.

YAML

```
Events:
  KinesisEvent:
    Type: Kinesis
    Properties:
      StartingPosition: TRIM_HORIZON
      BatchSize: 10
      Enabled: false
```

MQ

The object describing an MQ event source type. For more information, see Using Lambda with Amazon MQ in the AWS Lambda Developer Guide.

AWS SAM generates an AWS::Lambda::EventSourceMapping resource when this event type is set.

Note: To have an Amazon MQ queue in a virtual private cloud (VPC) but your Lambda function in a public network, your function's execution role must include the following permissions: ec2:CreateNetworkInterface, ec2:DeleteNetworkInterface, ec2:DescribeNetworkInterfaces, ec2:DescribeSecurityGroups, ec2:DescribeSubnets, ec2:DescribeVpcs. For more information, see Execution role permissions in the AWS Lambda Developer Guide.

114
Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

```
BatchSize: Integer
Broker: String
Enabled: Boolean
Queues: List
SecretsManagerKmsKeyId: String
SourceAccessConfigurations: List
```

Properties

BatchSize

The maximum number of items to retrieve in a single batch.

Type: Integer

Required: No

Default: 100

*AWS CloudFormation compatibility:* This property is passed directly to the `BatchSize` property of an `AWS::Lambda::EventSourceMapping` resource.

Minimum: 1

Maximum: 10000

Broker

The Amazon Resource Name (ARN) of the Amazon MQ broker.

Type: String

Required: Yes

*AWS CloudFormation compatibility:* This property is passed directly to the `EventSourceArn` property of an `AWS::Lambda::EventSourceMapping` resource.

Enabled

If true, the event source mapping is active. To pause polling and invocation, set to `false`.

Type: Boolean

Required: No

*AWS CloudFormation compatibility:* This property is passed directly to the `Enabled` property of an `AWS::Lambda::EventSourceMapping` resource.

Queues

The name of the Amazon MQ broker destination queue to consume.

Type: List

Required: Yes
AWS CloudFormation compatibility: This property is passed directly to the `Queues` property of an `AWS::Lambda::EventSourceMapping` resource.

**SecretsManagerKmsKeyId**

The AWS Key Management Service (AWS KMS) key ID of a customer managed key from AWS Secrets Manager. This property is required if you use a customer managed key from Secrets Manager, but your Lambda execution role doesn’t include the `kms:Decrypt` permission.

The value of this property is a UUID. For example: `1abc23d4-567f-8ab9-cde0-1fab234c5d67`.

*Type*: String

*Required*: Conditional

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn’t have an AWS CloudFormation equivalent.

**SourceAccessConfigurations**

The Secrets Manager secret that stores your broker credentials. Specify secrets using the `SourceAccessConfigurations` data type.

*Type*: List

*Required*: Yes

AWS CloudFormation compatibility: This property is passed directly to the `SourceAccessConfigurations` property of an `AWS::Lambda::EventSourceMapping` resource.

**Examples**

**Amazon MQ event source**

The following is an example of an MQ event source type for an Amazon MQ broker.

**YAML**

```yaml
Events:  
MQEvent:  
  Type: MQ  
  Properties:  
    Queues: List of queues  
    SourceAccessConfigurations:  
      - Type: String  
        URI: String  
        BatchSize: 200  
        Enabled: True
```

**MSK**

The object describing an MSK event source type. For more information, see Using AWS Lambda with Amazon MSK in the AWS Lambda Developer Guide.

AWS SAM generates an `AWS::Lambda::EventSourceMapping` resource when this event type is set.

**Syntax**

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.
YAML

```yaml
StartingPosition: String
Stream: String
Topics: List
```

**Properties**

**StartingPosition**

The position in a stream from which to start reading.

*Valid values*: TRIM_HORIZON or LATEST

*Type*: String

*Required*: Yes

*AWS CloudFormation compatibility*: This property is passed directly to the `StartingPosition` property of an `AWS::Lambda::EventSourceMapping` resource.

**Stream**

The Amazon Resource Name (ARN) of the data stream or a stream consumer.

*Type*: String

*Required*: Yes

*AWS CloudFormation compatibility*: This property is passed directly to the `EventSourceArn` property of an `AWS::Lambda::EventSourceMapping` resource.

**Topics**

The name of the Kafka topic.

*Type*: List

*Required*: Yes

*AWS CloudFormation compatibility*: This property is passed directly to the `Topics` property of an `AWS::Lambda::EventSourceMapping` resource.

**Examples**

**Amazon MSK Example for Existing Cluster**

The following is an example of an `MSK` event source type for an Amazon MSK cluster that already exists in an AWS account.

YAML

```yaml
Events:
  MSKEvent:
    Type: MSK
    Properties:
      StartingPosition: LATEST
      Topics:
```

117
Amazon MSK Example for Cluster Declared in Same Template

The following is an example of an MSK event source type for an Amazon MSK cluster that is declared in the same template file.

YAML

```
Events:
  MSKEvent:
    Type: MSK
    Properties:
      StartingPosition: LATEST
      Stream:
        Ref: MyMskCluster   # This must be the name of an MSK cluster declared in the same
                          template file
      Topics:
        - MyTopic
```

S3

The object describing an S3 event source type.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

```
Bucket: String
Events: String | List
Filter: NotificationFilter
```

Properties

Bucket

S3 bucket name. This bucket must exist in the same template.

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is similar to the BucketName property of an AWS::S3::Bucket resource. This is a required field in SAM. This field only accepts a reference to the S3 bucket created in this template

Events

The Amazon S3 bucket event for which to invoke the AWS Lambda function. See Amazon S3 supported event types for a list of valid values.

Type: String | List

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the Event property of the AWS::S3::Bucket LambdaConfiguration data type.
Filter

The filtering rules that determine which Amazon S3 objects invoke the Lambda function. For information about Amazon S3 key name filtering, see Configuring Amazon S3 Event Notifications in the Amazon Simple Storage Service Developer Guide.

Type: NotificationFilter

Required: No

AWS CloudFormation compatibility: This property is passed directly to the Filter property of the AWS::S3::Bucket LambdaConfiguration data type.

Examples

S3-Event

Example of an S3 Event.

YAML

```yaml
Events:
  S3Event:
    Type: S3
    Properties:
      Bucket:
        Ref: ImagesBucket       # This must be the name of an S3 bucket declared in the same template file
      Events: s3:ObjectCreated:*
      Filter:
        S3Key:
          Rules:
            - Name: prefix       # or "suffix"
              Value: value       # The value to search for in the S3 object key names
```

Schedule

The object describing a Schedule event source type, which sets your serverless function as the target of an EventBridge rule that triggers on a schedule. For more information, see What Is Amazon EventBridge? in the Amazon EventBridge User Guide.

AWS Serverless Application Model (AWS SAM) generates an AWS::Events::Rule resource when this event type is set.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

```yaml
DeadLetterConfig: DeadLetterConfig (p. 121)
Description: String
Enabled: Boolean
Input: String
Name: String
RetryPolicy: RetryPolicy
Schedule: String
```
Properties

DeadLetterConfig

Configure the Amazon Simple Queue Service (Amazon SQS) queue where EventBridge sends events after a failed target invocation. Invocation can fail, for example, when sending an event to a Lambda function that doesn't exist, or when EventBridge has insufficient permissions to invoke the Lambda function. For more information, see Event retry policy and using dead-letter queues in the Amazon EventBridge User Guide.

Note: The AWS::Serverless::Function (p. 67) resource type has a similar data type, DeadLetterQueue, which handles failures that occur after successful invocation of the target Lambda function. Examples of these types of failures include Lambda throttling, or errors returned by the Lambda target function. For more information about the function DeadLetterQueue property, see AWS Lambda function dead-letter queues in the AWS Lambda Developer Guide.

Type: DeadLetterConfig (p. 121)

Required: No

AWS CloudFormation compatibility: This property is similar to the DeadLetterConfig property of the AWS::Events::Rule Target data type. The AWS SAM version of this property includes additional subproperties, in case you want AWS SAM to create the dead-letter queue for you.

Description

A description of the rule.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the Description property of an AWS::Events::Rule resource.

Enabled

Indicates whether the rule is enabled.

To disable the rule, set this property to False.

Type: Boolean

Required: No

AWS CloudFormation compatibility: This property is similar to the State property of an AWS::Events::Rule resource. If this property is set to True then AWS SAM passes ENABLED, otherwise it passes DISABLED.

Input

Valid JSON text passed to the target. If you use this property, nothing from the event text itself is passed to the target.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the Target property of an AWS::Events::Rule Target resource.

Name

The name of the rule. If you don't specify a name, AWS CloudFormation generates a unique physical ID and uses that ID for the rule name.
Type: String

Required: No

*AWS CloudFormation compatibility:* This property is passed directly to the `Name` property of an `AWS::Events::Rule` resource.

RetryPolicy

A `RetryPolicy` object that includes information about the retry policy settings. For more information, see *Event retry policy and using dead-letter queues* in the *Amazon EventBridge User Guide*.

Type: `RetryPolicy`

Required: No

*AWS CloudFormation compatibility:* This property is passed directly to the `RetryPolicy` property of the `AWS::Events::Rule` Target data type.

Schedule

The scheduling expression that determines when and how often the rule runs. For more information, see *Schedule Expressions for Rules*.

Type: String

Required: Yes

*AWS CloudFormation compatibility:* This property is passed directly to the `ScheduleExpression` property of an `AWS::Events::Rule` resource.

Examples

CloudWatch Schedule Event

CloudWatch Schedule Event Example

**YAML**

```
CWSchedule:
  Type: Schedule
  Properties:
    Schedule: 'rate(1 minute)'
    Name: TestSchedule
    Description: test schedule
    Enabled: False
```

DeadLetterConfig

The object used to specify the Amazon Simple Queue Service (Amazon SQS) queue where EventBridge sends events after a failed target invocation. Invocation can fail, for example, when sending an event to a Lambda function that doesn't exist, or insufficient permissions to invoke the Lambda function. For more information, see *Event retry policy and using dead-letter queues* in the *Amazon EventBridge User Guide*.

**Note:** The `AWS::Serverless::Function` (p. 67) resource type has a similar data type, `DeadLetterQueue` which handles failures that occur after successful invocation of the target Lambda function. Examples of this type of failure include Lambda throttling, or errors returned by the Lambda target function. For more information about the function `DeadLetterQueue` property, see *AWS Lambda function dead letter queues* in the AWS Lambda Developer Guide.
Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

**YAML**

```
Arn: String
QueueLogicalId: String
Type: String
```

**Properties**

**Arn**

The Amazon Resource Name (ARN) of the Amazon SQS queue specified as the target for the dead-letter queue.

*Note:* Specify either the `Type` property or `Arn` property, but not both.

*Type:* String

*Required:* No

*AWS CloudFormation compatibility:* This property is passed directly to the `Arn` property of the `AWS::Events::Rule DeadLetterConfig` data type.

**QueueLogicalId**

The custom name of the dead letter queue that AWS SAM creates if `Type` is specified.

*Note:* If the `Type` property is not set, this property is ignored.

*Type:* String

*Required:* No

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**Type**

The type of the queue. When this property is set, AWS SAM automatically creates a dead-letter queue and attaches necessary resource-based policy to grant permission to rule resource to send events to the queue.

*Note:* Specify either the `Type` property or `Arn` property, but not both.

*Valid values:* SQS

*Type:* String

*Required:* No

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**Examples**

**DeadLetterConfig**

DeadLetterConfig
YAML

```
DeadLetterConfig:
  Type: SQS
  QueueLogicalId: MyDLQ
```

**SNS**

The object describing an SNS event source type.

SAM generates `AWS::SNS::Subscription` resource when this event type is set

**Syntax**

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

**YAML**

```
FilterPolicy: SnsFilterPolicy
Region: String
SqsSubscription: Boolean | SqsSubscriptionObject (p. 124)
Topic: String
```

**Properties**

*FilterPolicy*

The filter policy JSON assigned to the subscription. For more information, see `GetSubscriptionAttributes` in the Amazon Simple Notification Service API Reference.

*Type*: SnsFilterPolicy

*Required*: No

*AWS CloudFormation compatibility*: This property is passed directly to the `FilterPolicy` property of an `AWS::SNS::Subscription` resource.

*Region*

For cross-region subscriptions, the region in which the topic resides.

If no region is specified, CloudFormation uses the region of the caller as the default.

*Type*: String

*Required*: No

*AWS CloudFormation compatibility*: This property is passed directly to the `Region` property of an `AWS::SNS::Subscription` resource.

*SqsSubscription*

Set this property to true, or specify `SqsSubscriptionObject` to enable batching SNS topic notifications in an SQS queue. Setting this property to true creates a new SQS queue, whereas specifying a `SqsSubscriptionObject` uses an existing SQS queue.

*Type*: Boolean | `SqsSubscriptionObject (p. 124)`

*Required*: No
**AWS CloudFormation compatibility**: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**Topic**

The ARN of the topic to subscribe to.

*Type: String*

*Required: Yes*

**AWS CloudFormation compatibility**: This property is passed directly to the `TopicArn` property of an `AWS::SNS::Topic` resource.

**Examples**

**SNS Event Source Example**

**SNS Event Source Example**

**YAML**

```yaml
Events:
  SNS:
    Type: SNS
    Properties:
      Topic:
        arn:aws:sns:us-east-1:123456789012:my_topic
      SqsSubscription: True
      FilterPolicy:
        store:
          - example_corp
        price_usd:
          - numeric:
            + ">="
            - 100
```

**SqsSubscriptionObject**

Specify an existing SQS queue option to SNS event

**Syntax**

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

**YAML**

```yaml
BatchSize: String
Enabled: Boolean
QueueArn: String
QueuePolicyLogicalId: String
QueueUrl: String
```

**Properties**

**BatchSize**

The maximum number of items to retrieve in a single batch for the SQS queue.

*Type: String*
**Required:** No

**Default:** 10

**AWS CloudFormation compatibility:** This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**Enabled**

Disables the SQS event source mapping to pause polling and invocation.

**Type:** Boolean

**Required:** No

**Default:** True

**AWS CloudFormation compatibility:** This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**QueueArn**

Specify an existing SQS queue arn.

**Type:** String

**Required:** Yes

**AWS CloudFormation compatibility:** This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**QueuePolicyLogicalId**

Give a custom logicalId name for the AWS::SQS::QueuePolicy resource.

**Type:** String

**Required:** No

**AWS CloudFormation compatibility:** This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**QueueUrl**

Specify the queue URL associated with the QueueArn property.

**Type:** String

**Required:** Yes

**AWS CloudFormation compatibility:** This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**Examples**

**Existing SQS for SNS event**

Example to add existing SQS queue for subscribing to an SNS topic.

**YAML**

```yaml
QueuePolicyLogicalId: CustomQueuePolicyLogicalId
QueueArn: 
```
**SQS**

The object describing an SQS event source type. For more information, see Using AWS Lambda with Amazon SQS in the AWS Lambda Developer Guide.

SAM generates AWS::Lambda::EventSourceMapping resource when this event type is set.

**Syntax**

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

**YAML**

```yaml
BatchSize: Integer
Enabled: Boolean
MaximumBatchingWindowInSeconds: Integer
Queue: String
```

**Properties**

**BatchSize**

The maximum number of items to retrieve in a single batch.

- *Type*: Integer
- *Required*: No
- *Default*: 10

*AWS CloudFormation compatibility*: This property is passed directly to the `BatchSize` property of an AWS::Lambda::EventSourceMapping resource.

- *Minimum*: 1
- *Maximum*: 10000

**Enabled**

Disables the event source mapping to pause polling and invocation.

- *Type*: Boolean
- *Required*: No

*AWS CloudFormation compatibility*: This property is passed directly to the `Enabled` property of an AWS::Lambda::EventSourceMapping resource.

**MaximumBatchingWindowInSeconds**

The maximum amount of time, in seconds, to gather records before invoking the function.

- *Type*: Integer
- *Required*: No
**AWS CloudFormation compatibility:** This property is passed directly to the `MaximumBatchingWindowInSeconds` property of an `AWS::Lambda::EventSourceMapping` resource.

**Queue**

The ARN of the queue.

*Type:* String  
*Required:* Yes

**AWS CloudFormation compatibility:** This property is passed directly to the `EventSourceArn` property of an `AWS::Lambda::EventSourceMapping` resource.

**Examples**

**SQS Event**

**YAML**

```yaml
Events:
  SQSEvent:
    Type: SQS
    Properties:
      BatchSize: 10
      Enabled: false
```

**FunctionCode**

The deployment package for a Lambda function.

**Syntax**

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

**YAML**

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

**Properties**

**Bucket**

An Amazon S3 bucket in the same AWS Region as your function.

*Type:* String  
*Required:* Yes

**AWS CloudFormation compatibility:** This property is passed directly to the `S3Bucket` property of the `AWS::Lambda::Function Code` data type.
Key

The Amazon S3 key of the deployment package.

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the S3Key property of the AWS::Lambda::Function Code data type.

Version

For versioned objects, the version of the deployment package object to use.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the S3ObjectVersion property of the AWS::Lambda::Function Code data type.

Examples

FunctionCode

Function Code example

YAML

```
FunctionCode:
  Bucket: mybucket-name
  Key: mykey-name
  Version: 121212
```

AWS::Serverless::HttpApi

Creates an Amazon API Gateway HTTP API, which enables you to create RESTful APIs with lower latency and lower costs than REST APIs. For more information, see Working with HTTP APIs in the API Gateway Developer Guide.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

```
Type: AWS::Serverless::HttpApi
Properties:
  AccessLogSettings: AccessLogSettings
  Auth: HttpApiAuth (p. 135)
  CorsConfiguration: String | HttpApiCorsConfiguration (p. 141)
  DefaultRouteSettings: RouteSettings
  DefinitionBody: String
  DefinitionUri: String | HttpApiDefinition (p. 143)
  Description: String
  DisableExecuteApiEndpoint: Boolean
```
Properties

AccessLogSettings

The settings for access logging in a stage.

*Type:* AccessLogSettings

*Required:* No

*AWS CloudFormation compatibility:* This property is passed directly to the `AccessLogSettings` property of an `AWS::ApiGatewayV2::Stage` resource.

Auth

Configures authorization for controlling access to your API Gateway HTTP API.

For more information, see [Controlling access to HTTP APIs with JWT authorizers](https://docs.aws.amazon.com/apigateway/latest/developerguide/authorizers-jwt.html) in the *API Gateway Developer Guide*.

*Type:* `HttpApiAuth (p. 135)`

*Required:* No

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

CorsConfiguration

Manages cross-origin resource sharing (CORS) for all your API Gateway HTTP APIs. Specify the domain to allow as a string, or specify an `HttpApiCorsConfiguration` object. Note that CORS requires AWS SAM to modify your OpenAPI definition, so CORS works only if the `DefinitionBody` property is specified.

For more information, see [Configuring CORS for an HTTP API](https://docs.aws.amazon.com/apigateway/latest/developerguide/cors.html) in the *API Gateway Developer Guide*.

*Note:* If `CorsConfiguration` is set both in an OpenAPI definition and at the property level, then AWS SAM merges both configuration sources with the properties taking precedence.

*Note:* If this property is set to `true`, then all origins are allowed.

*Type:* `String | HttpApiCorsConfiguration (p. 141)`

*Required:* No

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

DefaultRouteSettings

The default route settings for this HTTP API. These settings apply to all routes unless overridden by the `RouteSettings` property for certain routes.

*Type:* `RouteSettings`

*Required:* No
**AWS CloudFormation compatibility:** This property is passed directly to the `RouteSettings` property of an `AWS::ApiGatewayV2::Stage` resource.

**DefinitionBody**

The OpenAPI definition that describes your HTTP API. If you don't specify a `DefinitionUri` or a `DefinitionBody`, AWS SAM generates a `DefinitionBody` for you based on your template configuration.

*Type:* String

*Required:* No

**AWS CloudFormation compatibility:** This property is similar to the `Body` property of an `AWS::ApiGatewayV2::Api` resource. If certain properties are provided, AWS SAM may insert content into or modify the `DefinitionBody` before it is passed to AWS CloudFormation. Properties include `Auth` and an `EventSource` of type `HttpApi` for a corresponding `AWS::Serverless::Function` resource.

**DefinitionUri**

The Amazon Simple Storage Service (Amazon S3) URI, local file path, or location object of the the OpenAPI definition that defines the HTTP API. The Amazon S3 object that this property references must be a valid OpenAPI definition file. If you don't specify a `DefinitionUri` or a `DefinitionBody` are specified, AWS SAM generates a `DefinitionBody` for you based on your template configuration.

If you provide a local file path, the template must go through the workflow that includes the `sam deploy` or `sam package` command for the definition to be transformed properly.

Intrinsic functions are not supported in external OpenAPI definition files that you reference with `DefinitionUri`. To import an OpenAPI definition into the template, use the `DefinitionBody` property with the `Include transform`.

*Type:* String | [HttpApiDefinition (p. 143)]

*Required:* No

**AWS CloudFormation compatibility:** This property is similar to the `BodyS3Location` property of an `AWS::ApiGatewayV2::Api` resource. The nested Amazon S3 properties are named differently.

**Description**

A description of the HttpApi resource.

*Note:* This property requires AWS SAM to modify the HttpApi resource's OpenAPI definition, to set the description field. The following two scenarios result in an error: 1) The `DefinitionBody` property is specified with the `description` field set in the OpenAPI definition (since this is a conflict that AWS SAM won't resolve), or 2) The `DefinitionUri` property is specified (since AWS SAM won't modify an OpenAPI definition that it retrieves from Amazon S3).

*Type:* String

*Required:* No

**AWS CloudFormation compatibility:** This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**DisableExecuteApiEndpoint**

Specifies whether clients can invoke your HTTP API by using the default `execute-api` endpoint `https://{api_id}.execute-api.{region}.amazonaws.com`. By default, clients can invoke
your API with the default endpoint. To require that clients only use a custom domain name to invoke your API, disable the default endpoint.

_type_: Boolean

_required_: No

.aws cloudformation compatibility: This property is passed directly to the disableexecuteapiproperty of an _aws-api-gateway-v2-api resource.

_domain_

Configures a custom domain for this API Gateway HTTP API.

_type_: HttpApiDomainConfiguration (p. 144)

_required_: No

.aws cloudformation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

_failonwarnings_

Specifies whether to roll back the HTTP API creation (true) or not (false) when a warning is encountered. The default value is false.

_type_: Boolean

_required_: No

.aws cloudformation compatibility: This property is passed directly to the failonwarnings property of an _aws-api-gateway-v2-api resource.

_route_settings_

The route settings, per route, for this HTTP API. For more information, see Working with routes for HTTP APIs in the API Gateway Developer Guide.

_type_: RouteSettings

_required_: No

.aws cloudformation compatibility: This property is passed directly to the _route_settings_ property of an _aws-api-gateway-v2-stage resource.

_stage_name_

The name of the API stage. If no name is specified, AWS SAM uses the $default stage from API Gateway.

_type_: String

_required_: No

_default_: $default

.aws cloudformation compatibility: This property is passed directly to the _stage_name_ property of an _aws-api-gateway-v2-stage resource.

_stage_variables_

A map that defines the stage variables. Variable names can have alphanumeric and underscore characters. The values must match [A-Za-z0-9-_~:/?#&=,]+.
Type: Json

Required: No

AWS CloudFormation compatibility: This property is passed directly to the `StageVariables` property of an `AWS::ApiGatewayV2::Stage` resource.

Tags

A map (string to string) that specifies the tags to add to this API Gateway stage. Keys and values are limited to alphanumeric characters. Keys can be 1 to 127 Unicode characters in length and cannot include the prefix `aws:`. Values can be 1 to 255 Unicode characters in length.

Type: Map

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

Additional notes: The Tags property requires AWS SAM to modify your OpenAPI definition, so tags are added only if the `DefinitionBody` property is specified—no tags are added if the `DefinitionUri` property is specified. AWS SAM automatically adds an `httpapi:createdBy:SAM` tag. Tags are also added to the `AWS::ApiGatewayV2::Stage` resource and the `AWS::ApiGatewayV2::DomainName` resource (if `DomainName` is specified).

Return Values

Ref

When you pass the logical ID of this resource to the intrinsic Ref function, Ref returns the API ID of the underlying AWS::ApiGatewayV2::Api resource, for example, a1bcdef2gh.

For more information about using the Ref function, see Ref in the AWS CloudFormation User Guide.

Examples

Simple HttpApi

The following example shows the minimum needed to set up an HTTP API endpoint backed by a Lambda function. This example uses the default HTTP API that AWS SAM creates.

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Description: AWS SAM template with a simple API definition
Resources:
  ApiFunction:
    Type: AWS::Serverless::Function
    Properties:
      Events:
        ApiEvent:
          Type: HttpApi
          Handler: index.handler
          InlineCode: |
            def handler(event, context):
              return {'body': 'Hello World!', 'statusCode': 200}
          Runtime: python3.7
    Transform: AWS::Serverless-2016-10-31
```
HttpApi with Auth

The following example shows how to set up authorization on HTTP API endpoints.

```
YAML

Properties:
  FailOnWarnings: True
  Auth:
    DefaultAuthorizer: OAuth2
    Authorizers:
      OAuth2:
        AuthorizationScopes:
        - scope4
        JwtConfiguration:
          issuer: "https://www.example.com/v1/connect/oauth2"
          audience:
            - MyApi
          IdentitySource: "#request.querystring.param"
      OpenIdAuth:
        AuthorizationScopes:
        - scope1
        - scope2
        OpenIdConnectUrl: "https://www.example.com/v1/connect/oidc/.well-known/openid-configuration"
        JwtConfiguration:
          issuer: "https://www.example.com/v1/connect/oidc"
          audience:
            - MyApi
          IdentitySource: "#request.querystring.param"
```

HttpApi with OpenAPI definition

The following example shows how to add an OpenAPI definition to the template.

Note that AWS SAM fills in any missing Lambda integrations for HttpApi events that reference this HTTP API. AWS SAM also also adds any missing paths that HttpApi events reference.

```
YAML

Properties:
  FailOnWarnings: True
  DefinitionBody:
    info:
      version: '1.0'
      title:
        Ref: AWS::StackName
    paths:
      ":":
        get:
          security:
            - OpenIdAuth:
              - scope1
              - scope2
          responses: {}
    openapi: 3.0.1
    securitySchemes:
      OpenIdAuth:
        type: openidConnect
        x-amazon-apigateway-authorizer:
          identitySource: "#request.querystring.param"
        type: jwt
```
jwtConfiguration:
  audience:
    - MyApi
  issuer: https://www.example.com/v1/connect/oidc
  openIdConnectUrl: https://www.example.com/v1/connect/oidc/.well-known/openid-configuration

HttpApi with configuration settings

The following example shows how to add HTTP API and stage configurations to the template.

**YAML**

```yaml
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Parameters:
  StageName:
    Type: String
    Default: Prod
Resources:
  HttpApiFunction:
    Type: AWS::Serverless::Function
    Properties:
      InlineCode: |
        def handler(event, context):
            import json
            return {
                "statusCode": 200,
                "body": json.dumps(event),
            }
      Handler: index.handler
      Runtime: python3.7
      Events:
        ExplicitApi: # warning: creates a public endpoint
          Type: HttpApi
          Properties:
            ApiId: !Ref HttpApi
            Method: GET
            Path: /path
            TimeoutInMillis: 15000
            PayloadFormatVersion: "2.0"
            RouteSettings:
              ThrottlingBurstLimit: 600
      HttpApi:
        Type: AWS::Serverless::HttpApi
        Properties:
          StageName: !Ref StageName
          Tags:
            Tag: Value
          AccessLogSettings:
            DestinationArn: !GetAtt AccessLogs.Arn
            Format: $context.requestId
          DefaultRouteSettings:
            ThrottlingBurstLimit: 200
          RouteSettings:
            "GET /path":
              ThrottlingBurstLimit: 500 # overridden in HttpApi Event
          StageVariables:
            StageVar: Value
          FailOnWarnings: True
```

AccessLogs:
HttpApiAuth

Configure authorization to control access to your Amazon API Gateway HTTP API.

For more information about configuring access to HTTP APIs, see Controlling and managing access to an HTTP API in API Gateway in the API Gateway Developer Guide.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

```
Authorizers: OAuth2Authorizer (p. 139) | LambdaAuthorizer (p. 136)
DefaultAuthorizer: String
```

Properties

Authorizers

The authorizer used to control access to your API Gateway API.

- **Type:** OAuth2Authorizer (p. 139) | LambdaAuthorizer (p. 136)
- **Required:** No
- **Default:** None

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

*Additional notes:* AWS SAM adds the authorizers to the OpenAPI definition.

DefaultAuthorizer

Specify the default authorizer to use for authorizing API calls to your API Gateway API.

- **Type:** String
- **Required:** No
- **Default:** None

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.
Examples

OAuth 2.0 Authorizer

OAuth 2.0 authorizer example

YAML

```yaml
Auth:
  Authorizers:
    OAuth2Authorizer:
      AuthorizationScopes:
        - scope1
        - scope2
      JwtConfiguration:
        issuer: "https://www.example.com/v1/connect/oauth2"
        audience:
        - MyApi
      IdentitySource: "$request.querystring.param"
    DefaultAuthorizer: OAuth2Authorizer

LambdaAuthorizer

Configure a Lambda authorizer to control access to your Amazon API Gateway HTTP API with an AWS Lambda function.

For more information and examples, see Working with AWS Lambda authorizers for HTTP APIs in the API Gateway Developer Guide.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

```yaml
AuthorizerPayloadFormatVersion: String
EnableSimpleResponses: Boolean
FunctionArn: String
FunctionInvokeRole: String
Identity: LambdaAuthorizationIdentity (p. 138)

Properties

AuthorizerPayloadFormatVersion

Specifies the format of the payload sent to an HTTP API Lambda authorizer. Required for HTTP API Lambda authorizers.

This is passed through to the authorizerPayloadFormatVersion section of an x-amazon-apigateway-authorizer in the securitySchemes section of an OpenAPI definition.

Valid values: 1.0 or 2.0

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.
EnableSimpleResponses

Specifies whether a Lambda authorizer returns a response in a simple format. By default, a Lambda authorizer must return an AWS Identity and Access Management (IAM) policy. If enabled, the Lambda authorizer can return a boolean value instead of an IAM policy.

This is passed through to the enableSimpleResponses section of an x-amazon-apigateway-authorizer in the securitySchemes section of an OpenAPI definition.

Type: Boolean

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn’t have an AWS CloudFormation equivalent.

FunctionArn

The Amazon Resource Name (ARN) of the Lambda function that provides authorization for the API.

This is passed through to the authorizerUri section of an x-amazon-apigateway-authorizer in the securitySchemes section of an OpenAPI definition.

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn’t have an AWS CloudFormation equivalent.

FunctionInvokeRole

The ARN of the IAM role that has the credentials required to invoke the authorizer.

This is passed through to the authorizerCredentials section of an x-amazon-apigateway-authorizer in the securitySchemes section of an OpenAPI definition.

Type: String

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn’t have an AWS CloudFormation equivalent.

Identity

Specifies an IdentitySource in an incoming request for an authorizer.

This is passed through to the identitySource section of an x-amazon-apigateway-authorizer in the securitySchemes section of an OpenAPI definition.

Type: LambdaAuthorizationIdentity (p. 138)

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn’t have an AWS CloudFormation equivalent.

Examples

LambdaAuthorizer

LambdaAuthorizer example
YAML

```yaml
Auth:
  Authorizers:
    MyLambdaAuthorizer:
      AuthorizerPayloadFormatVersion: 2.0
      FunctionArn:
        Fn::GetAtt:
        - MyAuthFunction
        - Arn
      FunctionInvokeRole:
        Fn::GetAtt:
        - LambdaAuthInvokeRole
        - Arn
      Identity:
        Headers:
        - Authorization
```

**LambdaAuthorizationIdentity**

Use property can be used to specify an IdentitySource in an incoming request for a Lambda authorizer. For more information about identity sources, see [Identity sources](#) in the *API Gateway Developer Guide*.

**Syntax**

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

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

**Properties**

**Context**

Converts the given context strings to a list of mapping expressions in the format `$context.contextString`.

*Type:* List

*Required:* No

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**Headers**

Converts the headers to a list of mapping expressions in the format `$request.header.name`.

*Type:* List

*Required:* No

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.
QueryStrings

Converts the given query strings to a list of mapping expressions in the format
$request.querystring.queryString.

*Type*: List

*Required*: No

*AWS CloudFormation compatibility*: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

ReauthorizeEvery

The time-to-live (TTL) period, in seconds, that specifies how long API Gateway caches authorizer results. If you specify a value greater than 0, API Gateway caches the authorizer responses. The maximum value is 3600, or 1 hour.

*Type*: Integer

*Required*: No

*AWS CloudFormation compatibility*: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

StageVariables

Converts the given stage variables to a list of mapping expressions in the format
$stageVariables.stageVariable.

*Type*: List

*Required*: No

*AWS CloudFormation compatibility*: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

Examples

LambdaRequestIdentity

Lambda request identity example

YAML

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

OAuth2Authorizer

Definition for an OAuth 2.0 authorizer, also known to as a JSON Web Token (JWT) authorizer.

For more information, see Controlling access to HTTP APIs with JWT authorizers in the *API Gateway Developer Guide*. 
Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

```
AuthorizationScopes: List
IdentitySource: String
JwtConfiguration: Map
```

Properties

AuthorizationScopes

List of authorization scopes for this authorizer.

Type: List

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

IdentitySource

Identity source expression for this authorizer.

Type: String

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

JwtConfiguration

JWT configuration for this authorizer.

This is passed through to the jwtConfiguration section of an x-amazon-apigateway-authorizer in the securitySchemes section of an OpenAPI definition.

Type: Map

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

Examples

OAuth 2.0 authorizer

OAuth 2.0 authorizer Example

YAML

```
Auth:
  Authorizers:
```

140
HttpApiCorsConfiguration

Manage cross-origin resource sharing (CORS) for your HTTP APIs. Specify the domain to allow as a string or specify a dictionary with additional Cors configuration. NOTE: Cors requires SAM to modify your OpenAPI definition, so it only works with inline OpenApi defined in the DefinitionBody property.

For more information about CORS, see Configuring CORS for an HTTP API in the API Gateway Developer Guide.

Note: If HttpApiCorsConfiguration is set both in OpenAPI and at the property level, AWS SAM merges them with the properties taking precedence.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

```
AllowCredentials: Boolean
AllowHeaders: List
AllowMethods: List
AllowOrigins: List
ExposeHeaders: List
MaxAge: Integer
```

Properties

AllowCredentials

Specifies whether credentials are included in the CORS request.

Type: Boolean

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

AllowHeaders

Represents a collection of allowed headers.

Type: List

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.
AllowMethods

Represents a collection of allowed HTTP methods.

Type: List

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

AllowOrigins

Represents a collection of allowed origins.

Type: List

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

ExposeHeaders

Represents a collection of exposed headers.

Type: List

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

MaxAge

The number of seconds that the browser should cache preflight request results.

Type: Integer

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

Examples

HttpApiCorsConfiguration

HTTP API Cors Configuration example.

YAML

```yaml
CorsConfiguration:
  AllowOrigins:
    - "https://example.com"
  AllowHeaders:
    - x-apigateway-header
  AllowMethods:
    - GET
  MaxAge: 600
  AllowCredentials: True
```
HttpApiDefinition

An OpenAPI document defining the API.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

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

Properties

Bucket

The name of the Amazon S3 bucket where the OpenAPI file is stored.

- **Type**: String
- **Required**: Yes

AWS CloudFormation compatibility: This property is passed directly to the `Bucket` property of the `AWS::ApiGatewayV2::Api BodyS3Location` data type.

Key

The Amazon S3 key of the OpenAPI file.

- **Type**: String
- **Required**: Yes

AWS CloudFormation compatibility: This property is passed directly to the `Key` property of the `AWS::ApiGatewayV2::Api BodyS3Location` data type.

Version

For versioned objects, the version of the OpenAPI file.

- **Type**: String
- **Required**: No

AWS CloudFormation compatibility: This property is passed directly to the `Version` property of the `AWS::ApiGatewayV2::Api BodyS3Location` data type.

Examples

Definition Uri example

API Definition example

YAML

```
DefinitionUri:
```
HttpApiDomainConfiguration

Configures a custom domain for an API.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

```
BasePath: List
CertificateArn: String
DomainName: String
EndpointConfiguration: String
MutualTlsAuthentication: MutualTlsAuthentication
Route53: Route53Configuration (p. 146)
SecurityPolicy: String
```

Properties

BasePath

A list of the basepaths to configure with the Amazon API Gateway domain name.

*Type*: List

*Required*: No

*Default*: /

*AWS CloudFormation compatibility*: This property is similar to the `ApiMappingKey` property of an AWS::ApiGatewayV2::ApiMapping resource. AWS SAM creates multiple AWS::ApiGatewayV2::ApiMapping resources, one per value specified in this property.

CertificateArn

The Amazon Resource Name (ARN) of an AWS managed certificate for this domain name's endpoint. AWS Certificate Manager is the only supported source.

*Type*: String

*Required*: Yes

*AWS CloudFormation compatibility*: This property is passed directly to the `CertificateArn` property of an AWS::ApiGatewayV2::DomainName DomainNameConfiguration resource.

DomainName

The custom domain name for your API Gateway API. Uppercase letters are not supported.

AWS SAM generates an AWS::ApiGatewayV2::DomainName resource when this property is set. For information about this scenario, see DomainName property is specified (p. 180). For information about generated AWS CloudFormation resources, see Generated AWS CloudFormation resources (p. 174).
**Type**: String

**Required**: Yes

**AWS CloudFormation compatibility**: This property is passed directly to the `DomainName` property of an `AWS::ApiGatewayV2::DomainName` resource.

**EndpointConfiguration**

Defines the type of API Gateway endpoint to map to the custom domain. The value of this property determines how the `CertificateArn` property is mapped in AWS CloudFormation.

The only valid value for HTTP APIs is `REGIONAL`.

**Type**: String

**Required**: No

**Default**: `REGIONAL`

**AWS CloudFormation compatibility**: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**MutualTlsAuthentication**

The mutual transport layer security (TLS) authentication configuration for a custom domain name.

**Type**: `MutualTlsAuthentication`

**Required**: No

**AWS CloudFormation compatibility**: This property is passed directly to the `MutualTlsAuthentication` property of an `AWS::ApiGatewayV2::DomainName` resource.

**Route53**

Defines an Amazon Route 53 configuration.

**Type**: `Route53Configuration (p. 146)`

**Required**: No

**AWS CloudFormation compatibility**: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**SecurityPolicy**

The TLS version of the security policy for this domain name.

The only valid value for HTTP APIs is `TLS_1_2`.

**Type**: String

**Required**: No

**AWS CloudFormation compatibility**: This property is passed directly to the `SecurityPolicy` property of the `AWS::ApiGatewayV2::DomainName DomainNameConfiguration data type`.

**Examples**

**DomainName**

DomainName example
YAML

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

Route53Configuration

Configures the Route53 record sets for an API.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

```yaml
DistributionDomainName: String
EvaluateTargetHealth: Boolean
HostedZoneId: String
HostedZoneName: String
IpV6: Boolean
```

Properties

DistributionDomainName

Configures a custom distribution of the API custom domain name.

Type: String

Required: No

Default: Use the API Gateway distribution.

AWS CloudFormation compatibility: This property is passed directly to the `DNSName` property of an `AWS::Route53::RecordSetGroup AliasTarget` resource.

Additional notes: The domain name of a CloudFront distribution.

EvaluateTargetHealth

When EvaluateTargetHealth is true, an alias record inherits the health of the referenced AWS resource, such as an Elastic Load Balancing load balancer or another record in the hosted zone.

Type: Boolean

Required: No

AWS CloudFormation compatibility: This property is passed directly to the `EvaluateTargetHealth` property of an `AWS::Route53::RecordSetGroup AliasTarget` resource.

Additional notes: You can't set EvaluateTargetHealth to true when the alias target is a CloudFront distribution.
HostedZoneId

The ID of the hosted zone that you want to create records in.

Specify either HostedZoneName or HostedZoneId, but not both. If you have multiple hosted zones with the same domain name, you must specify the hosted zone using HostedZoneId.

*Type:* String

*Required:* No

*AWS CloudFormation compatibility:* This property is passed directly to the HostedZoneId property of an AWS::Route53::RecordSetGroup RecordSet resource.

HostedZoneName

The name of the hosted zone that you want to create records in.

Specify either HostedZoneName or HostedZoneId, but not both. If you have multiple hosted zones with the same domain name, you must specify the hosted zone using HostedZoneId.

*Type:* String

*Required:* No

*AWS CloudFormation compatibility:* This property is passed directly to the HostedZoneName property of an AWS::Route53::RecordSetGroup RecordSet resource.

IpV6

When this property is set, AWS SAM creates a AWS::Route53::RecordSet resource and sets Type to AAAA for the provided HostedZone.

*Type:* Boolean

*Required:* No

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**Examples**

**Route 53 Configuration Example**

This example shows how to configure Route 53.

**YAML**

```
Domain:
  DomainName: www.example.com
  CertificateArn: arn-example
  EndpointConfiguration: EDGE
  Route53:
    HostedZoneId: Z1PA6795UKMFR9
    EvaluateTargetHealth: true
    DistributionDomainName: xyz
```

**AWS::Serverless::LayerVersion**

Creates a Lambda LayerVersion that contains library or runtime code needed by a Lambda Function.
The AWS::Serverless::LayerVersion (p. 147) resource also supports the Metadata resource attribute, so you can instruct AWS SAM to build layers included in your application. For more information about building layers, see Building layers (p. 202).

**Important Note:** Since the release of the UpdateReplacePolicy resource attribute in AWS CloudFormation, AWS::Lambda::LayerVersion (recommended) offers the same benefits as AWS::Serverless::LayerVersion (p. 147).

When a Serverless LayerVersion is transformed, SAM also transforms the logical id of the resource so that old LayerVersions are not automatically deleted by CloudFormation when the resource is updated.

## Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

**YAML**

```yaml
Type: AWS::Serverless::LayerVersion
Properties:
  CompatibleRuntimes: List
  ContentUri: String | LayerContent (p. 150)
  Description: String
  LayerName: String
  LicenseInfo: String
  RetentionPolicy: String
```

### Properties

**CompatibleRuntimes**

List of runtimes compatible with this LayerVersion.

- **Type:** List
- **Required:** No

  **AWS CloudFormation compatibility:** This property is passed directly to the `CompatibleRuntimes` property of an AWS::Lambda::LayerVersion resource.

**ContentUri**

AWS S3 Uri, local file path, or LayerContent object of the layer code.

- If an AWS S3 Uri or LayerContent object is provided, The AWS S3 object referenced must be a valid ZIP archive that contains the contents of an AWS Lambda layer.
- If a local file path is provided, the template must go through the workflow that includes the `sam deploy` or `sam package` command, in order for the content to be transformed properly.

- **Type:** String | LayerContent (p. 150)
- **Required:** Yes

  **AWS CloudFormation compatibility:** This property is similar to the `Content` property of an AWS::Lambda::LayerVersion resource. The nested Amazon S3 properties are named differently.

**Description**

Description of this layer.
**Type**: String

*Required*: No

AWS CloudFormation compatibility: This property is passed directly to the `Description` property of an AWS::Lambda::LayerVersion resource.

**LayerName**

The name or Amazon Resource Name (ARN) of the layer.

*Type*: String

*Required*: No

*Default*: Resource logical id

AWS CloudFormation compatibility: This property is similar to the `LayerName` property of an AWS::Lambda::LayerVersion resource. If you don’t specify a name, the logical id of the resource will be used as the name.

**LicenseInfo**

Information about the license for this LayerVersion.

*Type*: String

*Required*: No

AWS CloudFormation compatibility: This property is passed directly to the `LicenseInfo` property of an AWS::Lambda::LayerVersion resource.

**RetentionPolicy**

Specifies whether old versions of your LayerVersion are retained or deleted after an update.

*Valid values*: Retain or Delete

*Type*: String

*Required*: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn’t have an AWS CloudFormation equivalent.


### Return Values

**Ref**

When the logical ID of this resource is provided to the `Ref` intrinsic function, it returns the resource ARN of the underlying Lambda LayerVersion.

For more information about using the `Ref` function, see `Ref` in the AWS CloudFormation User Guide.

### Examples

**LayerVersionExample**

Example of a LayerVersion
YAML

Properties:
- LayerName: MyLayer
- Description: Layer description
- ContentUri: 's3://my-bucket/my-layer.zip'
- CompatibleRuntimes:
  - nodejs10.x
  - nodejs12.x
- LicenseInfo: 'Available under the MIT-0 license.'
- RetentionPolicy: Retain

LayerContent

A ZIP archive that contains the contents of an AWS Lambda layer.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

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

Properties

Bucket

The Amazon S3 bucket of the layer archive.

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the S3Bucket property of the AWS::Lambda::LayerVersion Content data type.

Key

The Amazon S3 key of the layer archive.

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the S3Key property of the AWS::Lambda::LayerVersion Content data type.

Version

For versioned objects, the version of the layer archive object to use.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the S3ObjectVersion property of the AWS::Lambda::LayerVersion Content data type.
Examples

LayerContent

Layer Content example

YAML

```
LayerContent:
  Bucket: mybucket-name
  Key: mykey-name
  Version: 121212
```

AWS::Serverless::SimpleTable

Creates a DynamoDB table with a single attribute primary key. It is useful when data only needs to be accessed via a primary key.

To use the more advanced functionality of DynamoDB, use an AWS::DynamoDB::Table resource instead.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

```
Type: AWS::Serverless::SimpleTable
Properties:
  PrimaryKey: PrimaryKeyObject (p. 153)
  ProvisionedThroughput: ProvisionedThroughput
  SSESpecification: SSESpecification
  TableName: String
  Tags: Map
```

Properties

PrimaryKey

Attribute name and type to be used as the table's primary key. If not provided, the primary key will be a String with a value of id.

Note: The value of this property cannot be modified after this resource is created.

Type: PrimaryKeyObject (p. 153)

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

ProvisionedThroughput

Read and write throughput provisioning information.

If ProvisionedThroughput is not specified BillingMode will be specified as PAY_PER_REQUEST.

Type: ProvisionedThroughput
**AWS Serverless Application Model Developer Guide**

**AWS::Serverless::SimpleTable**

**Properties**:

- **TableName**
  - **Type**: String
  - **Required**: No
  - **AWS CloudFormation compatibility**: This property is passed directly to the **TableName** property of an **AWS::DynamoDB::Table** resource.

- **Tags**
  - **Type**: Map
  - **Required**: No
  - **AWS CloudFormation compatibility**: This property is similar to the **Tags** property of an **AWS::DynamoDB::Table** resource. The Tags property in SAM consists of Key:Value pairs; in CloudFormation it consists of a list of Tag objects.

**Return Values**

**Ref**

When the logical ID of this resource is provided to the Ref intrinsic function, it returns the resource name of the underlying DynamoDB table.

For more information about using the Ref function, see **Ref** in the *AWS CloudFormation User Guide*.

**Examples**

**SimpleTableExample**

Example of a SimpleTable

**YAML**

```yaml
Properties:
  TableName: my-table
```
PrimaryKeyIdObject

The object describing the properties of a primary key.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

**YAML**

```yaml
Name: String
Type: String
```

Properties

Name

Attribute name of the primary key.

*Type: String*

*Required: Yes*

**AWS CloudFormation compatibility:** This property is passed directly to the `AttributeName` property of the `AWS::DynamoDB::Table AttributeDefinition` data type.

**Additional notes:** This property is also passed to the `AttributeName` property of an `AWS::DynamoDB::Table KeySchema` data type.

Type

The data type for the primary key.

*Valid values:* String, Number, Binary

*Type: String*

*Required: Yes*

**AWS CloudFormation compatibility:** This property is passed directly to the `AttributeType` property of the `AWS::DynamoDB::Table AttributeDefinition` data type.

Examples

**PrimaryKeyId**

Primary key example.

**YAML**

```yaml
Properties:
  PrimaryKeyId:
    Name: MyPrimaryKey
```
**AWS::Serverless::StateMachine**

Creates an AWS Step Functions state machine, which you can use to orchestrate AWS Lambda functions and other AWS resources to form complex and robust workflows.

For more information about Step Functions, see the AWS Step Functions Developer Guide.

**Note**: To manage AWS SAM templates that contain Step Functions state machines, you must use version 0.52.0 or later of the AWS SAM CLI. To check which version you have, run the command `sam --version`.

**Syntax**

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

**YAML**

```yaml
Type: AWS::Serverless::StateMachine
Properties:
  Definition: Map
  DefinitionSubstitutions: Map
  DefinitionUri: String | S3Location
  Events: EventSource (p. 158)
  Logging: LoggingConfiguration
  Name: String
  PermissionsBoundary: String
  Policies: String | List | Map
  Role: String
  Tags: Map
  Tracing: TracingConfiguration
  Type: String
```

**Properties**

**Definition**

The state machine definition is an object, where the format of the object matches the format of your AWS SAM template file, for example, JSON or YAML. State machine definitions adhere to the Amazon States Language.

For an example of an inline state machine definition, see Examples (p. 157).

You must provide either a **Definition** or a **DefinitionUri**.

**Type: Map**

**Required**: Conditional

**AWS CloudFormation compatibility**: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**DefinitionSubstitutions**

A string-to-string map that specifies the mappings for placeholder variables in the state machine definition. This enables you to inject values obtained at runtime (for example, from intrinsic functions) into the state machine definition.
**AWS Serverless Application Model Developer Guide**

**AWS::Serverless::StateMachine**

- **Type**: Map
- **Required**: No

**AWS CloudFormation compatibility**: This property is similar to the `DefinitionSubstitutions` property of an `AWS::StepFunctions::StateMachine` resource. If any intrinsic functions are specified in an inline state machine definition, AWS SAM adds entries to this property to inject them into the state machine definition.

**DefinitionUri**

The Amazon Simple Storage Service (Amazon S3) URI or local file path of the state machine definition written in the Amazon States Language.

If you provide a local file path, the template must go through the workflow that includes the `sam deploy` or `sam package` command to correctly transform the definition. To do this, you must use version 0.52.0 or later of the AWS SAM CLI.

You must provide either a `Definition` or a `DefinitionUri`.

- **Type**: String | `S3Location`
- **Required**: Conditional

**AWS CloudFormation compatibility**: This property is passed directly to the `DefinitionS3Location` property of an `AWS::StepFunctions::StateMachine` resource.

**Events**

Specifies the events that trigger this state machine. Events consist of a type and a set of properties that depend on the type.

- **Type**: `EventSource` (p. 158)
- **Required**: No

**AWS CloudFormation compatibility**: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**Logging**

Defines which execution history events are logged and where they are logged.

- **Type**: `LoggingConfiguration`
- **Required**: No

**AWS CloudFormation compatibility**: This property is passed directly to the `LoggingConfiguration` property of an `AWS::StepFunctions::StateMachine` resource.

**Name**

The name of the state machine.

- **Type**: String
- **Required**: No

**AWS CloudFormation compatibility**: This property is passed directly to the `StateMachineName` property of an `AWS::StepFunctions::StateMachine` resource.

**PermissionsBoundary**

The ARN of a permissions boundary to use for this state machine's execution role. This property only works if the role is generated for you.
Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the PermissionsBoundary property of an AWS::IAM::Role resource.

Policies

One or more policies that this state machine's execution role needs.

This property accepts a single string or a list of strings. The property can be the name of AWS managed AWS Identity and Access Management (IAM) policies, AWS SAM policy templates, or one or more inline policy documents formatted as a map.

You provide either a Role or Policies.

If the Role property is set, this property is ignored.

Type: String | List | Map

Required: Conditional

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

Role

The ARN of an IAM role to use as this state machine's execution role.

You must provide either a Role or Policies.

Type: String

Required: Conditional

AWS CloudFormation compatibility: This property is passed directly to the RoleArn property of an AWS::StepFunctions::StateMachine resource.

Tags

A string-to-string map that specifies the tags added to the state machine and the corresponding execution role.

Type: Map

Required: No

AWS CloudFormation compatibility: This property is passed directly to the Tags property of an AWS::StepFunctions::StateMachine resource.

Tracing

Selects whether or not AWS X-Ray is enabled for the state machine. For more information about using X-Ray with Step Functions, see AWS X-Ray and Step Functions in the AWS Step Functions Developer Guide.

Type: TracingConfiguration

Required: No

AWS CloudFormation compatibility: This property is passed directly to the TracingConfiguration property of an AWS::StepFunctions::StateMachine resource.
Type

The type of the state machine.

*Valid values:* STANDARD or EXPRESS

*Type:* String

*Required:* No

*Default:* STANDARD

AWS CloudFormation compatibility: This property is passed directly to the `StateMachineType` property of an `AWS::StepFunctions::StateMachine` resource.

Return Values

Ref

When you provide the logical ID of this resource to the Ref intrinsic function, Ref returns the Amazon Resource Name (ARN) of the underlying `AWS::StepFunctions::StateMachine` resource.

For more information about using the `Ref` function, see `Ref` in the AWS CloudFormation User Guide.

Fn::GetAtt

Fn::GetAtt returns a value for a specified attribute of this type. The following are the available attributes and sample return values.

For more information about using `Fn::GetAtt`, see `Fn::GetAtt` in the AWS CloudFormation User Guide.

Name

Returns the name of the state machine, such as `HelloWorld-StateMachine`.

Examples

State Machine Definition File

The following is an example of a state machine defined with a definition file. The `my_state_machine.asl.json` file must be written in the Amazon States Language.

In this example, the `DefinitionSubstitution` entries allow the state machine to include resources that are declared in the AWS SAM template file.

YAML

```
MySampleStateMachine:
  Type: AWS::Serverless::StateMachine
  Properties:
    DefinitionUri: statemachine/my_state_machine.asl.json
    Role: arn:aws:iam::123456123456:role/service-role/my-sample-role
    Tracing:
      Enabled: True
    DefinitionSubstitutions:
      MyFunctionArn: !GetAtt MyFunction.Ar
```
**Inline State Machine Definition**

The following is an example of an inline state machine definition.

In this example, the AWS SAM template file is written in YAML, so the state machine definition is also in YAML. To declare an inline state machine definition in JSON, write your AWS SAM template file in JSON.

**YAML**

```
MySampleStateMachine:
  Type: AWS::Serverless::StateMachine
  Properties:
    Definition:
      StartAt: MyLambdaState
      States:
        MyLambdaState:
          Type: Task
          End: true
          Role: arn:aws:iam::123456123456:role/service-role/my-sample-role
          Tracing:
            Enabled: True
```

**EventSource**

The object describing the source of events which trigger the state machine. Each event consists of a type and a set of properties that depend on that type. For more information about the properties of each event source, see the subtopic corresponding to that type.

**Syntax**

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

**YAML**

```
Properties:
  Schedule (p. 170) | CloudWatchEvent (p. 165) | EventBridgeRule (p. 166) | Api (p. 159)
  Type: String
```

**Properties**

An object describing the properties of this event mapping. The set of properties must conform to the defined Type.

**Type:** Schedule (p. 170) | CloudWatchEvent (p. 165) | EventBridgeRule (p. 166) | Api (p. 159)

**Required:** Yes

**AWS CloudFormation compatibility:** This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**Type**

```
The event type.
```
Valid values: Api, Schedule, CloudWatchEvent, EventBridgeRule

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

Examples

**API**

The following is an example of an event of the API type.

YAML

```yaml
ApiEvent:
  Type: Api
  Properties:
    Method: get
    Path: /group/{user}
    RestApiId:
      Ref: MyApi
```

**Api**

The object describing an Api event source type. If an AWS::Serverless::Api (p. 32) resource is defined, the path and method values must correspond to an operation in the OpenAPI definition of the API.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

```yaml
Auth: ApiStateMachineAuth (p. 160)
Method: String
Path: String
RestApiId: String
```

Properties

Auth

The authorization configuration for this API, path, and method.

Use this property to override the API's DefaultAuthorizer setting for an individual path, when no DefaultAuthorizer is specified, or to override the default ApiKeyRequired setting.

Type: ApiStateMachineAuth (p. 160)

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.
Method

The HTTP method for which this function is invoked.

_Type:_ String

_Required:_ Yes

_AWS CloudFormation compatibility:_ This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

Path

The URI path for which this function is invoked. The value must start with `/`.

_Type:_ String

_Required:_ Yes

_AWS CloudFormation compatibility:_ This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

RestApiId

The identifier of a _RestApi_ resource, which must contain an operation with the given path and method. Typically, this is set to reference an _AWS::Serverless::Api_ (p. 32) resource that is defined in this template.

If you don't define this property, AWS SAM creates a default _AWS::Serverless::Api_ (p. 32) resource using a generated _OpenApi_ document. That resource contains a union of all paths and methods defined by _Api_ events in the same template that do not specify a _RestApiId_.

This property can't reference an _AWS::Serverless::Api_ (p. 32) resource that is defined in another template.

_Type:_ String

_Required:_ No

_AWS CloudFormation compatibility:_ This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

Examples

ApiEvent

The following is an example of an event of the _Api_ type.

_YAML_

```yaml
Events:
  ApiEvent:
    Type: Api
    Properties:
      Path: /path
      Method: get
      RequestParameters:
        - method.request.header.Authorization
```

StateMachineAuth

Configures authorization at the event level, for a specific API, path, and method.
Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

**YAML**

```yaml
ApiKeyRequired: Boolean
AuthorizationScopes: List
Authorizer: String
```

Properties

**ApiKeyRequired**

Requires an API key for this API, path, and method.

*Type: Boolean*

*Required: No*

**AWS CloudFormation compatibility:** This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**AuthorizationScopes**

The authorization scopes to apply to this API, path, and method.

The scopes that you specify will override any scopes applied by the DefaultAuthorizer property if you have specified it.

*Type: List*

*Required: No*

**AWS CloudFormation compatibility:** This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**Authorizer**

The Authorizer for a specific state machine.

If you have specified a global authorizer for the API and want to make this state machine public, override the global authorizer by setting Authorizer to NONE.

*Type: String*

*Required: No*

**AWS CloudFormation compatibility:** This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**ResourcePolicy**

Configure the resource policy for this API and path.

*Type: ResourcePolicyStatement (p. 162)*

*Required: No*

**AWS CloudFormation compatibility:** This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.
Examples

StateMachine-Auth

The following example specifies authorization at the state machine level.

YAML

```
Auth:
  ApiKeyRequired: true
  Authorizer: NONE
```

ResourcePolicyStatement

Configures a resource policy for all methods and paths of an API. For more information about resource policies, see Controlling access to an API with API Gateway resource policies in the API Gateway Developer Guide.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

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

Properties

**AwsAccountBlacklist**

The AWS accounts to block.

*Type:* List

*Required:* No

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**AwsAccountWhitelist**

The AWS accounts to allow. For an example use of this property, see the Examples section at the bottom of this page.

*Type:* List

*Required:* No

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.
CustomStatements

A list of custom resource policy statements to apply to this API. For an example use of this property, see the Examples section at the bottom of this page.

_Type_: List

_Required_: No

_AWS CloudFormation compatibility_: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

IntrinsicVpcBlacklist

The list of virtual private clouds (VPCs) to block, where each VPC is specified as a reference such as a _dynamic reference_ or the _Ref_ _intrinsic function_. For an example use of this property, see the Examples section at the bottom of this page.

_Type_: List

_Required_: No

_AWS CloudFormation compatibility_: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

IntrinsicVpcWhitelist

The list of VPCs to allow, where each VPC is specified as a reference such as a _dynamic reference_ or the _Ref_ _intrinsic function_.

_Type_: List

_Required_: No

_AWS CloudFormation compatibility_: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

IntrinsicVpceBlacklist

The list of VPC endpoints to block, where each VPC endpoint is specified as a reference such as a _dynamic reference_ or the _Ref_ _intrinsic function_.

_Type_: List

_Required_: No

_AWS CloudFormation compatibility_: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

IntrinsicVpceWhitelist

The list of VPC endpoints to allow, where each VPC endpoint is specified as a reference such as a _dynamic reference_ or the _Ref_ _intrinsic function_. For an example use of this property, see the Examples section at the bottom of this page.

_Type_: List

_Required_: No

_AWS CloudFormation compatibility_: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

IpRangeBlacklist

The IP addresses or address ranges to block. For an example use of this property, see the Examples section at the bottom of this page.
**IpRangeWhitelist**

The IP addresses or address ranges to allow.

*Type: List*

*Required: No*

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**SourceVpcBlacklist**

The source VPC or VPC endpoints to block. Source VPC names must start with "vpc-" and source VPC endpoint names must start with "vpce-". For an example use of this property, see the Examples section at the bottom of this page.

*Type: List*

*Required: No*

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**SourceVpcWhitelist**

The source VPC or VPC endpoints to allow. Source VPC names must start with "vpc-" and source VPC endpoint names must start with "vpce-".

*Type: List*

*Required: No*

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

**Examples**

**Resource Policy Example**

The following example blocks two IP addresses and a source VPC, and allows an AWS account.

**YAML**

```
Auth:
  ResourcePolicy:
    CustomStatements: [{
      "Effect": "Allow",
      "Principal": "*",
      "Action": "execute-api:Invoke",
      "Resource": "execute-api:/Prod/GET/pets",
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": "1.2.3.4"
        }
      }
    }]
```
CloudWatchEvent

The object describing a CloudWatchEvent event source type.

AWS Serverless Application Model (AWS SAM) generates an AWS::Events::Rule resource when this event type is set.

**Important Note:** EventBridgeRule (p. 166) is the preferred event source type to use, instead of CloudWatchEvent. EventBridgeRule and CloudWatchEvent use the same underlying service, API, and AWS CloudFormation resources. However, AWS SAM will add support for new features only to EventBridgeRule.

**Syntax**

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

**YAML**

```yaml
EventBusName: String
Input: String
InputPath: String
Pattern: EventPattern
```

**Properties**

**EventBusName**

The event bus to associate with this rule. If you omit this property, AWS SAM uses the default event bus.

*Type: String*

*Required: No*

*Default: Default event bus*

*AWS CloudFormation compatibility:* This property is passed directly to the **EventBusName** property of an AWS::Events::Rule resource.

**Input**

Valid JSON text passed to the target. If you use this property, nothing from the event text itself is passed to the target.
**Type**: String

**Required**: No

**AWS CloudFormation compatibility**: This property is passed directly to the `Input` property of an `AWS::Events::Rule` Target resource.

### InputPath

When you don't want to pass the entire matched event to the target, use the `InputPath` property to describe which part of the event to pass.

**Type**: String

**Required**: No

**AWS CloudFormation compatibility**: This property is passed directly to the `InputPath` property of an `AWS::Events::Rule` Target resource.

### Pattern

Describes which events are routed to the specified target. For more information, see [Events and Event Patterns in EventBridge](https://docs.aws.amazon.com/AmazonEventBridge/latest/userguide/what-is-eventbridge.html) in the *Amazon EventBridge User Guide*.

**Type**: `EventPattern`

**Required**: Yes

**AWS CloudFormation compatibility**: This property is passed directly to the `EventPattern` property of an `AWS::Events::Rule` resource.

### Examples

#### CloudWatchEvent

The following is an example of a CloudWatchEvent event source type.

**YAML**

```yaml
CWEvent:
  Type: CloudWatchEvent
  Properties:
    Input: '{"Key": "Value"}'
    Pattern:
      detail:
        state:
          - terminated
```

#### EventBridgeRule

The object describing an EventBridgeRule event source type, which sets your state machine as the target for an Amazon EventBridge rule. For more information, see [What Is Amazon EventBridge?](https://docs.aws.amazon.com/AmazonEventBridge/latest/userguide/what-is-eventbridge.html) in the *Amazon EventBridge User Guide*.

AWS SAM generates an `AWS::Events::Rule` resource when this event type is set.

### Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.
YAML

```yaml
DeadLetterConfig: DeadLetterConfig (p. 168)
EventBusName: String
Input: String
InputPath: String
Pattern: EventPattern
RetryPolicy: RetryPolicy
```

Properties

DeadLetterConfig

Configure the Amazon Simple Queue Service (Amazon SQS) queue where EventBridge sends events after a failed target invocation. Invocation can fail, for example, when sending an event to a Lambda function that doesn’t exist, or when EventBridge has insufficient permissions to invoke the Lambda function. For more information, see Event retry policy and using dead-letter queues in the Amazon EventBridge User Guide.

**Type:** DeadLetterConfig (p. 168)

**Required:** No

**AWS CloudFormation compatibility:** This property is similar to the DeadLetterConfig property of the AWS::Events::Rule Target data type. The AWS SAM version of this property includes additional subproperties, in case you want AWS SAM to create the dead-letter queue for you.

EventBusName

The event bus to associate with this rule. If you omit this property, AWS SAM uses the default event bus.

**Type:** String

**Required:** No

**Default:** Default event bus

**AWS CloudFormation compatibility:** This property is passed directly to the EventBusName property of an AWS::Events::Rule resource.

Input

Valid JSON text passed to the target. If you use this property, nothing from the event text itself is passed to the target.

**Type:** String

**Required:** No

**AWS CloudFormation compatibility:** This property is passed directly to the Input property of an AWS::Events::Rule Target resource.

InputPath

When you don’t want to pass the entire matched event to the target, use the InputPath property to describe which part of the event to pass.

**Type:** String

**Required:** No
AWS CloudFormation compatibility: This property is passed directly to the InputPath property of an AWS::Events::Rule Target resource.

Pattern

Describes which events are routed to the specified target. For more information, see Events and Event Patterns in EventBridge in the Amazon EventBridge User Guide.

Type: EventPattern

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the EventPattern property of an AWS::Events::Rule resource.

RetryPolicy

A RetryPolicy object that includes information about the retry policy settings. For more information, see Event retry policy and using dead-letter queues in the Amazon EventBridge User Guide.

Type: RetryPolicy

Required: No

AWS CloudFormation compatibility: This property is passed directly to the RetryPolicy property of the AWS::Events::Rule Target data type.

Examples

EventBridgeRule

The following is an example of an EventBridgeRule event source type.

YAML

```yaml
EBRule:
  Type: EventBridgeRule
  Properties:
    Input: '{"Key": "Value"}'
    Pattern:
      detail:
        state:
          - terminated
```

DeadLetterConfig

The object used to specify the Amazon Simple Queue Service (Amazon SQS) queue where EventBridge sends events after a failed target invocation. Invocation can fail, for example, when sending an event to a state machine that doesn’t exist, or insufficient permissions to invoke the state machine. For more information, see Event retry policy and using dead-letter queues in the Amazon EventBridge User Guide.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

```yaml
Arn: String
```
### Properties

**Arn**

The Amazon Resource Name (ARN) of the Amazon SQS queue specified as the target for the dead-letter queue.

**Note:** Specify either the `Type` property or `Arn` property, but not both.

**Type:** String  
**Required:** No  

*AWS CloudFormation compatibility:* This property is passed directly to the `Arn` property of the `AWS::Events::Rule:DeadLetterConfig` data type.

**QueueLogicalId**

The custom name of the dead letter queue that AWS SAM creates if `Type` is specified.

**Note:** If the `Type` property is not set, this property is ignored.

**Type:** String  
**Required:** No  

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn’t have an AWS CloudFormation equivalent.

**Type**

The type of the queue. When this property is set, AWS SAM automatically creates a dead-letter queue and attaches necessary resource-based policy to grant permission to rule resource to send events to the queue.

**Note:** Specify either the `Type` property or `Arn` property, but not both.

**Valid values:** SQS  
**Type:** String  
**Required:** No  

*AWS CloudFormation compatibility:* This property is unique to AWS SAM and doesn’t have an AWS CloudFormation equivalent.

### Examples

**DeadLetterConfig**

DeadLetterConfig

**YAML**

```
DeadLetterConfig:
  Type: SQS
  QueueLogicalId: MyDLQ
```
Schedule

The object describing a Schedule event source type, which sets your state machine as the target of an EventBridge rule that triggers on a schedule. For more information, see What Is Amazon EventBridge? in the Amazon EventBridge User Guide.

AWS Serverless Application Model (AWS SAM) generates an AWS::Events::Rule resource when this event type is set.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

```yaml
DeadLetterConfig: DeadLetterConfig (p. 172)
Description: String
Enabled: Boolean
Input: String
Name: String
RetryPolicy: RetryPolicy
Schedule: String
```

Properties

DeadLetterConfig

Configure the Amazon Simple Queue Service (Amazon SQS) queue where EventBridge sends events after a failed target invocation. Invocation can fail, for example, when sending an event to a Lambda function that doesn't exist, or when EventBridge has insufficient permissions to invoke the Lambda function. For more information, see Event retry policy and using dead-letter queues in the Amazon EventBridge User Guide.

Type: DeadLetterConfig (p. 172)

Required: No

AWS CloudFormation compatibility: This property is similar to the DeadLetterConfig property of the AWS::Events::Rule Target data type. The AWS SAM version of this property includes additional subproperties, in case you want AWS SAM to create the dead-letter queue for you.

Description

A description of the rule.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the Description property of an AWS::Events::Rule resource.

Enabled

Indicates whether the rule is enabled.

To disable the rule, set this property to False.

Type: Boolean

Required: No
**AWS CloudFormation compatibility**: This property is similar to the `State` property of an `AWS::Events::Rule` resource. If this property is set to `True` then AWS SAM passes `ENABLED`, otherwise it passes `DISABLED`.

**Input**

Valid JSON text passed to the target. If you use this property, nothing from the event text itself is passed to the target.

*Type*: String  
*Required*: No

**AWS CloudFormation compatibility**: This property is passed directly to the `Target` property of an `AWS::Events::Rule` Target resource.

**Name**

The name of the rule. If you don’t specify a name, AWS CloudFormation generates a unique physical ID and uses that ID for the rule name.

*Type*: String  
*Required*: No

**AWS CloudFormation compatibility**: This property is passed directly to the `Name` property of an `AWS::Events::Rule` resource.

**RetryPolicy**

A `RetryPolicy` object that includes information about the retry policy settings. For more information, see [Event retry policy and using dead-letter queues](https://docs.aws.amazon.com/AmazonEventBridge/latest/userguide/EventBridge-DeadLetterQueue.html) in the Amazon EventBridge User Guide.

*Type*: `RetryPolicy`  
*Required*: No

**AWS CloudFormation compatibility**: This property is passed directly to the `RetryPolicy` property of the `AWS::Events::Rule` Target data type.

**Schedule**

The scheduling expression that determines when and how often the rule runs. For more information, see [Schedule Expressions for Rules](https://docs.aws.amazon.com/AmazonEventBridge/latest/userguide/EventBridge-Schedule-Expressions.html).

*Type*: String  
*Required*: Yes

**AWS CloudFormation compatibility**: This property is passed directly to the `ScheduleExpression` property of an `AWS::Events::Rule` resource.

**Examples**

**CloudWatch Schedule Event**

CloudWatch Schedule Event Example

**YAML**

```
CWSchedule: 
```
Type: Schedule
Properties:
  Schedule: 'rate(1 minute)'
  Name: TestSchedule
  Description: test schedule
  Enabled: False

DeadLetterConfig

The object used to specify the Amazon Simple Queue Service (Amazon SQS) queue where EventBridge sends events after a failed target invocation. Invocation can fail, for example, when sending an event to a state machine that doesn’t exist, or insufficient permissions to invoke the state machine. For more information, see Event retry policy and using dead-letter queues in the Amazon EventBridge User Guide.

Syntax

To declare this entity in your AWS Serverless Application Model (AWS SAM) template, use the following syntax.

YAML

```yaml
Arn: String
QueueLogicalId: String
Type: String
```

Properties

Arn

The Amazon Resource Name (ARN) of the Amazon SQS queue specified as the target for the dead-letter queue.

Note: Specify either the Type property or Arn property, but not both.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the Arn property of the AWS::Events::Rule DeadLetterConfig data type.

QueueLogicalId

The custom name of the dead letter queue that AWS SAM creates if Type is specified.

Note: If the Type property is not set, this property is ignored.

Type: String

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

Type

The type of the queue. When this property is set, AWS SAM automatically creates a dead-letter queue and attaches necessary resource-based policy to grant permission to rule resource to send events to the queue.

Note: Specify either the Type property or Arn property, but not both.
Resource attributes

Resource attributes are attributes that you can add to a resource to control additional behaviors and relationships. For more information about resource attributes, see Resource Attribute Reference in the AWS CloudFormation User Guide.

AWS SAM resources support a subset of resource attributes that are supported by AWS CloudFormation resources. To see which AWS SAM resources support which resource attributes, see the following table.

<table>
<thead>
<tr>
<th>Resource Type</th>
<th>CreationPolicy</th>
<th>DeletionPolicy</th>
<th>DependsOn</th>
<th>Metadata</th>
<th>UpdatePolicy</th>
<th>UpdateReplacePolicy</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS::Serverless::Api (p. 32)</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>AWS::Serverless::Application (p. 63)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AWS::Serverless::Function (p. 67)</td>
<td>✓</td>
<td>✓</td>
<td>✓ (p. 204)*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AWS::Serverless::HttpApi (p. 128)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AWS::Serverless::LayerVersion (p. 147)</td>
<td>✓</td>
<td>✓</td>
<td>✓ (p. 202)**</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AWS::Serverless::SimpleTable (p. 151)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AWS::Serverless::StateMachine (p. 154)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes:

* For more information about using the Metadata resource attribute with the AWS::Serverless::Function resource type, see Building custom runtimes (p. 204).
**For more information about using the Metadata resource attribute with the AWS::Serverless::LayerVersion resource type, see Building layers (p. 202).**

Intrinsic functions

Intrinsic functions are built-in functions that enable you to assign values to properties that are only available at runtime. For more information about intrinsic functions, see Intrinsic Function Reference in the AWS CloudFormation User Guide.

Generated AWS CloudFormation resources

When AWS Serverless Application Model (AWS SAM) processes your AWS SAM template file, it generates AWS CloudFormation resources. The set of AWS CloudFormation resources that AWS SAM generates differs depending on the scenarios that you specify. A *scenario* is the combination of AWS SAM resources and properties specified in your template file. You can reference the generated AWS CloudFormation resources elsewhere within your template file, similar to how you reference resources that you declare explicitly in your template file.

For example, if you specify an AWS::Serverless::Function resource in your AWS SAM template file, AWS SAM always generates an AWS::Lambda::Function base resource. If you also specify the optional AutoPublishAlias property, AWS SAM additionally generates AWS::Lambda::Alias and AWS::Lambda::Version resources.

This section lists the scenarios and the AWS CloudFormation resources that they generate, and shows how to reference the generated AWS CloudFormation resources in your AWS SAM template file.

Topics

- Referencing generated AWS CloudFormation resources (p. 174)
- Generated AWS CloudFormation resource reference (p. 175)
- AWS CloudFormation resources generated when AWS::Serverless::Api is specified (p. 176)
- AWS CloudFormation resources generated when AWS::Serverless::Function is specified (p. 177)
- AWS CloudFormation resources generated when AWS::Serverless::HttpApi is specified (p. 179)

Referencing generated AWS CloudFormation resources

You have two options for referencing generated AWS CloudFormation resources within your AWS SAM template file, by LogicalId or by referenceable property.

Referencing generated AWS CloudFormation resources by LogicalId

The AWS CloudFormation resources that AWS SAM generates each have a LogicalId, which is an alphanumeric (A-Z, a-z, 0-9) identifier that is unique within a template file. AWS SAM uses the LogicalIds of the AWS SAM resources in your template file to construct the LogicalIds of the AWS CloudFormation resources it generates. You can use the LogicalId of a generated AWS CloudFormation resource to access properties of that resource within your template file, just like you would for an AWS CloudFormation resource that you have explicitly declared. For more information about LogicalIds in AWS CloudFormation and AWS SAM templates, see Resources in the AWS CloudFormation User Guide.
Note
The LogicalIds of some generated resources include a unique hash value to avoid namespace clashes. The LogicalIds of these resources are derived when the stack is created. You can retrieve them only after the stack has been created using the AWS Management Console, AWS CLI, or one of the AWS SDKs. We don’t recommend referencing these resources by LogicalId because the hash values might change.

Referencing generated AWS CloudFormation resources by referenceable property

For some generated resources, AWS SAM provides a referenceable property of the AWS SAM resource. You can use this property to reference a generated AWS CloudFormation resource and its properties within your AWS SAM template file.

Note
Not all generated AWS CloudFormation resources have referenceable properties. For those resources, you must use the LogicalId.

Generated AWS CloudFormation resource reference

The following table summarizes the AWS SAM resources and properties that make up the scenarios that generate AWS CloudFormation resources.

The topics in the AWS SAM Resources column provide details about the base resources that are generated when you specify the AWS SAM resource. The topics in the Scenarios column provide details about the additional resources generated for that scenario.

<table>
<thead>
<tr>
<th>AWS SAM Resources</th>
<th>Scenarios</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS::Serverless::Api (p. 176)</td>
<td>• DomainName property is specified (p. 177)</td>
</tr>
<tr>
<td></td>
<td>• UsagePlan property is specified (p. 177)</td>
</tr>
<tr>
<td>AWS::Serverless::Function (p. 177)</td>
<td>• AutoPublishAlias property is specified (p. 178)</td>
</tr>
<tr>
<td></td>
<td>• Role property is not specified (p. 178)</td>
</tr>
<tr>
<td></td>
<td>• OnSuccess (or OnFailure) property is specified for Amazon SNS events (p. 178)</td>
</tr>
<tr>
<td></td>
<td>• OnSuccess (or OnFailure) property is specified for Amazon SQS events (p. 179)</td>
</tr>
</tbody>
</table>
AWS CloudFormation resources generated when AWS::Serverless::Api is specified

When an AWS::Serverless::Api is specified, AWS Serverless Application Model (AWS SAM) always generates the following AWS CloudFormation resources: an AWS::ApiGateway::RestApi, an AWS::ApiGateway::Stage, and an AWS::ApiGateway::Deployment.

AWS::ApiGateway::RestApi

LogicalId: <api#LogicalId>

Referenceable property: N/A (you must use the LogicalId to reference this AWS CloudFormation resource)

AWS::ApiGateway::Stage

LogicalId: <api#LogicalId><stage#name>Stage

<stage#name> is the string that the StageName property is set to. For example, if you set StageName to Gamma, the LogicalId is MyRestApiGammaStage.

Referenceable property: <api#LogicalId>.Stage

AWS::ApiGateway::Deployment

LogicalId: <api#LogicalId>Deployment<sha>

<sha> is a unique hash value that is generated when the stack is created. For example, MyRestApiDeployment926eeb5ff1.

Referenceable property: <api#LogicalId>.Deployment

In addition to these AWS CloudFormation resources, when AWS::Serverless::Api is specified, AWS SAM generates additional AWS CloudFormation resources for the following scenarios.

Scenarios

- DomainName property is specified (p. 177)
- UsagePlan property is specified (p. 177)
DomainName property is specified

When the DomainName property of the Domain property of an AWS::Serverless::Api is specified, AWS SAM generates the AWS::ApiGateway::DomainName AWS CloudFormation resource.

AWS::ApiGateway::DomainName

LogicalId: ApiGatewayDomainName<sha>

<sha> is a unique hash value that is generated when the stack is created. For example: ApiGatewayDomainName926eeb5ff1.

Referenceable property: <api#LogicalId>.DomainName

UsagePlan property is specified

When the UsagePlan property of the Auth property of an AWS::Serverless::Api is specified, AWS SAM generates the following AWS CloudFormation resources: AWS::ApiGateway::UsagePlan, AWS::ApiGateway::UsagePlanKey, and AWS::ApiGateway::ApiKey.

AWS::ApiGateway::UsagePlan

LogicalId: <api#LogicalId>UsagePlan

Referenceable property: <api#LogicalId>.UsagePlan

AWS::ApiGateway::UsagePlanKey

LogicalId: <api#LogicalId>UsagePlanKey

Referenceable property: <api#LogicalId>.UsagePlanKey

AWS::ApiGateway::ApiKey

LogicalId: <api#LogicalId>ApiKey

Referenceable property: <api#LogicalId>.ApiKey

AWS CloudFormation resources generated when AWS::Serverless::Function is specified

When an AWS::Serverless::Function is specified, AWS Serverless Application Model (AWS SAM) creates the AWS::Lambda::Function AWS CloudFormation resource.

AWS::Lambda::Function

LogicalId: <function#LogicalId>

Referenceable property: N/A (you must use the LogicalId to reference this AWS CloudFormation resource)

In addition to this AWS CloudFormation resource, when AWS::Serverless::Function is specified, AWS SAM also generates AWS CloudFormation resources for the following scenarios.

Scenarios
• AutoPublishAlias property is specified (p. 178)
• Role property is not specified (p. 178)
• OnSuccess (or OnFailure) property is specified for Amazon SNS events (p. 178)
• OnSuccess (or OnFailure) property is specified for Amazon SQS events (p. 179)

AutoPublishAlias property is specified

When the AutoPublishAlias property of an AWS::Serverless::Function is specified, AWS SAM generates the following AWS CloudFormation resources: AWS::Lambda::Alias and AWS::Lambda::Version.

AWS::Lambda::Alias

LogicalId: <function#LogicalId>Alias<alias­name>

<alias­name> is the string that AutoPublishAlias is set to. For example, if you set AutoPublishAlias to live, the LogicalId is: MyFunctionAliaslive.

Referenceable property: <function#LogicalId>.Alias

AWS::Lambda::Version

LogicalId: <function#LogicalId>Version<sha>

<sha> is a unique hash value that is generated when the stack is created. For example, MyFunctionVersion926eeb5ff1.

Referenceable property: <function#LogicalId>.Version

Role property is not specified

When the Role property of an AWS::Serverless::Function is not specified, AWS SAM generates the AWS::IAM::Role AWS CloudFormation resource.

AWS::IAM::Role

LogicalId: <function#LogicalId>Role

Referenceable property: N/A (you must use the LogicalId to reference this AWS CloudFormation resource)

OnSuccess (or OnFailure) property is specified for Amazon SNS events

When the OnSuccess (or OnFailure) property of the DestinationConfig property of the EventInvokeConfig property of an AWS::Serverless::Function is specified, and the destination type is SNS but the destination ARN is not specified, AWS SAM generates the following AWS CloudFormation resources: AWS::Lambda::EventInvokeConfig and AWS::SNS::Topic.

AWS::Lambda::EventInvokeConfig

LogicalId: <function#LogicalId>EventInvokeConfig

Referenceable property: N/A (you must use the LogicalId to reference this AWS CloudFormation resource)
AWS::SNS::Topic

LogicalId: <function-LogicalId>OnSuccessTopic (or <function-LogicalId>OnFailureTopic)

Referenceable property: <function-LogicalId>.DestinationTopic

If both onSuccess and onFailure are specified for an Amazon SNS event, to distinguish between the generated resources, you must use the LogicalId.

OnSuccess (or OnFailure) property is specified for Amazon SQS events

When the OnSuccess (or OnFailure) property of the DestinationConfig property of the EventInvokeConfig property of an AWS::Serverless::Function is specified, and the destination type is SQS but the destination ARN is not specified, AWS SAM generates the following AWS CloudFormation resources: AWS::Lambda::EventInvokeConfig and AWS::SQS::Queue.

AWS::Lambda::EventInvokeConfig

LogicalId: <function-LogicalId>EventInvokeConfig

Referenceable property: N/A (you must use the LogicalId to reference this AWS CloudFormation resource)

AWS::SQS::Queue

LogicalId: <function-LogicalId>OnSuccessQueue (or <function-LogicalId>OnFailureQueue)

Referenceable property: <function-LogicalId>.DestinationQueue

If both onSuccess and onFailure are specified for an Amazon SQS event, to distinguish between the generated resources, you must use the LogicalId.

AWS CloudFormation resources generated when AWS::Serverless::HttpApi is specified

When an AWS::Serverless::HttpApi is specified, AWS Serverless Application Model (AWS SAM) generates the AWS::ApiGatewayV2::Api AWS CloudFormation resource.

AWS::ApiGatewayV2::Api

LogicalId: <httpapi-LogicalId>

Referenceable property: N/A (you must use the LogicalId to reference this AWS CloudFormation resource)

In addition to this AWS CloudFormation resource, when AWS::Serverless::HttpApi is specified, AWS SAM also generates AWS CloudFormation resources for the following scenarios:

Scenarios

- StageName property is specified (p. 180)
- StageName property is not specified (p. 180)
- DomainName property is specified (p. 180)
StageName property is specified

When the StageName property of an AWS::Serverless::HttpApi is specified, AWS SAM generates the AWS::ApiGatewayV2::Stage AWS CloudFormation resource.

**AWS::ApiGatewayV2::Stage**

- LogicalId: `<httpapi#LogicalId><stage#name>Stage`

  `<stage#name>` is the string that the StageName property is set to. For example, if you set StageName to Gamma, the LogicalId is: *MyHttpApiGammaStage*.

  *Referenceable property: `<httpapi#LogicalId>.Stage`

StageName property is not specified

When the StageName property of an AWS::Serverless::HttpApi is not specified, AWS SAM generates the AWS::ApiGatewayV2::Stage AWS CloudFormation resource.

**AWS::ApiGatewayV2::Stage**

- LogicalId: `<httpapi#LogicalId>ApiGatewayDefaultStage`

  *Referenceable property: `<httpapi#LogicalId>.Stage`

DomainName property is specified

When the DomainName property of the Domain property of an AWS::Serverless::HttpApi is specified, AWS SAM generates the AWS::ApiGatewayV2::DomainName AWS CloudFormation resource.

**AWS::ApiGatewayV2::DomainName**

- LogicalId: `ApiGatewayDomainNameV2<sha>`

  `<sha>` is a unique hash value that is generated when the stack is created. For example, `ApiGatewayDomainNameV2926eeb5ff1`.

  *Referenceable property: `<httpapi#LogicalId>.DomainName`

API Gateway extensions

API Gateway extensions are extensions to the OpenAPI specification that support the AWS-specific authorization and API Gateway-specific API integrations. For more information about API Gateway extensions, see [API Gateway Extensions to OpenAPI](#).

AWS SAM supports a subset of API Gateway extensions. To see which API Gateway extensions are supported by AWS SAM, see the following table.

<table>
<thead>
<tr>
<th>API Gateway Extension</th>
<th>Supported by AWS SAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>x-amazon-apigateway-any-method Object</td>
<td>Yes</td>
</tr>
<tr>
<td>x-amazon-apigateway-api-key-source Property</td>
<td>No</td>
</tr>
<tr>
<td>Field</td>
<td>Value</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>x-amazon-apigateway-auth Object</td>
<td>Yes</td>
</tr>
<tr>
<td>x-amazon-apigateway-authorizer Object</td>
<td>Yes</td>
</tr>
<tr>
<td>x-amazon-apigateway-authtype Property</td>
<td>Yes</td>
</tr>
<tr>
<td>x-amazon-apigateway-binary-media-types Property</td>
<td>Yes</td>
</tr>
<tr>
<td>x-amazon-apigateway-documentation Object</td>
<td>No</td>
</tr>
<tr>
<td>x-amazon-apigateway-endpoint-configuration Object</td>
<td>No</td>
</tr>
<tr>
<td>x-amazon-apigateway-gateway-responses Object</td>
<td>Yes</td>
</tr>
<tr>
<td>x-amazon-apigateway-gateway-responses.gatewayResponse Object</td>
<td>Yes</td>
</tr>
<tr>
<td>x-amazon-apigateway-gateway-responses.responseParameters Object</td>
<td>Yes</td>
</tr>
<tr>
<td>x-amazon-apigateway-gateway-responses.responseTemplates Object</td>
<td>Yes</td>
</tr>
<tr>
<td>x-amazon-apigateway-integration Object</td>
<td>Yes</td>
</tr>
<tr>
<td>x-amazon-apigateway-integration.requestTemplates Object</td>
<td>Yes</td>
</tr>
<tr>
<td>x-amazon-apigateway-integration.requestParameters Object</td>
<td>No</td>
</tr>
<tr>
<td>x-amazon-apigateway-integration.responses Object</td>
<td>Yes</td>
</tr>
<tr>
<td>x-amazon-apigateway-integration.response Object</td>
<td>Yes</td>
</tr>
<tr>
<td>x-amazon-apigateway-integration.responseTemplates Object</td>
<td>Yes</td>
</tr>
<tr>
<td>x-amazon-apigateway-integration.responseParameters Object</td>
<td>Yes</td>
</tr>
<tr>
<td>x-amazon-apigateway-request-validator Property</td>
<td>No</td>
</tr>
<tr>
<td>x-amazon-apigateway-request-validators Object</td>
<td>No</td>
</tr>
<tr>
<td>x-amazon-apigateway-request-validators.requestValidator Object</td>
<td>No</td>
</tr>
</tbody>
</table>
Authoring serverless applications

When you author a serverless application using AWS SAM, you construct an AWS SAM template to declare and configure the components of your application.

This section contains topics about validating your AWS SAM template and building your application with dependencies. It also contains topics about using AWS SAM for certain use cases such as working with Lambda layers, using nested applications, controlling access to API Gateway APIs, orchestrating AWS resources with Step Functions, and code signing your applications.

Topics
- Validating AWS SAM template files (p. 182)
- Working with layers (p. 182)
- Using nested applications (p. 184)
- Controlling access to API Gateway APIs (p. 186)
- Orchestrating AWS resources with AWS Step Functions (p. 195)
- Configuring code signing for AWS SAM applications (p. 196)

Validating AWS SAM template files

Validate your templates with `sam validate (p. 258)`. Currently, this command validates that the template provided is valid JSON / YAML. As with most AWS SAM CLI commands, it looks for a `template.[yaml|yml]` file in your current working directory by default. You can specify a different template file/location with the `-t` or `--template` option.

Example:

```
  sam validate
  <path-to-file>/template.yml is a valid SAM Template
```

**Note**
The `sam validate` command requires AWS credentials to be configured. For more information, see Configuration and Credential Files.

Working with layers

Using AWS SAM, you can include layers in your serverless applications. For more information about layers, see AWS Lambda layers in the AWS Lambda Developer Guide.

This topic provides information about the following:
- Including layers in your application
- How layers are cached locally

For information about building custom layers, see Building layers (p. 202).
Including layers in your application

To include layers in your application, use the Layers property of the AWS::Serverless::Function resource type.

Following is an example AWS SAM template with a Lambda function that includes a layer:

```
ServerlessFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: .
    Handler: my_handler
    Runtime: Python3.7
    Layers:
      - <LayerVersion ARN>
```

How layers are cached locally

When you invoke your function using one of the `sam local` commands, the layers package of your function is downloaded and cached on your local host.

The following table shows the default cache directory locations for different operating systems.

<table>
<thead>
<tr>
<th>OS</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows 7</td>
<td>C:\Users&lt;user&gt;\AppData\Roaming\AWS SAM</td>
</tr>
<tr>
<td>Windows 8</td>
<td>C:\Users&lt;user&gt;\AppData\Roaming\AWS SAM</td>
</tr>
<tr>
<td>Windows 10</td>
<td>C:\Users&lt;user&gt;\AppData\Roaming\AWS SAM</td>
</tr>
<tr>
<td>macOS</td>
<td>~/.aws-sam/layers-pkg</td>
</tr>
<tr>
<td>Unix</td>
<td>~/.aws-sam/layers-pkg</td>
</tr>
</tbody>
</table>

After the package is cached, the AWS SAM CLI overlays the layers onto a Docker image that’s used to invoke your function. The AWS SAM CLI generates the names of the images it builds, as well as the LayerVersions that are held in the cache. You can find more details about the schema in the following sections.

To inspect the overlaid layers, execute the following command to start a bash session in the image that you want to inspect:

```
docker run -it --entrypoint=/bin/bash samcli/lambda:<Tag following the schema outlined in Docker Image Tag Schema> -i
```

Layer Caching Directory name schema

Given a LayerVersionArn that’s defined in your template, the AWS SAM CLI extracts the LayerName and Version from the ARN. It creates a directory to place the layer contents in named `LayerName-Version-<first 10 characters of sha256 of ARN>`.  

Example:

```
Directory name = myLayer-1-926eeb5ff1
```
Docker Images tag schema

To compute the unique layers hash, combine all unique layer names with a delimiter of '-', take the SHA256 hash, and then take the first 10 characters.

Example:

```yaml
ServerlessFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: .
    Handler: my_handler
    Runtime: Python3.7
    Layers:
```

Unique names are computed the same as the Layer Caching Directory name schema:

```yaml
```

To compute the unique layers hash, combine all unique layer names with a delimiter of '-', take the sha256 hash, and then take the first 25 characters:

```yaml
myLayer-1-926eeb5ff1-mySecondLayer-1-6bc1022bdf = 2dd7ac5fffb30d515926aefffd
```

Then combine this value with the function's runtime, with a delimiter of '-':

```yaml
python3.7-2dd7ac5fffb30d515926aefffd
```

Using nested applications

A serverless application can include one or more nested applications. You can deploy a nested application as a stand-alone artifact or as a component of a larger application.

As serverless architectures grow, common patterns emerge in which the same components are defined in multiple application templates. You can now separate out common patterns as dedicated applications, and then nest them as part of new or existing application templates. With nested applications, you can stay more focused on the business logic that’s unique to your application.

To define a nested application in your serverless application, use the `AWS::Serverless::Application` resource type.

You can define nested applications from the following two sources:

- An AWS Serverless Application Repository application – You can define nested applications by using applications that are available to your account in the AWS Serverless Application Repository. These can be private applications in your account, applications that are privately shared with your account, or applications that are publicly shared in the AWS Serverless Application Repository. For more information about the different deployment permissions levels, see Application Deployment Permissions and Publishing Applications in the AWS Serverless Application Repository Developer Guide.

- A local application – You can define nested applications by using applications that are stored on your local file system.
See the following sections for details on how to use AWS SAM to define both of these types of nested applications in your serverless application.

**Note**

The maximum number of applications that can be nested in a serverless application is 200.

The maximum number of parameters a nested application can have is 60.

### Defining a nested application from the AWS Serverless Application Repository

You can define nested applications by using applications that are available in the AWS Serverless Application Repository. You can also store and distribute applications that contain nested applications using the AWS Serverless Application Repository. To review details of a nested application in the AWS Serverless Application Repository, you can use the AWS SDK, the AWS CLI, or the Lambda console.

To define an application that's hosted in the AWS Serverless Application Repository in your serverless application's AWS SAM template, use the **Copy as SAM Resource** button on the detail page of every AWS Serverless Application Repository application. To do this, follow these steps:

1. Make sure that you're signed in to the AWS Management Console.
2. Find the application that you want to nest in the AWS Serverless Application Repository by using the steps in the Browsing, Searching, and Deploying Applications section of the AWS Serverless Application Repository Developer Guide.
3. Choose the **Copy as SAM Resource** button. The SAM template section for the application that you're viewing is now in your clipboard.
4. Paste the SAM template section into the **Resources:** section of the SAM template file for the application that you want to nest in this application.

The following is an example SAM template section for a nested application that's hosted in the AWS Serverless Application Repository:

```yaml
Transform: AWS::Serverless-2016-10-31
Resources:
  applicationaliasname:
    Type: AWS::Serverless::Application
    Properties:
      Location:
        ApplicationId: arn:aws:serverlessrepo:us-east-1:123456789012:applications/application-alias-name
        SemanticVersion: 1.0.0
      Parameters:
        # Optional parameter that can have default value overridden
        ParameterName1: 15 # Uncomment to override default value
        # Required parameter that needs value to be provided
        ParameterName2: YOUR_VALUE
```

If there are no required parameter settings, you can omit the **Parameters:** section of the template.

**Important**

Applications that contain nested applications hosted in the AWS Serverless Application Repository 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, you aren't able to deploy the parent application. For more information about permissions to deploy
Defining a nested application from the local file system

You can define nested applications by using applications that are stored on your local file system. You do this by specifying the path to the AWS SAM template file that's stored on your local file system.

The following is an example SAM template section for a nested local application:

```yaml
Transform: AWS::Serverless-2016-10-31

Resources:
  applicationName:
    Type: AWS::Serverless::Application
    Properties:
      Location: ../my-other-app/template.yaml
      Parameters:
        # Optional parameter that can have default value overridden
        ParameterName1: 15 # Uncomment to override default value
        # Required parameter that needs value to be provided
        ParameterName2: YOUR_VALUE
```

If there are no parameter settings, you can omit the `Parameters:` section of the template.

Deploying nested applications

You can deploy your nested application by using the AWS SAM CLI command `sam deploy`. For more details, see Deploying serverless applications (p. 219).

Note
When you deploy an application that contains nested applications, you must acknowledge that. You do this by passing `CAPABILITY_AUTO_EXPAND` to the `CreateCloudFormationChangeSet API`, or using the `aws serverlessrepo create-cloud-formation-change-set` AWS CLI command.

For more information about acknowledging nested applications, see Acknowledging IAM Roles, Resource Policies, and Nested Applications when Deploying Applications in the AWS Serverless Application Repository Developer Guide.

Controlling access to API Gateway APIs

To control who can access your Amazon API Gateway APIs, you can enable authorization within your AWS SAM template.

AWS SAM supports several mechanisms for controlling access to your API Gateway APIs. The set of supported mechanisms differs between `AWS::Serverless::HttpApi` and `AWS::Serverless::Api` resource types.

The following table summarizes the mechanisms that each resource type supports.

<table>
<thead>
<tr>
<th>Mechanisms for controlling access</th>
<th>AWS::Serverless::HttpApi</th>
<th>AWS::Serverless::Api</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lambda authorizers</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Controlling access to APIs

<table>
<thead>
<tr>
<th>Mechanisms for controlling access</th>
<th>AWS::Serverless::HttpApi</th>
<th>AWS::Serverless::Api</th>
</tr>
</thead>
<tbody>
<tr>
<td>IAM permissions</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Amazon Cognito user pools</td>
<td>✓ *</td>
<td>✓</td>
</tr>
<tr>
<td>API keys</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Resource policies</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>OAuth 2.0/JWT authorizers</td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

* You can use Amazon Cognito as a JSON Web Token (JWT) issuer with the AWS::Serverless::HttpApi resource type.

- **Lambda authorizers** – A Lambda authorizer (formerly known as a custom authorizer) is a Lambda function that you provide to control access to your API. When your API is called, this Lambda function is invoked with a request context or an authorization token that the client application provides. The Lambda function responds whether the caller is authorized to perform the requested operation.

  Both the AWS::Serverless::HttpApi and AWS::Serverless::Api resource types support Lambda authorizers.

  For more information about Lambda authorizers with AWS::Serverless::HttpApi, see Working with AWS Lambda authorizers for HTTP APIs in the API Gateway Developer Guide. For more information about Lambda authorizers with AWS::Serverless::Api, see Use API Gateway Lambda authorizers in the API Gateway Developer Guide.

  For examples of Lambda authorizers for either resource type, see Lambda authorizer examples (p. 189).

- **IAM permissions** – You can control who can invoke your API using AWS Identity and Access Management (IAM) permissions. Users calling your API must be authenticated with IAM credentials. Calls to your API succeed only if there is an IAM policy attached to the IAM user that represents the API caller, an IAM group that contains the user, or an IAM role that the user assumes.

  Only the AWS::Serverless::Api resource type supports IAM permissions.

  For more information, see Control access to an API with IAM permissions in the API Gateway Developer Guide. For an example, see IAM permission example (p. 191).

- **Amazon Cognito user pools** – Amazon Cognito user pools are user directories in Amazon Cognito. A client of your API must first sign in a user to the user pool and obtain an identity or access token for the user. Then the client calls your API with one of the returned tokens. The API call succeeds only if the required token is valid.

  The AWS::Serverless::Api resource type supports Amazon Cognito user pools. The AWS::Serverless::HttpApi resource type supports the use of Amazon Cognito as a JWT issuer.

  For more information, see Control access to a REST API using Amazon Cognito user pools as authorizer in the API Gateway Developer Guide. For an example, see Amazon Cognito user pool example (p. 191).

- **API keys** – API keys are alphanumeric string values that you distribute to application developer customers to grant access to your API.

  Only the AWS::Serverless::Api resource type supports API keys.

  For more information about API keys, see Creating and using usage plans with API keys in the API Gateway Developer Guide. For an example of API keys, see API key example (p. 192).
Choosing a mechanism to control access

The mechanism that you choose to use for controlling access to your API Gateway APIs depends on a few factors. For example, if you have a greenfield project without either authorization or access control set up, then Amazon Cognito user pools might be your best option. This is because when you set up user pools, you also automatically set up both authentication and access control.

However, if your application already has authentication set up, then using Lambda authorizers might be your best option. This is because you can call your existing authentication service and return a policy document based on the response. Also, if your application requires custom authentication or access control logic that user pools don't support, then Lambda authorizers might be your best option.

When you’ve chosen which mechanism to use, see the corresponding section in Examples (p. 188) for how to use AWS SAM to configure your application to use that mechanism.

Customizing error responses

You can use AWS SAM to customize the content of some API Gateway error responses. Only the AWS::Serverless::Api resource type supports customized API Gateway responses.

For more information about API Gateway responses, see Gateway responses in API Gateway in the API Gateway Developer Guide. For an example of customized responses, see Customized response example (p. 194).

Examples

- Lambda authorizer examples (p. 189)
- IAM permission example (p. 191)
- Amazon Cognito user pool example (p. 191)
- API key example (p. 192)
- Resource policy example (p. 193)
- OAuth 2.0/JWT authorizer example (p. 193)
- Customized response example (p. 194)
Lambda authorizer examples

The AWS::Serverless::Api resource type supports two types of Lambda authorizers: TOKEN authorizers and REQUEST authorizers. The AWS::Serverless::HttpApi resource type supports only REQUEST authorizers. The following are examples of each type.

Lambda TOKEN authorizer example (AWS::Serverless::Api)

You can control access to your APIs by defining a Lambda TOKEN authorizer within your AWS SAM template. To do this, you use the ApiAuth (p. 40) data type.

The following is an example AWS SAM template section for a Lambda TOKEN authorizer:

```
Resources:
  MyApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      Auth:
        DefaultAuthorizer: MyLambdaTokenAuthorizer
        Authorizers:
          MyLambdaTokenAuthorizer:
            FunctionArn: !GetAtt MyAuthFunction.Arn

  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: ./src
      Handler: index.handler
      Runtime: nodejs12.x
      Events:
        GetRoot:
          Type: Api
          Properties:
            RestApiId: !Ref MyApi
            Path: /
            Method: get

  MyAuthFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: ./src
      Handler: authorizer.handler
      Runtime: nodejs12.x
```

For more information about Lambda authorizers, see Use API Gateway Lambda authorizers in the API Gateway Developer Guide.

Lambda REQUEST authorizer example (AWS::Serverless::Api)

You can control access to your APIs by defining a Lambda REQUEST authorizer within your AWS SAM template. To do this, you use the ApiAuth (p. 40) data type.

The following is an example AWS SAM template section for a Lambda REQUEST authorizer:

```
Resources:
  MyApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
```
Lambda authorizer examples

Auth:
  DefaultAuthorizer: MyLambdaRequestAuthorizer
  Authorizers:
    MyLambdaRequestAuthorizer:
      FunctionPayloadType: REQUEST
      FunctionArn: !GetAtt MyAuthFunction.Arn
      Identity:
        QueryStrings:
        - auth

MyFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: ./src
    Handler: index.handler
    Runtime: nodejs12.x
  Events:
    GetRoot:
      Type: Api
      Properties:
        RestApiId: !Ref MyApi
        Path: /
        Method: get

MyAuthFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: ./src
    Handler: authorizer.handler
    Runtime: nodejs12.x

For more information about Lambda authorizers, see Use API Gateway Lambda authorizers in the API Gateway Developer Guide.

Lambda authorizer example (AWS::Serverless::HttpApi)

You can control access to your HTTP APIs by defining a Lambda authorizer within your AWS SAM template. To do this, you use the HttpApiAuth (p. 135) data type.

The following is an example AWS SAM template section for a Lambda authorizer:

Resources:
  MyApi:
    Type: AWS::Serverless::HttpApi
    Properties:
      StageName: Prod
    Auth:
      DefaultAuthorizer: MyLambdaRequestAuthorizer
      Authorizers:
        MyLambdaRequestAuthorizer:
          FunctionArn: !GetAtt MyAuthFunction.Arn
          FunctionInvokeRole: !GetAtt MyAuthFunctionRole.Arn
          Identity:
            Headers:
              Authorization
            AuthorizerPayloadFormatVersion: 2.0
            EnableSimpleResponses: true

MyFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: ./src
    Handler: index.handler
IAM permission example

You can control access to your APIs by defining IAM permissions within your AWS SAM template. To do this, you use the `ApiAuth (p. 40)` data type.

The following is an example AWS SAM template section for IAM permissions:

```
Resources:
  MyApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      Auth:
        DefaultAuthorizer: AWS_IAM

  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: .
      Handler: index.handler
      Runtime: nodejs12.x
    Events:
      GetRoot:
        Type: Api
        Properties:
          RestApiId: !Ref MyApi
          Path: /
          Method: get
```

For more information about IAM permissions, see Control access to an API with IAM permissions in the `API Gateway Developer Guide`.

Amazon Cognito user pool example

You can control access to your APIs by defining Amazon Cognito user pools within your AWS SAM template. To do this, you use the `ApiAuth (p. 40)` data type.

The following is an example AWS SAM template section for a user pool:

```
Resources:
  MyApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
```
Cors: "*"
Auth:
  DefaultAuthorizer: MyCognitoAuthorizer
Authorizers:
  MyCognitoAuthorizer:
    UserPoolArn: !GetAtt MyCognitoUserPool.Arn

MyFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: ./src
    Handler: lambda.handler
    Runtime: nodejs12.x
  Events:
    Root:
      Type: Api
      Properties:
        RestApiId: !Ref MyApi
        Path: /
        Method: GET

MyCognitoUserPool:
  Type: AWS::Cognito::UserPool
  Properties:
    UserPoolName: !Ref CognitoUserPoolName
    Policies:
      PasswordPolicy:
        MinimumLength: 8
    UsernameAttributes:
      - email
    Schema:
      - AttributeDataType: String
        Name: email
        Required: false

MyCognitoUserPoolClient:
  Type: AWS::Cognito::UserPoolClient
  Properties:
    UserPoolId: !Ref MyCognitoUserPool
    ClientName: !Ref CognitoUserPoolClientName
    GenerateSecret: false

For more information about Amazon Cognito user pools, see Control access to a REST API using Amazon Cognito user pools as authorizer in the API Gateway Developer Guide.

API key example

You can control access to your APIs by requiring API keys within your AWS SAM template. To do this, you use the ApiAuth (p. 40) data type.

The following is an example AWS SAM template section for API keys:

Resources:
MyApi:
  Type: AWS::Serverless::Api
  Properties:
    StageName: Prod
  Auth:
    ApiKeyRequired: true # sets for all methods

MyFunction:
  Type: AWS::Serverless::Function
  Properties:
CodeUri: 
Handler: index.handler
Runtime: nodejs12.x
Events:
  ApiKey:
    Type: Api
    Properties:
      RestApiId: !Ref MyApi
      Path: /
      Method: get
      Auth:
        ApiKeyRequired: true

For more information about API keys, see Creating and using usage plans with API keys in the API Gateway Developer Guide.

Resource policy example

You can control access to your APIs by attaching a resource policy within your AWS SAM template. To do this, you use the ApiAuth (p. 40) data type.

The following is an example AWS SAM template section for resource policies:

Resources:
  ExplicitApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      EndpointConfiguration: PRIVATE
      Auth:
        ResourcePolicy:
          CustomStatements: {
            Effect: 'Allow',
            Action: 'execute-api:Invoke',
            Resource: ['execute-api:/*/*/*'],
            Principal: '*'
          }
  MinimalFunction:
    Type: 'AWS::Serverless::Function'
    Properties:
      CodeUri: s3://sam-demo-bucket/hello.zip
      Handler: hello.handler
      Runtime: python2.7
      Events:
        AddItem:
          Type: Api
          Properties:
            RestApiId: !Ref ExplicitApi
            Path: /add
            Method: post

For more information about resource policies, see Controlling access to an API with API Gateway resource policies in the API Gateway Developer Guide.

OAuth 2.0/JWT authorizer example

You can control access to your APIs using JWTs as part of OpenID Connect (OIDC) and OAuth 2.0 frameworks. To do this, you use the HttpApiAuth (p. 135) data type.

The following is an example AWS SAM template section for an OAuth 2.0/JWT authorizer:
Customized response example

You can customize some API Gateway error responses by defining response headers within your AWS SAM template. To do this, you use the Gateway Response Object data type.

The following is an example AWS SAM template section for API Gateway responses:

```json
Resources:
  MyApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      GatewayResponses:
        DEFAULT_4xx:
          ResponseParameters:
            Headers:
              Access-Control-Expose-Headers: "WWW-Authenticate"
              Access-Control-Allow-Origin: "*"
  GetFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.get
      Runtime: nodejs12.x
      InlineCode: module.exports = async () => throw new Error('Check out the response headers!')
      Events:
```

For more information about OAuth 2.0/JWT authorizers, see Controlling access to HTTP APIs with JWT authorizers in the API Gateway Developer Guide.
Orchestrating applications

GetResource:
  Type: Api
  Properties:
    Path: /error
    Method: get
    RestApiId: !Ref MyApi

For more information about API Gateway responses, see Gateway responses in API Gateway in the API Gateway Developer Guide.

Orchestrating AWS resources with AWS Step Functions

You can use AWS Step Functions to orchestrate AWS Lambda functions and other AWS resources to form complex and robust workflows.

Note
To manage AWS SAM templates that contain Step Functions state machines, you must use version 0.52.0 or later of the AWS SAM CLI. To check which version you have, execute the command `sam --version`.

Step Functions is based on the concepts of tasks and state machines. You define state machines using the JSON-based Amazon States Language. The Step Functions console displays a graphical view of your state machine's structure so you can visually check your state machine's logic and monitor executions.

With Step Functions support in AWS Serverless Application Model (AWS SAM), you can do the following:

- Define state machines, either directly within an AWS SAM template or in a separate file
- Create state machine execution roles through AWS SAM policy templates, inline policies, or managed policies
- Trigger state machine executions with API Gateway or Amazon EventBridge events, on a schedule within an AWS SAM template, or by calling APIs directly
- Use available AWS SAM Policy Templates for common Step Functions development patterns.

Example

The following example snippet from a AWS SAM template file defines a Step Functions state machine in a definition file. Note that the `my_state_machine.asl.json` file must be written in Amazon States Language.

```json
AWSTemplateFormatVersion: "2010-09-09"
Transform: AWS::Serverless-2016-10-31
Description: Sample SAM template with Step Functions State Machine

Resources:
  MyStateMachine:
    Type: AWS::Serverless::StateMachine
    Properties:
      DefinitionUri: statemachine/my_state_machine.asl.json
...
```

To download a sample AWS SAM application that includes a Step Functions state machine, see Create a Step Functions State Machine Using AWS SAM in the AWS Step Functions Developer Guide.
More information

To learn more about Step Functions and using it with AWS SAM, see the following:

- How AWS Step Functions works
- AWS Step Functions and AWS Serverless Application Model
- Tutorial: Create a Step Functions State Machine Using AWS SAM
- AWS SAM Specification: AWS::Serverless::StateMachine (p. 154)

Configuring code signing for AWS SAM applications

You can use AWS SAM to enable code signing with your serverless applications to help ensure that only trusted code is deployed. For more information about the code signing feature, see Configuring code signing for Lambda functions in the AWS Lambda Developer Guide.

Before you can configure code signing for your serverless application, you must create a signing profile using AWS Signer. You use this signing profile for the following tasks:

1. Creating a code signing configuration – Declare an AWS::Lambda::CodeSigningConfig resource to specify the signing profiles of trusted publishers and to set the policy action for validation checks. You can declare this object in the same AWS SAM template as your serverless function, in a different AWS SAM template, or in an AWS CloudFormation template. You then enable code signing for a serverless function by specify the CodeSigningConfigArn property the function with the Amazon Resource Name (ARN) of an AWS::Lambda::CodeSigningConfig resource.

2. Signing your code – Use the `sam package` or `sam deploy` command with the `--signing-profiles` option.

When you deploy a serverless application, Lambda performs validation checks on all functions that you've enabled code signing for. Lambda also performs validation checks on any layers that those functions depend on. For more information about Lambda's validation checks, see Signature validation in the AWS Lambda Developer Guide.

Example

Creating a signing profile

To create a signing profile, run the following command:

```bash
aws signer put-signing-profile --platform-id "AWSLambda-SHA384-ECDSA" --profile-name MySigningProfile
```

If the previous command is successful, you see the signing profile's ARN returned. For example:

```json
{
  "arn": "arn:aws:signer:us-east-1:111122223333:/signing-profiles/MySigningProfile",
  "profileVersion": "SAMPLEEverx",
  "profileVersionArn": "arn:aws:signer:us-east-1:111122223333:/signing-profiles/MySigningProfile/SAMPLEEverx"
}
```
Providing signing profiles with `sam deploy --guided`

When you run the `sam deploy --guided` command with a serverless application that's configured with code signing, AWS SAM prompts you to provide the signing profile to use for code signing. For more
Providing signing profiles with `sam deploy --guided`

Information about `sam deploy --guided` prompts, see `sam deploy (p. 241)` in the AWS SAM CLI command reference.
Building serverless applications

Building your serverless application involves taking your AWS SAM template file, application code, and any applicable language-specific files and dependencies, and placing all build artifacts in the proper format and location for subsequent steps in your workflow.

For example, you might want to locally test your application, or you might want to deploy your application using the AWS SAM CLI. Both of these activities use the build artifacts of your application as inputs.

This section shows you how to use the `sam build (p. 237)` command to build serverless applications using AWS SAM. You have the option to build all functions and layers of your application, or individual components of your application, like a specific function or layer.

Topics
- Building applications (p. 199)
- Building layers (p. 202)
- Building custom runtimes (p. 204)

Building applications

To build your serverless application, use the `sam build (p. 237)` command. This command also gathers the build artifacts of your application's dependencies and places them in the proper format and location for next steps, such as locally testing, packaging, and deploying.

You specify your application's dependencies in a manifest file, such as `requirements.txt` (Python) or `package.json` (Node.js), or by using the `Layers` property of a function resource. The `Layers` property contains a list of AWS Lambda layer resources that the Lambda function depends on.

The format of your application's build artifacts depends on each function's `PackageType` property. The options for this property are:

- **Zip** – A .zip file archive, which contains your application code and its dependencies. If you package your code as a .zip file archive, you must specify a Lambda runtime for your function.
- **Image** – A container image, which includes the base operating system, runtime, and extensions, in addition to your application code and its dependencies.

For more information about Lambda package types, see Lambda deployment packages in the AWS Lambda Developer Guide.

Building a .zip file archive

To build your serverless application as a .zip file archive, declare `PackageType: Zip` for your serverless function.

If your Lambda function depends on packages that have natively compiled programs, use the `--use-container` flag. This flag locally compiles your functions in a Docker container that behaves like a Lambda environment, so they're in the right format when you deploy them to the AWS Cloud.

For an example of building a .zip file archive application, see the Examples section later in this topic.
Building a container image

To build your serverless application as a container image, declare PackageType: Image for your serverless function. You must also declare the Metadata resource attribute with the following entries:

**Dockerfile**

- The name of the Dockerfile associated with the Lambda function.

**DockerContext**

- The location of the Dockerfile.

**DockerTag**

- (Optional) A tag to apply to the built image.

**DockerBuildArgs**

- Build arguments for the build.

The following is an example Metadata resource attribute section:

```
Metadata:
  Dockerfile: Dockerfile
  DockerContext: ./hello_world
  DockerTag: v1
```

To download a sample application that's configured with the Image package type, see Step 1: Download a sample AWS SAM application (p. 16) in Tutorial: Deploying a Hello World application. At the prompt asking which package type you want to install, choose Image.

Container environment variable file

To provide a JSON file that contains environment variables for the build container, use the `--container-env-var-file` argument with the `sam build` command. You can provide a single environment variable that applies to all serverless resources, or different environment variables for each resource.

**Format**

The format for passing environment variables to a build container depends on how many environment variables you provide for your resources.

To provide a single environment variable for all resources, specify a `Parameters` object like the following:

```
{
  "Parameters": {
    "GITHUB_TOKEN": "TOKEN_GLOBAL"
  }
}
```

To provide different environment variables for each resource, specify objects for each resource like the following:

```
{
}
```
Save your environment variables as a file, for example, named `env.json`. The following command uses this file to pass your environment variables to the build container:

```
sam build --use-container --container-env-var-file env.json
```

## Precedence

- The environment variables that you provide for specific resources take precedence over the single environment variable for all resources.
- Environment variables that you provide on the command line take precedence over environment variables in a file.

## Examples

### Example 1: .zip file archive

The following `sam build` commands build a .zip file archive:

```
# Build all functions and layers, and their dependencies
sam build

# Run the build process inside a Docker container that functions like a Lambda environment
sam build --use-container

# Build and run your functions locally
sam build && sam local invoke

# For more options
sam build --help
```

### Example 2: Container image

The following AWS SAM template builds as a container image:

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      PackageType: Image
      ImageConfig:
        Command: ["app.lambda_handler"]
    Metadata:
      Dockerfile: Dockerfile
      DockerContext: ./hello_world
      DockerTag: v1
```

The following is an example Dockerfile:
Building layers

You can use AWS SAM to build custom layers. For information about layers, see AWS Lambda layers in the AWS Lambda Developer Guide.

To build a custom layer, declare it in your AWS Serverless Application Model (AWS SAM) template file and include a Metadata resource attribute section with a BuildMethod entry. Valid values for BuildMethod are identifiers for an AWS Lambda runtime, or makefile.

If you specify makefile, provide the custom makefile, where you declare a build target of the form build-layer-logical-id that contains the build commands for your layer. Your makefile is responsible for compiling the layer if necessary, and copying the build artifacts into the proper location required for subsequent steps in your workflow. The location of the makefile is specified by the ContentUri property of the layer resource, and must be named Makefile.

Note
When you create a custom layer, AWS Lambda depends on environment variables to find your layer code. Lambda runtimes include paths in the /opt directory where your layer code is copied into. Your project's build artifact folder structure must match the runtime's expected folder structure so your custom layer code can be found.

For example, for Python you can place your code in the python/ subdirectory. For NodeJS, you can place your code in the nodejs/node_modules/ subdirectory.

For more information, see Including library dependencies in a layer in the AWS Lambda Developer Guide.

The following is an example Metadata resource attribute section.

```
Metadata:
  BuildMethod: python3.6
```

Note
If you don't include the Metadata resource attribute section, AWS SAM doesn't build the layer. Instead, it copies the build artifacts from the location specified in the CodeUri property of the layer resource. For more information, see the ContentUri (p. 148) property of the AWS::Serverless::LayerVersion resource type.

When you include the Metadata resource attribute section, you can use the `sam build` command to build the layer, both as an independent object, or as a dependency of an AWS Lambda function.

- **As an independent object.** You might want to build just the layer object, for example when you're locally testing a code change to the layer and don't need to build your entire application. To build the layer independently, specify the layer resource with the `sam build layer-logical-id` command.

- **As a dependency of a Lambda function.** When you include a layer's logical ID in the Layers property of a Lambda function in the same AWS SAM template file, the layer is a dependency of that Lambda function. When that layer also includes a Metadata resource attribute section with a BuildMethod
entry, you build the layer either by building the entire application with the `sam build` command or by specifying the function resource with the `sam build function-logical-id` command.

**Examples**

**Template example 1: Build a layer against the Python 3.6 runtime environment**

The following example AWS SAM template builds a layer against the Python 3.6 runtime environment.

Resources:

<table>
<thead>
<tr>
<th>MyLayer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type: AWS::Serverless::LayerVersion</td>
</tr>
<tr>
<td>Properties:</td>
</tr>
<tr>
<td>ContentUri: my_layer</td>
</tr>
<tr>
<td>CompatibleRuntimes:</td>
</tr>
<tr>
<td>- python3.6</td>
</tr>
<tr>
<td>Metadata:</td>
</tr>
<tr>
<td>BuildMethod: python3.6     # Required to have AWS SAM build this layer</td>
</tr>
</tbody>
</table>

**Template example 2: Build a layer using a custom makefile**

The following example AWS SAM template uses a custom makefile to build the layer.

Resources:

<table>
<thead>
<tr>
<th>MyLayer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type: AWS::Serverless::LayerVersion</td>
</tr>
<tr>
<td>Properties:</td>
</tr>
<tr>
<td>ContentUri: my_layer</td>
</tr>
<tr>
<td>CompatibleRuntimes:</td>
</tr>
<tr>
<td>- python3.8</td>
</tr>
<tr>
<td>Metadata:</td>
</tr>
<tr>
<td>BuildMethod: makefile</td>
</tr>
</tbody>
</table>

The following makefile contains the build target and commands that will be executed. Note that the `ContentUri` property is set to `my_layer`, so the makefile must be located in the root of the `my_layer` subdirectory, and the filename must be `Makefile`. Note also that the build artifacts are copied into the `python/` subdirectory so that AWS Lambda will be able to find the layer code.

```
build-MyLayer:
  mkdir -p "$(ARTIFACTS_DIR)/python"
  cp *.py "$(ARTIFACTS_DIR)/python"
  python -m pip install -r requirements.txt -t "$(ARTIFACTS_DIR)/python"
```

**Example sam build commands**

The following `sam build` commands build layers that include the `Metadata` resource attribute sections.

```
# Build the 'layer-logical-id' resource independently
sam build layer-logical-id

# Build the 'function-logical-id' resource and layers that this function depends on
sam build function-logical-id

# Build the entire application, including the layers that any function depends on
```
Building custom runtimes

You can use the `sam build (p. 237)` command to build custom runtimes required for your Lambda function. You declare your Lambda function to use a custom runtime by specifying `Runtime: provided` for the function.

To build a custom runtime, declare the `Metadata` resource attribute with a `BuildMethod: makefile` entry. You provide a custom makefile, where you declare a build target of the form `build-function-logical-id` that contains the build commands for your runtime. Your makefile is responsible for compiling the custom runtime if necessary, and copying the build artifacts into the proper location required for subsequent steps in your workflow. The location of the makefile is specified by the `CodeUri` property of the function resource, and must be named `Makefile`.

Examples

Example 1: Custom runtime for a function written in Rust

The following AWS SAM template declares a function that uses a custom runtime for a Lambda function written in Rust, and instructs `sam build` to execute the commands for the `build-HelloRustFunction` build target.

```
Resources:
HelloRustFunction:
  Type: AWS::Serverless::Function
Properties:
  FunctionName: HelloRust
  Handler: bootstrap.is.real.handler
  Runtime: provided
  MemorySize: 512
  CodeUri: .
  Metadata:
    BuildMethod: makefile

The following makefile contains the build target and commands that will be executed. Note that the `CodeUri` property is set to `.`, so the makefile must be located in the project root directory (that is, the same directory as the application's AWS SAM template file). The filename must be `Makefile`.

```
build-HelloRustFunction:
  cargo build --release --target x86_64-unknown-linux-musl
  cp ./target/x86_64-unknown-linux-musl/release/bootstrap $(ARTIFACTS_DIR)
```

For more information about setting up your development environment in order to execute the `cargo build` command in the previous makefile, see the Rust Runtime for AWS Lambda blog post.

Example 2: Makefile builder for Python3.7 (alternative to using the bundled builder)

You might want to use a library or module that is not included in a bundled builder. This example shows a AWS SAM template for a Python3.7 runtime with a makefile builder.

```
Resources:
```
HelloWorldFunction:
   Type: AWS::Serverless::Function
   Properties:
      CodeUri: hello_world/
      Handler: app.lambda_handler
      Runtime: python3.7
   Metadata:
      BuildMethod: makefile

The following makefile contains the build target and commands that will be executed. Note that the CodeUri property is set to hello_world, so the makefile must be located in the root of the hello_world subdirectory, and the filename must be Makefile.

```
build-HelloWorldFunction:
   cp *.py $(ARTIFACTS_DIR)
   cp requirements.txt $(ARTIFACTS_DIR)
   python -m pip install -r requirements.txt -t $(ARTIFACTS_DIR)
   rm -rf $(ARTIFACTS_DIR)/bin
```
Testing and debugging serverless applications

With the AWS SAM command line interface (CLI), you can locally test and "step-through" debug your serverless applications before uploading your application to the AWS Cloud. You can verify whether your application is behaving as expected, debug what's wrong, and fix any issues, before going through the steps of packaging and deploying your application.

When you locally invoke a Lambda function in debug mode within the AWS SAM CLI, you can then attach a debugger to it. With the debugger, you can step through your code line by line, see the values of various variables, and fix issues the same way you would for any other application.

**Note**
If your application includes one or more layers, when you locally run and debug your application the layers package is downloaded and cached on your local host. For more information, see How layers are cached locally (p. 183).

**Topics**
- Invoking functions locally (p. 206)
- Running API Gateway locally (p. 208)
- Integrating with automated tests (p. 211)
- Generating sample event payloads (p. 212)
- Step-through debugging Lambda functions locally (p. 212)
- Passing additional runtime debug arguments (p. 218)

Invoking functions locally

You can invoke your function locally by using the `sam local invoke (p. 248)` command and providing its function logical ID and an event file. Alternatively, `sam local invoke` also accepts stdin as an event.

**Note**
The `sam local invoke` command described in this section corresponds to the AWS CLI command `aws lambda invoke`. You can use either version of this command to invoke a Lambda function that you've uploaded to the AWS Cloud.

You must execute `sam local invoke` in the project directory containing the function you want to invoke.

**Examples:**

```bash
# Invoking function with event file
$ sam local invoke "Ratings" -e event.json

# Invoking function with event via stdin
$ echo '{"message": "Hey, are you there?" }' | sam local invoke --event - "Ratings"

# For more options
$ sam local invoke --help
```
Environment variable file

You can use the `--env-vars` argument with the `invoke` or `start-api` commands. You do this to provide a JSON file that contains values to override the environment variables that are already defined in your function template. You can structure the file as follows:
Alternatively, your environment file can contain a single `Parameters` entry with the environment variables for all functions. Note that you can't mix this format with the example above.

```json

{  
  "Parameters": {  
    "TABLE_NAME": "localtable",
    "BUCKET_NAME": "testBucket",
    "STAGE": "dev"
  }
}

```

Save your environment variables in a file named `env.json`. The following command uses this file to override the included environment variables:

```
sam local invoke --env-vars env.json
```

**Layers**

If your application includes layers, see [Working with layers (p. 182)](#) for more information about how to debug layers issues on your local host.

**Running API Gateway locally**

Use the `sam local start-api (p. 250)` command to start a local instance of API Gateway that you will use to test HTTP request/response functionality. This functionality features hot reloading to enable you to quickly develop and iterate over your functions.

**Note**

"Hot reloading" is when only the files that changed are refreshed without losing the state of the application. In contrast, "live reloading" is when the entire application is refreshed, such that the state of the application is lost.

You must execute `sam local start-api` in the project directory containing the function you want to invoke.

By default, AWS SAM uses Lambda proxy integrations, and supports both `HttpApi` and `Api` resource types. For more information about proxy integrations for `HttpApi` resource types, see [Working with Lambda proxy integrations for HTTP APIs](#). For more information about proxy integrations with `Api` resource types, see [Understand API Gateway Lambda proxy integration](#).

**Example:**

```
sam local start-api
```

AWS SAM automatically finds any functions within your AWS SAM template that have `HttpApi` or `Api` event sources defined. Then, it mounts them at the defined HTTP paths.
This animation shows running API Gateway locally using Microsoft Visual Studio Code:

In the following API example, the Ratings function mounts `ratings.py:handler()` at `/ratings` for GET requests:

```
Ratings:
  Type: AWS::Serverless::Function
  Properties:
```
Handler: ratings.handler
Runtime: python3.6
Events:
  Api:
    Type: Api
    Properties:
      Path: /ratings
      Method: get

Here is an example Api response:

```javascript
// Example of a Proxy Integration response
exports.handler = (event, context, callback) => {
  callback(null, {
    statusCode: 200,
    headers: { "x-custom-header" : "my custom header value" },
    body: "hello world"
  });
}
```

Environment Variable File

You can use the `--env-vars` argument with the `invoke` or `start-api` commands to provide a JSON file that contains values to override the environment variables already defined in your function template. You can structure the file as follows:

```json
{
  "MyFunction1": {
    "TABLE_NAME": "localtable",
    "BUCKET_NAME": "testBucket"
  },
  "MyFunction2": {
    "TABLE_NAME": "localtable",
    "STAGE": "dev"
  }
}
```

Alternatively, your environment file can contain a single `Parameters` entry with the environment variables for all functions. Note that you can't mix this format with the example above.

```json
{
  "Parameters": {
    "TABLE_NAME": "localtable",
    "BUCKET_NAME": "testBucket",
    "STAGE": "dev"
  }
}
```

Save your environment variables in a file named `env.json`. The following command uses this file to override the included environment variables:

```
sam local start-api --env-vars env.json
```

Layers

If your application includes layers, see [Working with layers (p. 182)](#) for more information about how to debug layers issues on your local host.
Integrating with automated tests

You can use the `sam local invoke` command to manually test your code by running Lambda functions locally. With the AWS SAM CLI, you can easily author automated integration tests by first running tests against local Lambda functions before deploying to the AWS Cloud.

The `sam local start-lambda` command starts a local endpoint that emulates the AWS Lambda invoke endpoint. You can invoke it from your automated tests. Because this endpoint emulates the AWS Lambda invoke endpoint, you can write tests once, and then run them (without any modifications) against the local Lambda function, or against a deployed Lambda function. You can also run the same tests against a deployed AWS SAM stack in your CI/CD pipeline.

This is how the process works:

1. **Start the local Lambda endpoint.**

   Start the local Lambda endpoint by running the following command in the directory that contains your AWS SAM template:

   ```bash
   sam local start-lambda
   ```

   This command starts a local endpoint at `http://127.0.0.1:3001` that emulates AWS Lambda. You can run your automated tests against this local Lambda endpoint. When you invoke this endpoint using the AWS CLI or SDK, it locally executes the Lambda function that's specified in the request, and returns a response.

2. **Run an integration test against the local Lambda endpoint.**

   In your integration test, you can use the AWS SDK to invoke your Lambda function with test data, wait for response, and verify that the response is what you expect. To run the integration test locally, you should configure the AWS SDK to send a Lambda Invoke API call to invoke the local Lambda endpoint that you started in previous step.

   The following is a Python example (the AWS SDKs for other languages have similar configurations):

   ```python
   import boto3
   import botocore

   # Set "running_locally" flag if you are running the integration test locally
   running_locally = True

   if running_locally:
       # Create Lambda SDK client to connect to appropriate Lambda endpoint
       lambda_client = boto3.client('lambda',
                                    region_name="us-west-2",
                                    endpoint_url="http://127.0.0.1:3001",
                                    use_ssl=False,
                                    verify=False,
                                    config=botocore.client.Config(  
                                        signature_version=botocore.UNSIGNED,
                                        read_timeout=1,
                                        retries={'max_attempts': 0},
                                    )
    
    else:
        lambda_client = boto3.client('lambda')

   # Invoke your Lambda function as you normally usually do. The function will run
   ```
Generating sample event payloads

To make local development and testing of Lambda functions easier, you can generate and customize event payloads for a number of AWS services like API Gateway, AWS CloudFormation, Amazon S3, and so on.

For the full list of services that you can generate sample event payloads for, use this command:

```
$ sam local generate-event --help
```

For the list of options you can use for a particular service, use this command:

```
$ sam local generate-event [SERVICE] --help
```

Examples:

```
# Generates the event from S3 when a new object is created
$ sam local generate-event s3 put

# Generates the event from S3 when an object is deleted
$ sam local generate-event s3 delete
```

Step-through debugging Lambda functions locally

You can use AWS SAM with a number of AWS toolkits to test and debug your serverless applications locally.

For example, you can perform step-through debugging of your Lambda functions. Step-through debugging makes it easier to understand what the code is doing. It tightens the feedback loop by making it possible for you to find and troubleshoot issues that you might run into in the cloud.

Using AWS Toolkits

AWS toolkits are plugins that provide you with the ability to perform many common debugging tasks, like setting breakpoints, executing code line by line, and inspecting the values of variables. Toolkits make it easier for you to develop, debug, and deploy serverless applications that are built using AWS. They provide an experience for building, testing, debugging, deploying, and invoking Lambda functions that's integrated into the integrated development environment (IDE).

For more information about AWS toolkits that you can use with AWS SAM, see the following:
Running AWS SAM locally

The commands `sam local invoke` and `sam local start-api` both support local step-through debugging of your Lambda functions. To run AWS SAM locally with step-through debugging support enabled, specify `--debug-port` or `-d` on the command line. For example:

```
# Invoke a function locally in debug mode on port 5858
sam local invoke -d 5858 <function logical id>

# Start local API Gateway in debug mode on port 5858
sam local start-api -d 5858
```

**Note**

If you're using `sam local start-api`, the local API Gateway instance exposes all of your Lambda functions. However, because you can specify a single debug port, you can only debug one function at a time. You need to call your API before the AWS SAM CLI binds to the port, which allows the debugger to connect.

**Topics**

The following topics provide examples of how to set up your environment to test and debug your serverless applications locally.

- Step-through debugging Node.js functions locally (p. 213)
- Step-through debugging Python functions locally (p. 215)
- Step-through debugging Golang functions locally (p. 217)

**Step-through debugging Node.js functions locally**

The following is an example that shows how to debug a Node.js function with Microsoft Visual Studio Code:
To set up Microsoft Visual Studio Code for step-through debugging Node.js functions with the AWS SAM CLI, use the following launch configuration. Before you do this, set the directory where the template.yaml file is located as the workspace root in Microsoft Visual Studio Code:

```json
{
  "version": "0.2.0",
  "configurations": [
```
```json
{
    "name": "Attach to SAM CLI",
    "type": "node",
    "request": "attach",
    "address": "localhost",
    "port": 5858,
    // From the sam init example, it would be "${workspaceRoot}/hello-world"
    "localRoot": "${workspaceRoot}/(directory of node app)",
    "remoteRoot": "/var/task",
    "protocol": "inspector",
    "stopOnEntry": false
}
}
```

**Note**
The `localRoot` is set based on what the CodeUri points at in the `template.yaml` file. If there are nested directories within the CodeUri, that needs to be reflected in the `localRoot`.

**Note**
Node.js versions earlier than 7 (for example, Node.js 4.3 and Node.js 6.10) use the `legacy` protocol, while Node.js versions including and later than 7 (for example, Node.js 8.10) use the `inspector` protocol. Be sure to specify the corresponding protocol in the `protocol` entry of your launch configuration. This was tested with Microsoft Visual Studio Code versions 1.26, 1.27, and 1.28 for the legacy and inspector protocols.

## Step-through debugging Python functions locally

Python step-through debugging requires you to enable remote debugging in your Lambda function code. This is a two-step process:

1. Install the ptvsd library and enable it within your code.
2. Configure your IDE to connect to the debugger that you configured for your function.

Because this might be your first time using the AWS SAM CLI, start with a boilerplate Python application, and install both the application’s dependencies and ptvsd:

```bash
sam init --runtime python3.6 --name python-debugging
cd python-debugging/
# Install dependencies of our boilerplate app
pip install -r hello_world/requirements.txt -t hello_world/build/
# Install ptvsd library for step through debugging
pip install ptvsd -t hello_world/build/
cp hello_world/app.py hello_world/build/
```

### Ptvsd configuration

Next, you need to enable ptvsd within your code. To do this, open `hello_world/build/app.py`, and add the following ptvsd specifics:

```python
import ptvsd

# Enable ptvsd on 0.0.0.0 address and on port 5890 that we'll connect later with our IDE
ptvsd.enable_attach(address=('0.0.0.0', 5890), redirect_output=True)
ptvsd.wait_for_attach()
```
Use 0.0.0.0 instead of localhost for listening across all network interfaces. 5890 is the debugging port that you want to use.

**Microsoft Visual Studio Code**

Now that you have the dependencies and ptvsd enabled within your code, you can configure Microsoft Visual Studio Code debugging. Assuming that you’re still in the application folder and have the code command in your path, open Microsoft Visual Studio Code by using this command:

```
code .
```

**Note**

If you don’t have code in your path, open a new instance of Microsoft Visual Studio Code from the python-debugging/ folder that you created earlier.

To set up Microsoft Visual Studio Code for debugging with the AWS SAM CLI, use the following launch configuration:

```json
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "SAM CLI Python Hello World",
      "type": "python",
      "request": "attach",
      "port": 5890,
      "host": "localhost",
      "pathMappings": [
        {
          "localRoot": "${workspaceFolder}/hello_world/build",
          "remoteRoot": "/var/task"
        }
      ]
    }
  ]
}
```

For Microsoft Visual Studio Code, the property localRoot under the pathMappings key is important. There are two reasons that help explain this setup:

- **localRoot**: This path is to be mounted in the Docker container, and needs to have both the application and dependencies at the root level.
- **workspaceFolder**: This path is the absolute path where the Microsoft Visual Studio Code instance was opened.

If you opened Microsoft Visual Studio Code in a different location other than python-debugging/, you need to replace it with the absolute path where python-debugging/ is located.

After the Microsoft Visual Studio Code debugger configuration is complete, make sure to add a breakpoint anywhere you want to in hello_world/build/app.py, and then proceed as follows:

1. Run the AWS SAM CLI to invoke your function.
2. Send a request to the URL to invoke the function and initialize ptvsd code execution.

# Remember to hit the URL before starting the debugger in Microsoft Visual Studio Code
Step-through debugging Golang functions locally

Golang function step-through debugging is slightly different when compared to Node.js, Java, and Python. We require Delve as the debugger, and wrap your function with it at runtime. The debugger is run in headless mode, listening on the debug port.

When you're debugging, you must compile your function in debug mode:

```
GOARCH=amd64 GOOS=linux go build -gcflags=''-N -l'' -o <output path> <path to code directory>
```

**Delve debugger**

You must compile Delve to run in the container and provide its local path with the `--debugger-path` argument.

Build Delve locally as follows:

```
GOARCH=amd64 GOOS=linux go build -o <delve folder path>/dlv github.com/go-delve/delve/cmd/dlv
```

**Delve debugger path**

The output path needs to end in `/dlv`. The Docker container expects the dlv binary file to be in the `<delve folder path>`. If it's not, a mounting issue occurs.

*Note*

The `--debugger-path` is the path to the directory that contains the dlv binary file that's compiled from the previous code.

**Example**

Invoke AWS SAM similar to the following:

```
sam local start-api -d 5986 --debugger-path <delve folder path>
```

**Delve debugger API version**

To run the Delve debugger with an API version of your choice, specify the desired API version using an additional debug argument (p. 218) `--delveAPI`.

*Note*

For IDEs such as GoLand, Microsoft Visual Studio Code, etc., it is important to run Delve in API version 2 mode.

**Example**

Invoke AWS SAM with the Delve debugger in API version 2 mode:

```
sam local start-api -d 5986 --debugger-path <delve folder path> --debug-args "--delveAPI=2"
```
Example

The following is an example launch configuration for Microsoft Visual Studio Code to attach to a debug session.

```json
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "Connect to Lambda container",
      "type": "go",
      "request": "launch",
      "mode": "remote",
      "remotePath": "",
      "port": <debug port>,
      "host": "127.0.0.1",
      "program": "${workspaceRoot}",
      "env": {},
      "args": [],
    }
  ]
}
```

**Passing additional runtime debug arguments**

To pass additional runtime arguments when you're debugging your function, use the environment variable `DEBUGGER_ARGS`. This passes a string of arguments directly into the run command that the AWS SAM CLI uses to start your function.

For example, if you want to load a debugger like iKPdb at the runtime of your Python function, you could pass the following as `DEBUGGER_ARGS`: `-m ikpdb --ikpdb-port=5858 --ikpdb-working-directory=/var/task/ --ikpdb-client-working-directory=/myApp --ikpdb-address=0.0.0.0`. This would load iKPdb at runtime with the other arguments you've specified.

In this case, your full AWS SAM CLI command would be:

```
DEBUGGER_ARGS="-m ikpdb --ikpdb-port=5858 --ikpdb-working-directory=/var/task/ --ikpdb-client-working-directory=/myApp --ikpdb-address=0.0.0.0" echo {} | sam local invoke -d 5858 myFunction
```

You can pass debugger arguments to the functions of all runtimes.
Deploying serverless applications

AWS SAM uses AWS CloudFormation as the underlying deployment mechanism. For more information, see What is AWS CloudFormation? in the AWS CloudFormation User Guide.

You can deploy your application using AWS SAM command line interface (CLI) commands. To automate your deployments, you can also use other AWS services that integrate with AWS SAM.

The standard inputs to deploying serverless applications are the build artifacts created using the `sam build` (p. 237) command. For more information about `sam build`, see Building serverless applications (p. 199).

Deploying using the AWS SAM CLI

After you develop and test your serverless application locally, you can deploy your application using the `sam deploy` command.

If you want AWS SAM to guide you through the deployment with prompts, specify the `--guided` flag. When you specify this flag, the `sam deploy` command zips your application artifacts, uploads them to either Amazon S3 (for .zip file archives) or Amazon ECR (for container images), and then deploys your application to the AWS Cloud.

Example:

```
# Deploy an application using prompts:
sam deploy --guided
```

Publishing serverless applications

The AWS Serverless Application Repository is a service that hosts serverless applications that are built using AWS SAM. If you want to share serverless applications with others, you can publish them in the AWS Serverless Application Repository. You can also search, browse, and deploy serverless applications that others have published. For more information, see What Is the AWS Serverless Application Repository? in the AWS Serverless Application Repository Developer Guide.

Automating deployments

To automate the deployment process of your serverless application, you can use AWS SAM with a number of other AWS services.

- **CodeBuild**: Use AWS CodeBuild to build, locally test, and package your serverless application. For more information, see What is AWS CodeBuild? in the AWS CodeBuild User Guide.
- **CodeDeploy**: Use AWS CodeDeploy to gradually deploy updates to your serverless applications. For more information, see Deploying serverless applications gradually (p. 220).
- **CodePipeline**: Use AWS CodePipeline to model, visualize, and automate the steps that are required to release your serverless application. For more information, see What is AWS CodePipeline? in the AWS CodePipeline User Guide.
Troubleshooting

AWS SAM CLI error: "Security Constraints Not Satisfied"

When executing `sam deploy --guided`, you are prompted with the question: **HelloWorldFunction may not have authorization defined, Is this okay? [y/N]**. If you respond to this prompt with "N" (the default response), you see the following error:

```
Error: Security Constraints Not Satisfied
```

The prompt is informing you that the application you’re about to deploy may have an API Gateway API configured without authorization. By responding "N" to this prompt (the default), you are saying that this is not OK.

To fix this, you have the following options:

- Configure your application with authorization. For information about configuring authorization, see [Controlling access to API Gateway APIs](p. 186).
- Respond to this question with "Y" to indicate that you are OK with deploying an application that has an API Gateway API configured without authorization.

Deploying serverless applications gradually

If you use AWS SAM to create your serverless application, it comes built-in with CodeDeploy to provide gradual Lambda deployments. With just a few lines of configuration, AWS SAM does the following for you:

- Deploys new versions of your Lambda function, and automatically creates aliases that point to the new version.
- Gradually shifts customer traffic to the new version until you’re satisfied that it’s working as expected, or you roll back the update.
- Defines pre-traffic and post-traffic test functions to verify that the newly deployed code is configured correctly and your application operates as expected.
- Rolls back the deployment if CloudWatch alarms are triggered.

**Note**

If you enable gradual deployments through your AWS SAM template, a CodeDeploy resource is automatically created for you. You can view the CodeDeploy resource directly through the AWS Management Console.

**Example**

The following example demonstrates a simple version of using CodeDeploy to gradually shift customers to your newly deployed version:

```
Resources:
MyLambdaFunction:
  Type: AWS::Serverless::Function
```
Properties:
  Handler: index.handler
  Runtime: nodejs12.x
  CodeUri: s3://bucket/code.zip
  AutoPublishAlias: live

DeploymentPreference:
  Type: Canary10Percent10Minutes
  Alarms:
  - !Ref AliasErrorMetricGreaterThanZeroAlarm
  - !Ref LatestVersionErrorMetricGreaterThanZeroAlarm

Hooks:
  PreTraffic: !Ref PreTrafficLambdaFunction
  PostTraffic: !Ref PostTrafficLambdaFunction

These revisions to the AWS SAM template do the following:

- **AutoPublishAlias**: By adding this property and specifying an alias name, AWS SAM:
  - Detects when new code is being deployed, based on changes to the Lambda function’s Amazon S3 URI.
  - Creates and publishes an updated version of that function with the latest code.
  - Creates an alias with a name that you provide (unless an alias already exists), and points to the updated version of the Lambda function. Function invocations should use the alias qualifier to take advantage of this. If you aren’t familiar with Lambda function versioning and aliases, see [AWS Lambda Function Versioning and Aliases](https://docs.aws.amazon.com/lambda/latest/dg/versions-aliases.html).

- **Deployment Preference Type**: In the previous example, 10 percent of your customer traffic is immediately shifted to your new version. After 10 minutes, all traffic is shifted to the new version. However, if your pre-hook/post-hook tests fail, or if a CloudWatch alarm is triggered, CodeDeploy rolls back your deployment. The following table outlines other traffic-shifting options that are available beyond the one used earlier. Note the following:

  - **Canary**: Traffic is shifted in two increments. You can choose from predefined canary options. The options specify the percentage of traffic that’s shifted to your updated Lambda function version in the first increment, and the interval, in minutes, before the remaining traffic is shifted in the second increment.
  - **Linear**: Traffic is shifted in equal increments with an equal number of minutes between each increment. You can choose from predefined linear options that specify the percentage of traffic that’s shifted in each increment and the number of minutes between each increment.
  - **All-at-once**: All traffic is shifted from the original Lambda function to the updated Lambda function version at once.

<table>
<thead>
<tr>
<th>Deployment Preference Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canary10Percent30Minutes</td>
</tr>
<tr>
<td>Canary10Percent5Minutes</td>
</tr>
<tr>
<td>Canary10Percent10Minutes</td>
</tr>
<tr>
<td>Canary10Percent15Minutes</td>
</tr>
<tr>
<td>Linear10PercentEvery10Minutes</td>
</tr>
<tr>
<td>Linear10PercentEvery1Minute</td>
</tr>
<tr>
<td>Linear10PercentEvery2Minutes</td>
</tr>
</tbody>
</table>
### Deployment Preference Type

<table>
<thead>
<tr>
<th>Linear10PercentEvery3Minutes</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllAtOnce</td>
</tr>
</tbody>
</table>

- **Alarms**: These are CloudWatch alarms that are triggered by any errors raised by the deployment. They automatically roll back your deployment. An example is if the updated code you’re deploying is creating errors within the application. Another example is if any AWS Lambda or custom CloudWatch metrics that you specified have breached the alarm threshold.

- **Hooks**: These are pre-traffic and post-traffic test functions that run sanity checks before traffic shifting starts to the new version, and after traffic shifting completes.
  - **PreTraffic**: Before traffic shifting starts, CodeDeploy invokes the pre-traffic hook Lambda function. This Lambda function must call back to CodeDeploy and indicate success or failure. If the function fails, it aborts and reports a failure back to AWS CloudFormation. If the function succeeds, CodeDeploy proceeds to traffic shifting.
  - **PostTraffic**: After traffic shifting completes, CodeDeploy invokes the post-traffic hook Lambda function. This is similar to the pre-traffic hook, where the function must call back to CodeDeploy to report a success or failure. Use post-traffic hooks to run integration tests or other validation actions.

For more information, see [SAM Reference to Safe Deployments](#).
Monitoring serverless applications

After you deploy your serverless application to the AWS Cloud, you need to verify that it's operating properly on an ongoing basis.

Topics
• Working with logs (p. 223)

Working with logs

To simplify troubleshooting, the AWS SAM CLI has a command called `sam logs` (p. 254). This command lets you fetch logs generated by your Lambda function from the command line.

**Note**
The `sam logs` command works for all AWS Lambda functions, not just the ones you deploy using AWS SAM.

Fetching logs by AWS CloudFormation stack

When your function is a part of an AWS CloudFormation stack, you can fetch logs by using the function's logical ID:

```
sam logs -n HelloWorldFunction --stack-name mystack
```

Fetching logs by Lambda function name

Or, you can fetch logs by using the function's name:

```
sam logs -n mystack-HelloWorldFunction-1FJ8PD
```

Tailing logs

Add the `--tail` option to wait for new logs and see them as they arrive. This is helpful during deployment or when you're troubleshooting a production issue.

```
sam logs -n HelloWorldFunction --stack-name mystack --tail
```

Viewing logs for a specific time range

You can view logs for a specific time range by using the `-s` and `-e` options:

```
sam logs -n HelloWorldFunction --stack-name mystack -s '10min ago' -e '2min ago'
```

Filtering logs

Use the `--filter` option to quickly find logs that match terms, phrases, or values in your log events:
sam logs -n HelloWorldFunction --stack-name mystack --filter "error"

In the output, the AWS SAM CLI underlines all occurrences of the word "error" so you can easily locate the filter keyword within the log output.

**Error highlighting**

When your Lambda function crashes or times out, the AWS SAM CLI highlights the timeout message in red. This helps you easily locate specific executions that are timing out within a giant stream of log output.

**JSON pretty printing**

If your log messages print JSON strings, the AWS SAM CLI automatically pretty prints the JSON to help you visually parse and understand the JSON.
Publishing serverless applications using the AWS SAM CLI

To make your AWS SAM application available for others to find and deploy, you can use the AWS SAM CLI to publish it to the AWS Serverless Application Repository. To publish your application using the AWS SAM CLI, you must define it using an AWS SAM template. You also must have tested it locally or in the AWS Cloud.

Follow the instructions in this topic to create a new application, create a new version of an existing application, or update the metadata of an existing application. (What you do depends on whether the application already exists in the AWS Serverless Application Repository, and whether any application metadata is changing.) For more information about application metadata, see AWS SAM template Metadata section properties (p. 228).

Prerequisites

Before you publish an application to the AWS Serverless Application Repository using the AWS SAM CLI, you must have the following:

- The AWS SAM CLI installed. For more information, see Installing the AWS SAM CLI (p. 3). To determine whether the AWS SAM CLI is installed, run the following command:

```
sam --version
```

- A valid AWS SAM template.
- Your application code and dependencies that the AWS SAM template references.
- A semantic version, required only to share your application publicly. This value can be as simple as 1.0.
- A URL that points to your application's source code.
- A README.md 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, required only to share your application publicly.
- If your application contains any nested applications, you must have already published them to the AWS Serverless Application Repository.
- A valid Amazon Simple Storage Service (Amazon S3) bucket policy that grants the service read permissions for artifacts that you upload to Amazon S3 when you package your application. To set up this policy, do the following:

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose the name of the Amazon S3 bucket that you used to package your application.
3. Choose Permissions.
4. On the Permissions tab, under Bucket policy, choose Edit.
5. On the Edit bucket policy page, paste the following policy statement into the Policy editor. In the policy statement, make sure to use your bucket name in the Resource element and your AWS account ID in the Condition element. The expression in the Condition element ensures that AWS Serverless Application Repository has permission to access only applications from the specified AWS account. For more information about policy statements, see IAM JSON policy elements reference in the IAM User Guide.

```json
{

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

6. Choose **Save changes**.

## Publishing a new application

### Step 1: Add a Metadata section to the AWS SAM template

First, add a Metadata section to your AWS SAM template. Provide the application information to be published to the AWS Serverless Application Repository.

The following is an example Metadata section:

```yaml
Metadata:
AWS::ServerlessRepo::Application:
  Name: my-app
  Description: hello world
  Author: user1
  SpdxLicenseId: Apache-2.0
  LicenseUrl: LICENSE.txt
  ReadmeUrl: README.md
  Labels: ['tests']
  HomePageUrl: https://github.com/user1/my-app-project
  SemanticVersion: 0.0.1
  SourceCodeUrl: https://github.com/user1/my-app-project

Resources:
  HelloWorldFunction:
    Type: AWS::Lambda::Function
    Properties:
      ...
      CodeUri: source-code1
      ...
```

For more information about the Metadata section of the AWS SAM template, see [AWS SAM template Metadata section properties (p. 228)](#).

### Step 2: Package the application

Run the following AWS SAM CLI command, which uploads the application's artifacts to Amazon S3 and outputs a new template file called `packaged.yaml`:

```bash
```
Step 3: Publish the application

To publish a private version of your AWS SAM application to the AWS Serverless Application Repository, run the following AWS SAM CLI command:

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

The output of the `sam publish` command includes a link to your application on the AWS Serverless Application Repository. You can also go directly to the AWS Serverless Application Repository landing page and search for your application.

Step 4: Share the application (optional)

By default, your application is set to private, so it isn't visible to other AWS accounts. To share your application with others, you must either make it public or grant permission to a specific list of AWS accounts.

For information about sharing your application using the AWS CLI, see AWS Serverless Application Repository Resource-Based Policy Examples in the AWS Serverless Application Repository Developer Guide. For information on sharing your application using the AWS Management Console, see Sharing an Application in the AWS Serverless Application Repository Developer Guide.

Publishing a new version of an existing application

After you've published an application to the AWS Serverless Application Repository, you might want to publish a new version of it. For example, you might have changed your Lambda function code or added a new component to your application architecture.

To update an application that you've previously published, publish the application again using the same process detailed previously. In the Metadata section of the AWS SAM template file, provide the same application name that you originally published it with, but include a new SemanticVersion value.

For example, consider an application published with the name `SampleApp` and a SemanticVersion of `1.0.0`. To update that application, the AWS SAM template must have the application name `SampleApp` and a SemanticVersion of `1.0.1` (or anything other than `1.0.0`).
Additional topics

- **AWS SAM template Metadata section properties (p. 228)**

### AWS SAM template Metadata section properties

`AWS::ServerlessRepo::Application` is a metadata key that you can use to specify application information that you want published to the AWS Serverless Application Repository.

**Note**

AWS CloudFormation intrinsic functions aren't supported by the `AWS::ServerlessRepo::Application` metadata key.

### Properties

This table provides information about the properties of the Metadata section of the AWS SAM template. This section is required to publish applications to the AWS Serverless Application Repository using the AWS SAM CLI.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>String</td>
<td>TRUE</td>
<td>The name of the application.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Minimum length=1. Maximum length=140.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Pattern: &quot;[a-zA-Z0-9-]+&quot;;</td>
</tr>
<tr>
<td>Description</td>
<td>String</td>
<td>TRUE</td>
<td>The description of the application.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Minimum length=1. Maximum length=256.</td>
</tr>
<tr>
<td>Author</td>
<td>String</td>
<td>TRUE</td>
<td>The name of the author publishing the application.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Minimum length=1. Maximum length=127.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Pattern: &quot;^[a-z0-9](([a-z0-9]</td>
</tr>
<tr>
<td>SpdxLicenseId</td>
<td>String</td>
<td>FALSE</td>
<td>A valid license identifier. To view the list of valid license identifiers, see <a href="https://spdx.org/licenses/">SPDX License List</a> on the Software Package Data Exchange (SPDX) website.</td>
</tr>
<tr>
<td>LicenseUrl</td>
<td>String</td>
<td>FALSE</td>
<td>The reference to a local license file, or an Amazon S3 link to a license file, that matches the spdxLicenseId value of your application.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>An AWS SAM template file that hasn't been packaged using the <code>sam package</code> command can have a reference to a local file for this property. However, for an application to be published using the <code>sam publish</code> command, this property must be a reference to an Amazon S3 bucket.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Maximum size: 5 MB.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>You must provide a value for this property in order to make your application public. Note that you cannot</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Required</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>---------</td>
<td>----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>update this property after your application has been published. So, to add a license to an application, you must either delete it first, or publish a new application with a different name.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ReadmeUrl</td>
<td>String</td>
<td>FALSE</td>
<td>The reference to a local readme file or an Amazon S3 link to the readme file that contains a more detailed description of the application and how it works. An AWS SAM template file that hasn't been packaged using the <code>sam package</code> command can have a reference to a local file for this property. However, to be published using the <code>sam publish</code> command, this property must be a reference to an Amazon S3 bucket. Maximum size: 5 MB.</td>
</tr>
<tr>
<td>HomePageUrl</td>
<td>String</td>
<td>FALSE</td>
<td>A URL with more information about the application—for example, the location of your GitHub repository for the application.</td>
</tr>
<tr>
<td>SemanticVersion</td>
<td>String</td>
<td>FALSE</td>
<td>The semantic version of the application. For the Semantic Versioning specification, see the <a href="https://semver.org">Semantic Versioning</a> website. You must provide a value for this property in order to make your application public.</td>
</tr>
<tr>
<td>SourceCodeUrl</td>
<td>String</td>
<td>FALSE</td>
<td>A link to a public repository for the source code of your application.</td>
</tr>
</tbody>
</table>

**Use cases**

This section lists the use cases for publishing applications, along with the metadata properties that are processed for that use case. Properties that are not listed for a given use case are ignored.

- **Creating a new application** – A new application is created if there is no application in the AWS Serverless Application Repository with a matching name for an account.
  - Name
  - SpdxLicenseId
  - LicenseUrl
  - Description
  - Author
  - ReadmeUrl
  - Labels
  - HomePageUrl
• SourceCodeUrl
• SemanticVersion
• The content of the AWS SAM template (for example, any event sources, resources, and Lambda function code)

• Creating an application version – An application version is created if there is already an application in the AWS Serverless Application Repository with a matching name for an account and the SemanticVersion is changing.
  • Description
  • Author
  • ReadmeUrl
  • Labels
  • HomePageUrl
  • SourceCodeUrl
  • SemanticVersion
  • The content of the AWS SAM template (for example, any event sources, resources, and Lambda function code)

• Updating an application – An application is updated if there is already an application in the AWS Serverless Application Repository with a matching name for an account and the SemanticVersion is not changing.
  • Description
  • Author
  • ReadmeUrl
  • Labels
  • HomePageUrl

Example

The following 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
```
Example serverless applications

The following examples show you how to download, test, and deploy a number of additional serverless applications—including how to configure event sources and AWS resources.

Topics
- Process DynamoDB events (p. 231)
- Process Amazon S3 events (p. 233)

Process DynamoDB events

With this example application, you build on what you learned in the overview and the Quick Start guide, and install another example application. This application consists of a Lambda function that's invoked by a DynamoDB table event source. The Lambda function is very simple—it logs data that was passed in through the event source message.

This exercise shows you how to mimic event source messages that are passed to Lambda functions when they're invoked.

Before you begin

Make sure that you've completed the required setup in the Installing the AWS SAM CLI (p. 3).

Step 1: Initialize the application

In this section, you download the application package, 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.

   ```sh
   sam init \
   --location gh:aws-samples/cookiecutter-aws-sam-dynamodb-python \
   --no-input
   ```

   Note that gh: in the command above gets expanded to the GitHub url https://github.com/.

2. Review the contents of the directory that the command created (dynamodb_event_reader/):

   - `template.yaml` – Defines two AWS resources that the Read DynamoDB application needs: a Lambda function and a DynamoDB table. The template also defines mapping between the two resources.
   - `read_dynamodb_event/` directory – Contains the DynamoDB application code.

Step 2: Test the application locally

For local testing, use the AWS SAM CLI to generate a sample DynamoDB event and invoke the Lambda function:
The `generate-event` command creates a test event source message like the messages that are created when all components are deployed to the AWS Cloud. This event source message is piped to the Lambda function `ReadDynamoDBEvent`.

Verify that the expected messages are printed to the console, based on the source code in `app.py`.

### Step 3: Package the application

After testing your application locally, you use the AWS SAM CLI to create a deployment package, which you use to deploy the application to the AWS Cloud.

#### To create a Lambda deployment package

1. 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.

   ```bash
   aws s3 mb s3://bucketname
   ```

2. Create the deployment package by running the following `package` CLI command at the command prompt.

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

   You specify the new template file, `packaged.yaml`, when you deploy the application in the next step.

### Step 4: Deploy the application

Now that you've created the deployment package, you use it to deploy the application to the AWS Cloud. You then test the application.

#### To deploy the serverless application to the AWS Cloud

- In the AWS SAM CLI, use the `deploy` CLI command to deploy all of the resources that you defined in the template.

   ```bash
   sam deploy \
   --template-file packaged.yaml \
   --stack-name sam-app \
   --capabilities CAPABILITY_IAM \
   --region us-east-1
   ```

   In the command, the `--capabilities` parameter allows AWS CloudFormation to create an IAM role.

   AWS CloudFormation creates the AWS resources that are defined in the template. You can access the names of these resources in the AWS CloudFormation console.
To test the serverless application in the AWS Cloud

1. Open the DynamoDB console.
2. Insert a record into the table that you just created.
3. Go to the Metrics tab of the table, and choose View all CloudWatch metrics. In the CloudWatch console, choose Logs to be able to view the log output.

Next steps

The AWS SAM GitHub repository contains additional example applications for you to download and experiment with. To access this repository, see AWS SAM example applications.

Process Amazon S3 events

With this example application, you build on what you learned in the previous examples, and install a more complex application. This application consists of a Lambda function that's invoked by an Amazon S3 object upload event source. This exercise shows you how to access AWS resources and make AWS service calls through a Lambda function.

This sample serverless application processes object-creation events in Amazon S3. For each image that's uploaded to a bucket, Amazon S3 detects the object-created event and invokes a Lambda function. The Lambda function invokes Amazon Rekognition to detect text that's in the image. It then stores the results returned by Amazon Rekognition in a DynamoDB table.

Note
With this example application, you perform steps in a slightly different order than in previous examples. The reason for this is that this example requires that AWS resources are created and IAM permissions are configured before you can test the Lambda function locally. We're going to leverage AWS CloudFormation to create the resources and configure the permissions for you. Otherwise, you would need to do this manually before you can test the Lambda function locally. Because this example is more complicated, be sure that you're familiar with installing the previous example applications before executing this one.

Before you begin

Make sure that you've completed the required setup in the Installing the AWS SAM CLI (p. 3).

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.

   ```shell
   sam init \
   --location https://github.com/aws-samples/cookiecutter-aws-sam-s3-rekognition-dynamodb-python \
   --no-input
   ```

2. Review the contents of the directory that the command created (aws_sam_ocr/):
Step 2: Package the application

Before you can test this application locally, you must use the AWS SAM CLI to create a deployment package, which you use to deploy the application to the AWS Cloud. This deployment creates the necessary AWS resources and permissions that are required to test the application locally.

To create a Lambda deployment package

1. 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.

   ```bash
   aws s3 mb s3://bucketname
   ```

2. Create the deployment package by running the following `package` CLI command at the command prompt.

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

   You specify the new template file, `packaged.yaml`, when you deploy the application in the next step.

Step 3: Deploy the application

Now that you've created the deployment package, you use it to deploy the application to the AWS Cloud. You then test the application by invoking it in the AWS Cloud.

To deploy the serverless application to the AWS Cloud

- In the AWS SAM CLI, use the `deploy` command to deploy all of the resources that you defined in the template.

   ```bash
   sam deploy \
   --template-file packaged.yaml \
   --stack-name aws-sam-ocr \
   --capabilities CAPABILITY_IAM \
   --region us-east-1
   ```

   In the command, the `--capabilities` parameter allows AWS CloudFormation to create an IAM role.

   AWS CloudFormation creates the AWS resources that are defined in the template. You can access the names of these resources in the AWS CloudFormation console.
To test the serverless application in the AWS Cloud

1. Upload an image to the Amazon S3 bucket that you created for this sample application.
2. Open the DynamoDB console and find the table that was created. See the table for results returned by Amazon Rekognition.
3. Verify that the DynamoDB table contains new records that contain text that Amazon Rekognition found in the uploaded image.

Step 4: Test the application locally

Before you can test the application locally, you must first retrieve the names of the AWS resources that were created by AWS CloudFormation.

- Retrieve the Amazon S3 key name and bucket name from AWS CloudFormation. Modify the SampleEvent.json file by replacing the values for the object key, bucket name, and bucket ARN.
- Retrieve the DynamoDB table name. This name is used for the following `sam local invoke` command.

Use the AWS SAM CLI to generate a sample Amazon S3 event and invoke the Lambda function:

```
TABLE_NAME=Table name obtained from AWS CloudFormation console
sam local invoke --event SampleEvent.json
```

The `TABLE_NAME=` portion sets the DynamoDB table name. The `--event` parameter specifies the file that contains the test event message to pass to the Lambda function.

You can now verify that the expected DynamoDB records were created, based on the results returned by Amazon Rekognition.

Next steps

The AWS SAM GitHub repository contains additional example applications for you to download and experiment with. To access this repository, see AWS SAM example applications.
AWS SAM reference

AWS SAM specification

The AWS SAM specification is an open-source specification under the Apache 2.0 license. The current version of the AWS SAM specification is available in the AWS Serverless Application Model (AWS SAM) specification (p. 26).

AWS SAM templates are an extension of AWS CloudFormation templates. For the full reference for AWS CloudFormation templates, see AWS CloudFormation Template Reference.

AWS SAM CLI command reference

The AWS SAM CLI is a command line tool that operates on an AWS SAM template and application code. With the AWS SAM CLI, you can invoke Lambda functions locally, create a deployment package for your serverless application, deploy your serverless application to the AWS Cloud, and so on.

You can use the AWS SAM CLI commands to develop, test, and deploy your serverless applications to the AWS Cloud. The following are some examples of AWS SAM CLI commands:

• `sam init` – If you’re a first-time AWS SAM CLI user, you can run the `sam init` command without any parameters to create a Hello World application. The command generates a preconfigured AWS SAM template and example application code in the language that you choose.
• `sam local invoke` and `sam local start-api` – Use these commands to test your application code locally, before deploying it to the AWS Cloud.
• `sam logs` – Use this command to fetch logs generated by your Lambda function. This can help you with testing and debugging your application after you’ve deployed it to the AWS Cloud.
• `sam package` – Use this command to bundle your application code and dependencies into a "deployment package". The deployment package is needed to upload your application to the AWS Cloud.
• `sam deploy` – Use this command to deploy your serverless application to the AWS Cloud. It creates the AWS resources and sets permissions and other configurations that are defined in the AWS SAM template.

For instructions about installing the AWS SAM CLI, see Installing the AWS SAM CLI (p. 3).

AWS SAM policy templates

AWS SAM allows you to choose from a list of policy templates to scope the permissions of your Lambda functions to the resources that are used by your application.

Topics

• AWS Serverless Application Model (AWS SAM) specification (p. 26)
• AWS SAM CLI command reference (p. 237)
• AWS SAM policy templates (p. 261)
AWS SAM CLI command reference

This section is the reference for the AWS SAM CLI commands. For instructions about installing the AWS SAM CLI, see Installing the AWS SAM CLI (p. 3).

Topics
- sam build (p. 237)
- sam deploy (p. 241)
- sam init (p. 244)
- sam local generate-event (p. 247)
- sam local invoke (p. 248)
- sam local start-api (p. 250)
- sam local start-lambda (p. 252)
- sam logs (p. 254)
- sam package (p. 255)
- sam publish (p. 257)
- sam validate (p. 258)

sam build

Builds a serverless application and prepares it for subsequent steps in your workflow, like locally testing the application or deploying it to the AWS Cloud. If you provide a RESOURCE_Logical_ID, then AWS SAM builds only that resource. To build a resource of a nested application or stack, you can provide the application or stack logical ID along with the resource logical ID using the format StackLogicalId/ResourceLogicalId.

The sam build command processes your AWS SAM template file, application code, and any applicable language-specific files and dependencies. The command also copies build artifacts in the format and location expected for subsequent steps in your workflow. You specify dependencies in a manifest file that you include in your application, such as requirements.txt for Python functions, or package.json for Node.js functions.

The format of your application's build artifacts depends on its package type. You specify your AWS Lambda function's package type with the PackageType property. The options are:

- **Zip** – A .zip file archive, which contains your application code and its dependencies. If you package your code as a .zip file archive, you must specify a Lambda runtime for your function.
- **Image** – A container image, which includes the base operating system, runtime, and extensions, in addition to your application code and its dependencies.

For more information about Lambda package types, see Lambda deployment packages in the AWS Lambda Developer Guide.

If a resource includes a Metadata resource attribute with a BuildMethod entry, sam build builds that resource according to the value of the BuildMethod entry. Valid values for BuildMethod are 1) One of the identifiers for a Lambda runtime, or 2) The makefile identifier.

- **Lambda runtime identifier** – Build the resource against a Lambda runtime. For the list of supported runtime identifiers, see Lambda runtimes in the AWS Lambda Developer Guide.
- **makefile** identifier – Run the commands of the build target for the resource. In this case, your makefile must be named Makefile and include a build target named `build-resource-logical-id`.

To build layers and custom runtimes, you can also use the `Metadata` resource attribute with a `BuildMethod` entry. For information about building layers, see Building layers (p. 202). For information about building custom runtimes, see Building custom runtimes (p. 204).

For serverless function resources that have the `Image` package type, use the `Metadata` resource attribute to configure Docker image settings that are required to build a container image. For more information about building container images, see Building a container image (p. 200).

For a complete example that uses this command, including locally testing and deploying to the AWS Cloud, see Tutorial: Deploying a Hello World application (p. 14). The `sam build` command is part of Step 2: Build your application (p. 17).

**Usage:**

```
sam build [OPTIONS] [RESOURCE_LOGICAL_ID]
```

**Examples:**

To use these commands, update your SAM template to specify the path to your function's source code in the resource's Code or CodeUri property.

To build on your workstation, run this command in the directory containing your SAM template. Built artifacts are written to the .aws-sam/build directory.

```
$ sam build
```

To build inside a Lambda-like Docker container

```
$ sam build --use-container
```

To build with environment variables passed to the build container from the command line

```
$ sam build --use-container --container-env-var Function1.GITHUB_TOKEN=<token1> --container-env-var GLOBAL_ENV_VAR=<global-token>
```

To build with environment variables passed to the build container from a file

```
$ sam build --use-container --container-env-var-file <env-file.json>
```

To build and run your functions locally

```
$ sam build && sam local invoke
```

To build and package for deployment

```
$ sam build && sam package --s3-bucket <bucketname>
```

To build the 'MyFunction' resource

```
$ sam build MyFunction
```

To build the 'MyFunction' resource of the 'MyNestedStack' nested stack

```
$ sam build MyNestedStack/MyFunction
```

**Options:**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-b, --build-dir DIRECTORY</td>
<td>The path to a directory where the built artifacts are stored. This directory and all of its content are removed with this option.</td>
</tr>
<tr>
<td>-s, --base-dir DIRECTORY</td>
<td>Resolves relative paths to the Lambda function's source code with respect to this directory. Use this option if the AWS SAM template and</td>
</tr>
</tbody>
</table>
### Option | Description
--- | ---
None | your source code aren't in the same directory. By default, relative paths are resolved with respect to the template's location.

- **-u, --use-container**<br>If your functions depend on packages that have natively compiled dependencies, use this option to build your function inside a Lambda-like Docker container.

- **-e, --container-env-var TEXT**<br>Environment variables to pass to the build container. You can specify this option multiple times. Each instance of this option takes a key-value pair, where the key is the resource and environment variable, and the value is the environment variable's value. For example: `--container-env-var Function1.GITHUB_TOKEN=TOKEN1 --container-env-var Function2.GITHUB_TOKEN=TOKEN2`.

- **-ef, --container-env-var-file PATH**<br>The path and file name of a JSON file that contains values for the container's environment variables. For more information about container environment variable files, see Container environment variable file (p. 200).

- **-m, --manifest PATH**<br>The path to a custom dependency manifest file (for example, package.json) to use instead of the default.

- **-t, --template-file, --template PATH**<br>The path and file name of AWS SAM template file [default: template.[yaml|yml]].

- **--parameter-overrides**<br>(Optional) A string that contains AWS CloudFormation parameter overrides encoded as key-value pairs. Uses the same format as the AWS Command Line Interface (AWS CLI). For example: `{"ParameterKey=KeyPairName, ParameterValue=MyKey ParameterKey=InstanceType, ParameterValue=t1.micro"}`.

- **--skip-pull-image**<br>Specifies whether the command should skip pulling down the latest Docker image for the Lambda runtime.

- **--docker-network TEXT**<br>Specifies the name or ID of an existing Docker network that Lambda Docker containers should connect to, along with the default bridge network. If not specified, the Lambda containers connect only to the default bridge Docker network.

- **--parallel**<br>Enabled parallel builds. Use this option to build your AWS SAM template's functions and layers in parallel. By default, the functions and layers are built in sequence.

- **--cached**<br>Enable cached builds. Use this option to reuse build artifacts that haven't changed from previous builds. AWS SAM evaluates whether you've changed any files in your project directory. **Note:** AWS SAM doesn't evaluate whether you've changed third-party modules that your project depends on, where you haven't provided a specific version. For example, if your Python function includes a requirements.txt file with the entry `requests=1.x`, and the latest request module version changes from 1.1 to 1.2, then AWS SAM doesn't pull the latest version until you run a non-cached build.

- **--cache-dir**<br>The directory where the cache artifacts are stored when --cached is specified. The default cache directory is .aws-sam/cache.

- **--profile TEXT**<br>The specific profile from your credential file that gets AWS credentials.
**Option** | **Description**
---|---
--region TEXT | The AWS Region to deploy to. For example, us-east-1.
--config-file PATH | The path and file name of the configuration file containing default parameter values to use. The default value is "samconfig.toml" in the root of the project directory. For more information about configuration files, see AWS SAM CLI configuration file (p. 259).
--config-env TEXT | The environment name specifying the default parameter values in the configuration file to use. The default value is "default". For more information about configuration files, see AWS SAM CLI configuration file (p. 259).
--debug | Turns on debug logging to print debug messages that the AWS SAM CLI generates, and to display timestamps.
--help | Shows this message and exits.

**Examples**

**Building a resource using a Lambda runtime identifier**

Here's an example AWS SAM template showing how to build a resource using a Lambda runtime identifier:

```
Resources:
  MyLayer:
    Type: AWS::Serverless::LayerVersion
    Properties:
      ContentUri: my_layer
      CompatibleRuntimes:
      - python3.6
    Metadata:
      BuildMethod: python3.6
```

With this template, the following command will build the **MyLayer** resource against the Python 3.6 runtime environment:

```
sam build MyLayer
```

**Building a resource using the **makefile** identifier**

Here's an example AWS SAM showing how to build a resource using the **makefile** identifier:

```
Resources:
  MyLayer:
    Type: AWS::Serverless::LayerVersion
    Properties:
      ContentUri: my_layer
      CompatibleRuntimes:
      - python3.8
    Metadata:
      BuildMethod: makefile
```

This is an example of an associated makefile. The file must be named Makefile, and include a build target with the commands you want to run:
With this template and makefile, the following command will execute the commands for the `build-MyLayer` target:

```bash
sam build MyLayer
```

**Passing environment variables to a build container**

Here's an example showing how to pass environment variables to a build container using a file.

First, create a file named `env.json` with the following contents:

```json
{
  "MyFunction1": {
    "GITHUB_TOKEN": "TOKEN1"
  },
  "MyFunction2": {
    "GITHUB_TOKEN": "TOKEN2"
  }
}
```

Then, run the following command:

```bash
sam build --use-container --container-env-var-file env.json
```

For more information about container environment variable files, see [Container environment variable file](p. 200).

**sam deploy**

Deploys an AWS SAM application.

This command comes with a guided interactive mode, which you can enable by specifying the `--guided` parameter. The interactive mode walks you through the parameters required for deployment, provides default options, and optionally saves these options in a configuration file in your project directory. When you perform subsequent deployments of your application using `sam deploy`, the AWS SAM CLI retrieves the required parameters from the configuration file.

Objects declared in the `Parameters` section of the AWS SAM template file appear as additional interactive mode prompts. You're prompted to provide values for each parameter. For examples of these objects and the corresponding prompts, see the Examples of additional interactive prompts section later in this topic.

Serverless applications that you configure with code signing generate more interactive mode prompts. You're asked whether you want your code to be signed, and if so, you're prompted to enter signing profile names and owners. For examples of these prompts, see the Examples of additional interactive prompts section later in this topic.

For more information about settings that are optionally stored when specifying the `--guided` parameter, see [AWS SAM CLI configuration file](p. 259).

Deploying AWS Lambda functions through AWS CloudFormation requires an Amazon Simple Storage Service (Amazon S3) bucket for the Lambda deployment package. The AWS SAM CLI creates and manages this Amazon S3 bucket for you.
Usage:

```bash
sam deploy [OPTIONS] [ARGS]...
```

Options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-g, --guided</td>
<td>Specify this flag to have AWS SAM use prompts to guide you through the deployment.</td>
</tr>
<tr>
<td>-t, --template-file, --template PATH</td>
<td>The path and file name where your AWS SAM template is located. Default: template.[yaml</td>
</tr>
<tr>
<td>--stack-name TEXT</td>
<td>(Required) The name of the AWS CloudFormation stack that you're deploying to. If you specify an existing stack, the command updates the stack. If you specify a new stack, the command creates it.</td>
</tr>
<tr>
<td>--s3-bucket TEXT</td>
<td>The URI of the Amazon S3 bucket where this command uploads your AWS CloudFormation template. This is required to deploy templates that are larger than 51,200 bytes.</td>
</tr>
<tr>
<td>--s3-prefix TEXT</td>
<td>The prefix added to the names of the artifacts that are uploaded to the Amazon S3 bucket. The prefix name is a path name (folder name) for the Amazon S3 bucket.</td>
</tr>
<tr>
<td>--image-repository TEXT</td>
<td>The name of the Amazon Elastic Container Registry (Amazon ECR) repository where this command uploads your function's image. Required for functions declared with the Image package type.</td>
</tr>
<tr>
<td>--signing-profiles LIST</td>
<td>(Optional) The list of signing profiles to sign your deployment packages with. This parameter takes a list of key-value pairs, where the key is the name of the function or layer to sign, and the value is the signing profile, with an optional profile owner delimited with :. For example, FunctionNameToSign=SigningProfileName1 LayerNameToSign=SigningProfileName2:SigningProfileOwner.</td>
</tr>
<tr>
<td>--capabilities LIST</td>
<td>A list of capabilities that you must specify to allow AWS CloudFormation to create certain stacks. Some stack templates 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 stacks, you must explicitly acknowledge their capabilities by specifying this parameter. The only valid values are CAPABILITY_IAM and CAPABILITY_NAMED_IAM. If you have IAM resources, you can specify either capability. If you have IAM resources with custom names, you must specify CAPABILITY_NAMED_IAM. If you don't specify this parameter, this operation returns an InsufficientCapabilities error.</td>
</tr>
<tr>
<td>--region TEXT</td>
<td>The AWS Region to deploy to. For example, us-east-1.</td>
</tr>
<tr>
<td>--profile TEXT</td>
<td>The specific profile from your credential file that gets AWS credentials.</td>
</tr>
<tr>
<td>--kms-key-id TEXT</td>
<td>The ID of an AWS Key Management Service (AWS KMS) key used to encrypt artifacts that are at rest in the Amazon S3 bucket.</td>
</tr>
<tr>
<td>--force-upload</td>
<td>Specify this flag to upload artifacts even if they match existing artifacts in the Amazon S3 bucket. Matching artifacts are overwritten.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>--no-execute-changeset</code></td>
<td>Indicates whether to execute the changeset. Specify this flag if you want to view your stack changes before executing the changeset. This command creates an AWS CloudFormation changeset and then exits without executing the changeset. To execute the changeset, run the same command without this flag.</td>
</tr>
<tr>
<td><code>--role-arn TEXT</code></td>
<td>The Amazon Resource Name (ARN) of an IAM role that AWS CloudFormation assumes when executing the changeset.</td>
</tr>
<tr>
<td>`--fail-on-empty-changeset</td>
<td>--no-fail-on-empty-changeset`</td>
</tr>
<tr>
<td>`--confirm-changeset</td>
<td>--no-confirm-changeset`</td>
</tr>
<tr>
<td><code>--use-json</code></td>
<td>Output JSON for the AWS CloudFormation template. The default output is YAML.</td>
</tr>
<tr>
<td><code>--metadata</code></td>
<td>(Optional) A map of metadata to attach to all artifacts that are referenced in your template.</td>
</tr>
<tr>
<td><code>--notification-arns LIST</code></td>
<td>A list of Amazon Simple Notification Service (Amazon SNS) topic ARNs that AWS CloudFormation associates with the stack.</td>
</tr>
<tr>
<td><code>--tags LIST</code></td>
<td>A list of tags to associate with the stack that is created or updated. AWS CloudFormation also propagates these tags to resources in the stack if the resource supports it.</td>
</tr>
<tr>
<td><code>--parameter-overrides</code></td>
<td>A string that contains AWS CloudFormation parameter overrides encoded as key-value pairs. Use the same format as the AWS Command Line Interface (AWS CLI). For example, ParameterKey=ParameterValue InstanceType=t1.micro.</td>
</tr>
<tr>
<td><code>--config-file PATH</code></td>
<td>The path and file name of the configuration file containing default parameter values to use. The default value is &quot;samconfig.toml&quot; in the root of the project directory. For more information about configuration files, see AWS SAM CLI configuration file (p. 259).</td>
</tr>
<tr>
<td><code>--config-env TEXT</code></td>
<td>The environment name specifying the default parameter values in the configuration file to use. The default value is &quot;default&quot;. For more information about configuration files, see AWS SAM CLI configuration file (p. 259).</td>
</tr>
<tr>
<td><code>--no-progressbar</code></td>
<td>Do not display a progress bar when uploading artifacts to Amazon S3.</td>
</tr>
<tr>
<td><code>--debug</code></td>
<td>Turns on debug logging to print debug message generated by the AWS SAM CLI and display timestamps.</td>
</tr>
<tr>
<td><code>--help</code></td>
<td>Shows this message and exits.</td>
</tr>
</tbody>
</table>
Examples

Parameters

Here is an example object declared in the Parameters section, and the corresponding prompt that appears when using `sam deploy --guided`.

**AWS SAM template:**

```
Parameters:
  MyPar:
    Type: String
    Default: MyParVal
```

**Corresponding `sam deploy --guided` prompt:**

```
Parameter MyPar [MyParVal]:
```

Code signing

Here is an example function configured with code signing.

**AWS SAM template:**

```
Resources:
  HelloWorld:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: hello_world/
      Handler: app.lambda_handler
      Runtime: python3.7
```

**Corresponding `sam deploy --guided` prompts:**

```
#Found code signing configurations in your function definitions
Do you want to sign your code? [Y/n]:
#Please provide signing profile details for the following functions & layers
#Signing profile details for function 'HelloWorld'
Signing Profile Name:
Signing Profile Owner Account ID (optional):
#Signing profile details for layer 'MyLayer', which is used by functions {'HelloWorld'}
Signing Profile Name:
Signing Profile Owner Account ID (optional):
```

**sam init**

Initializes a serverless application with an AWS SAM template. The template provides a folder structure for your AWS Lambda functions, and is connected to event sources such as APIs, Amazon Simple Storage Service (Amazon S3) buckets, or Amazon DynamoDB tables. This application includes everything that you need to get started and to eventually extend it into a production-scale application.

For some sample applications, you can choose the package type of the application, either Zip or Image. For more information about Lambda package types, see Lambda deployment packages in the AWS Lambda Developer Guide.
**Usage:**

```
sam init [OPTIONS]
```

**Note**
With AWS SAM version 0.30.0 or later, you can initialize your application using one of two modes: 1) interactive workflow, or 2) providing all required parameters.

- **Interactive workflow:** Through the interactive initialize workflow, you can input either 1) your project name, preferred runtime, and template file, or 2) the location of a custom template.

- **Providing parameters:** Provide all required parameters.

If you provide a subset of required parameters, you are prompted for the additional required information.

**Examples:**

Initializes a new SAM project with required parameters passed as parameters

```
sam init --runtime python3.7 --dependency-manager pip --app-template hello-world --name sam-app
```

Initializes a new SAM project using custom template in a Git/Mercurial repository

```bash
# gh being expanded to github url
sam init --location gh:aws-samples/cookiecutter-aws-sam-python

sam init --location git+ssh://git@github.com/aws-samples/cookiecutter-aws-sam-python.git

sam init --location hg+ssh://hg@bitbucket.org/repo/template-name
```

# Initializes a new SAM project using custom template in a Zipfile

```
sam init --location /path/to/template.zip

sam init --location https://example.com/path/to/template.zip
```

# Initializes a new SAM project using cookiecutter template in a local path

```
sam init --location /path/to/template/folder
```

**Options:**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--no-interactive</td>
<td>Disable interactive prompting for init parameters, and fail if any required values are missing.</td>
</tr>
<tr>
<td>-l, --location TEXT</td>
<td>The template or application location (Git, Mercurial, HTTP/HTTPS, .zip file, path).</td>
</tr>
</tbody>
</table>

This parameter is required if --no-interactive is specified and --runtime, --name, and --app-template are not provided.

For Git repositories, you must use the location of the root of the repository.

For local paths, the template must be in either .zip file or **Cookiecutter** format.
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--package-type [Zip</td>
<td>The package type of the example application. Zip creates a .zip file archive, and Image creates a container image.</td>
</tr>
<tr>
<td>Image]</td>
<td></td>
</tr>
<tr>
<td>-r, --runtime [python2.7</td>
<td>The Lambda runtime of your application. This option applies only when the package type is Zip.</td>
</tr>
<tr>
<td>2.7</td>
<td>ruby2.5</td>
</tr>
<tr>
<td></td>
<td>This parameter is required if --no-interactive is specified, --image-type is specified as Zip, and --location is not specified.</td>
</tr>
<tr>
<td>--base-image [amazon/</td>
<td>The base image of your application. This option applies only when the package type is Image.</td>
</tr>
<tr>
<td>nodejs14.x-base</td>
<td>This parameter is required if --no-interactive is specified, --image-type is specified as Image, and --location is not specified.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>-d, --dependency-</td>
<td>The dependency manager of your Lambda runtime.</td>
</tr>
<tr>
<td>manager [gradle</td>
<td>maven</td>
</tr>
<tr>
<td>-o, --output-dir PATH</td>
<td>The location where the initialized application is output.</td>
</tr>
<tr>
<td>-n, --name TEXT</td>
<td>The name of your project to be generated as a directory. This parameter is required if --no-interactive is specified and --location is not provided.</td>
</tr>
</tbody>
</table>
Option | Description
--- | ---
|--app-template TEXT | The identifier of the managed application template that you want to use. If you're not sure, call `sam init` without options for an interactive workflow.
| | This parameter is required if `--no-interactive` is specified and `--location` is not provided.
| | This parameter is available only in AWS SAM CLI version 0.30.0 and later. Specifying this parameter with an earlier version results in an error.
|--no-input | Disables Cookiecutter prompting and accepts the vcfdefault values that are defined in the template configuration.
|--extra-content | Override any custom parameters in the template's `cookiecutter.json` configuration, for example, `{"customParam1": "customValue1", "customParam2": "customValue2"}`
|--config-file PATH | The path and file name of the configuration file containing default parameter values to use. The default value is `samconfig.toml` in the root of the project directory. For more information about configuration files, see AWS SAM CLI configuration file (p. 259).
|--config-env TEXT | The environment name specifying the default parameter values in the configuration file to use. The default value is "default". For more information about configuration files, see AWS SAM CLI configuration file (p. 259).
|--debug | Turns on debug logging to print debug messages that the AWS SAM CLI generates, and to display timestamps.
-h, --help | Shows this message and exits.

**sam local generate-event**

Generates sample payloads from different event sources, such as Amazon S3, Amazon API Gateway, and Amazon SNS. These payloads contain the information that the event sources send to your Lambda functions.

**Usage:**

```
sam local generate-event [OPTIONS] COMMAND [ARGS]...
```

**Examples:**

Generate the event that S3 sends to your Lambda function when a new object is uploaded

```
sam local generate-event s3 [put/delete]
```

# You can even customize the event by adding parameter flags. To find which flags apply to your command, run:

```
sam local generate-event s3 [put/delete] --help
```

# Then you can add in those flags that you wish to customize using

```
sam local generate-event s3 [put/delete] --bucket <bucket> --key <key>
```
# After you generate a sample event, you can use it to test your Lambda function locally

```bash
sam local generate-event s3 [put/delete] --bucket <bucket> --key <key> | sam local invoke -
```

Options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--config-file</code></td>
<td>The path and file name of the configuration file containing default parameter values to use. The default value is &quot;samconfig.toml&quot; in the root of the project directory. For more information about configuration files, see AWS SAM CLI configuration file (p. 259).</td>
</tr>
<tr>
<td><code>--config-env</code></td>
<td>The environment name specifying the default parameter values in the configuration file to use. The default value is &quot;default&quot;. For more information about configuration files, see AWS SAM CLI configuration file (p. 259).</td>
</tr>
<tr>
<td><code>--help</code></td>
<td>Shows this message and exits.</td>
</tr>
</tbody>
</table>

Commands:

- alexa-skills-kit
- alexa-smart-home
- apigateway
- batch
- cloudformation
- cloudfront
- cloudwatch
- codecommit
- codepipeline
- cognito
- config
- dynamodb
- kinesis
- lex
- rekognition
- s3
- ses
- sns
- sqs
- stepfunctions

**sam local invoke**

Invokes a local AWS Lambda function once and quits after invocation completes. To invoke a function of a nested application or stack, you can provide the application or stack logical ID along with the function logical ID using the format `StackLogicalId/FunctionLogicalId`. 
The `sam local invoke` command is useful for developing serverless functions that handle asynchronous events, such as Amazon Simple Storage Service (Amazon S3) or Amazon Kinesis events. It can also be useful if you want to compose a script of test cases. You can pass in the event body using the `--event` parameter. The runtime output (for example, logs) is output to `stderr`, and the Lambda function result is output to `stdout`.

**Note**
If there is more than one function defined in your AWS SAM template, you must provide the `FUNCTION_LOGICAL_ID` of the function you want to invoke.

**Usage:**

```bash
sam local invoke [OPTIONS] [FUNCTION_LOGICAL_ID]
```

**Options:**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-e, <code>--event PATH</code></td>
<td>The JSON file that contains event data that's passed to the Lambda function when it's invoked. If you don't specify this option, no event is assumed. To input JSON from stdin, you must pass in the value <code>-'</code>.</td>
</tr>
<tr>
<td><code>--no-event</code></td>
<td>Invokes the function with an empty event.</td>
</tr>
<tr>
<td>-t, <code>--template PATH</code></td>
<td>The AWS SAM template file [default: template.[yaml</td>
</tr>
<tr>
<td>-n, <code>--env-vars PATH</code></td>
<td>The JSON file that contains values for the Lambda function's environment variables. For more information about environment variable files, see Environment variable file (p. 207).</td>
</tr>
<tr>
<td><code>--parameter-overrides</code></td>
<td>(Optional) A string that contains AWS CloudFormation parameter overrides encoded as key-value pairs. Uses the same format as the AWS Command Line Interface (AWS CLI). For example: 'ParameterKey=KeyPairName,ParameterValue=MyKey ParameterKey=InstanceType,ParameterValue=t1.micro'.</td>
</tr>
<tr>
<td>-d, <code>--debug-port TEXT</code></td>
<td>When specified, starts the Lambda function container in debug mode and exposes this port on the local host.</td>
</tr>
<tr>
<td><code>--debugger-path TEXT</code></td>
<td>The host path to a debugger that's mounted into the Lambda container.</td>
</tr>
<tr>
<td><code>--debug-args TEXT</code></td>
<td>Additional arguments to pass to the debugger.</td>
</tr>
<tr>
<td>-v, <code>--docker-volume-basedir TEXT</code></td>
<td>The location of the base directory where the AWS SAM file exists. If Docker is running on a remote machine, you must mount the path where the AWS SAM file exists on the Docker machine and modify this value to match the remote machine.</td>
</tr>
<tr>
<td><code>--docker-network TEXT</code></td>
<td>The name or ID of an existing Docker network that Lambda Docker containers should connect to, along with the default bridge network. If this isn't specified, the Lambda containers connect only to the default bridge Docker network.</td>
</tr>
<tr>
<td><code>--container-env-vars</code></td>
<td>(Optional) Pass environment variables to the Lambda function image container when debugging locally.</td>
</tr>
<tr>
<td>-l, <code>--log-file TEXT</code></td>
<td>The log file to send runtime logs to.</td>
</tr>
<tr>
<td><code>--layer-cache-basedir DIRECTORY</code></td>
<td>Specifies the location of the base directory where the layers that your template uses are downloaded to.</td>
</tr>
</tbody>
</table>
sam local start-api

Allows you to run your serverless application locally for quick development and testing. When you run this command in a directory that contains your serverless functions and your AWS SAM template, it creates a local HTTP server that hosts all of your functions.

When it’s accessed (through a browser, CLI, and so on), it starts a Docker container locally to invoke the function. It reads the CodeUri property of the AWS::Serverless::Function resource to find the path in your file system that contains the Lambda function code. This could be the project's root directory for interpreted languages like Node.js and Python, or a build directory that stores your compiled artifacts or a Java Archive (JAR) file.

If you’re using an interpreted language, local changes are available immediately in the Docker container on every invoke. For more compiled languages or projects that require complex packing support, we recommend that you run your own building solution, and point AWS SAM to the directory or file that contains the build artifacts.

To see an end-to-end example that uses this command, see Tutorial: Deploying a Hello World application (p. 14). The sam local start-api command is part of Step 4: (Optional) Test your application locally (p. 20).

Usage:

```
sam local start-api [OPTIONS]
```

Options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--host TEXT</td>
<td>The local hostname or IP address to bind to (default: '127.0.0.1').</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>-p, --port INTEGER</td>
<td>The local port number to listen on (default: '3000').</td>
</tr>
</tbody>
</table>
| -s, --static-dir TEXT          | Any static asset (for example, CSS/JavaScript/HTML) files located in this directory are presented at /.
| -t, --template PATH           | The AWS SAM template file [default: template.[yaml|yml]].                    |
| -n, --env-vars PATH            | The JSON file that contains values for the Lambda function's environment variables. |
| --parameter-overrides          | Optional. A string that contains AWS CloudFormation parameter overrides encoded as key-value pairs. Use the same format as the AWS CLI—for example, 'ParameterKey=KeyPairName, ParameterValue=MyKey ParameterKey=InstanceType, ParameterValue=t1.micro'. |
| -d, --debug-port TEXT          | When specified, starts the Lambda function container in debug mode and exposes this port on the local host. |
| --debugger-path TEXT           | The host path to a debugger that will be mounted into the Lambda container. |
| --debug-args TEXT              | Additional arguments to be passed to the debugger.                         |
| --warm-containers [EAGER | LAZY] | Optional. Specifies how AWS SAM CLI manages containers for each function. Two options are available:  
| EAGER: Containers for all functions are loaded at startup and persist between invocations.  
| LAZY: Containers are only loaded when each function is first invoked. Those containers persist for additional invocations. |
| --debug-function               | Optional. Specifies the Lambda function to apply debug options to when --warm-containers is specified. This parameter applies to --debug-port, --debugger-path, and --debug-args. |
| -v, --docker-volume-basedir TEXT | The location of the base directory where the AWS SAM file exists. If Docker is running on a remote machine, you must mount the path where the AWS SAM file exists on the Docker machine, and modify this value to match the remote machine. |
| --docker-network TEXT          | The name or ID of an existing Docker network that the Lambda Docker containers should connect to, along with the default bridge Docker network. If this isn't specified, the Lambda containers only connect to the default bridge Docker network. |
| --container-env-vars           | Optional. Pass environment variables to image container when locally debugging. |
| -l, --log-file TEXT            | The log file to send runtime logs to.                                       |
| --layer-cache-basedir DIRECTORY| Specifies the location basedir where the Layers your template uses are downloaded to. |
| --skip-pull-image              | Specifies whether the CLI should skip pulling down the latest Docker image for the Lambda runtime. |
Option | Description
---|---
--force-image-build | Specifies whether CLI should rebuild the image used for invoking functions with layers.
--profile TEXT | The specific profile from your credential file that gets AWS credentials.
--region TEXT | The AWS Region to deploy to. For example, us-east-1.
--config-file PATH | The path and file name of the configuration file containing default parameter values to use. The default value is "samconfig.toml" in the root of the project directory. For more information about configuration files, see AWS SAM CLI configuration file (p. 259).
--config-env TEXT | The environment name specifying the default parameter values in the configuration file to use. The default value is "default". For more information about configuration files, see AWS SAM CLI configuration file (p. 259).
--debug | Turns on debug logging to print debug message generated by the AWS SAM CLI and display timestamps.
--help | Shows this message and exits.

**sam local start-lambda**

Enables you to programmatically invoke your Lambda function locally by using the AWS CLI or SDKs. This command starts a local endpoint that emulates AWS Lambda. You can run your automated tests against this local Lambda endpoint. When you send an invoke to this endpoint using the AWS CLI or SDK, it locally executes the Lambda function that's specified in the request.

**Usage:**

```
sam local start-lambda [OPTIONS]
```

**Examples:**

```
# SETUP
# -----
# Start the local Lambda endpoint by running this command in the directory that contains your AWS SAM template.

sam local start-lambda

# USING AWS CLI
# ------------
# Then, you can invoke your Lambda function locally using the AWS CLI

aws lambda invoke --function-name "HelloWorldFunction" --endpoint-url "http://127.0.0.1:3001" --no-verify-ssl out.txt

# USING AWS SDK
# ------------
# You can also use the AWS SDK in your automated tests to invoke your functions programmatically.
# Here is a Python example:
#
#     self.lambda_client = boto3.client('lambda',
```

252
AWS Serverless Application Model Developer Guide

sam local start-lambda

```python
# endpoint_url="http://127.0.0.1:3001",
# use_ssl=False,
# verify=False,
# config=Config(signature_version=UNSIGNED,
#                  read_timeout=0,
#                  retries={'max_attempts': 0}))
# self.lambda_client.invoke(FunctionName="HelloWorldFunction")
```

### Options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--host TEXT</code></td>
<td>The local hostname or IP address to bind to (default: '127.0.0.1').</td>
</tr>
<tr>
<td><code>-p, --port INTEGER</code></td>
<td>The local port number to listen on (default: '3001').</td>
</tr>
<tr>
<td><code>-t, --template PATH</code></td>
<td>The AWS SAM template file [default: template.[yaml</td>
</tr>
<tr>
<td><code>-n, --env-vars PATH</code></td>
<td>The JSON file that contains values for the Lambda function's environment variables.</td>
</tr>
<tr>
<td><code>--parameter-oversrides</code></td>
<td>Optional. A string that contains AWS CloudFormation parameter overrides encoded as key-value pairs. Use the same format as the AWS CLI—for example, 'ParameterKey=KeyPairName, ParameterValue=MyKey ParameterKey=InstanceType,ParameterValue=t1.micro'.</td>
</tr>
<tr>
<td><code>-d,--debug-port TEXT</code></td>
<td>When specified, starts the Lambda function container in debug mode, and exposes this port on the local host.</td>
</tr>
<tr>
<td><code>--debugger-path TEXT</code></td>
<td>The host path to a debugger to be mounted into the Lambda container.</td>
</tr>
<tr>
<td><code>--debug-args TEXT</code></td>
<td>Additional arguments to be passed to the debugger.</td>
</tr>
<tr>
<td>`--warm-containers [EAGER</td>
<td>LAZY]`</td>
</tr>
<tr>
<td><code>--debug-function</code></td>
<td>Optional. Specifies the Lambda function to apply debug options to when <code>--warm-containers</code> is specified. This parameter applies to <code>--debug-port</code>, <code>--debugger-path</code>, and <code>--debug-args</code>.</td>
</tr>
<tr>
<td><code>-v, --docker-volume-basedir TEXT</code></td>
<td>The location of the base directory where the AWS SAM file exists. If Docker is running on a remote machine, you must mount the path where the AWS SAM file exists on the Docker machine, and modify this value to match the remote machine.</td>
</tr>
<tr>
<td><code>--docker-network TEXT</code></td>
<td>The name or ID of an existing Docker network that Lambda Docker containers should connect to, along with the default bridge network. If this is specified, the Lambda containers only connect to the default bridge Docker network.</td>
</tr>
<tr>
<td><code>--container-env-vars</code></td>
<td>Optional. Pass environment variables to image container when locally debugging.</td>
</tr>
</tbody>
</table>
### Option Description

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-l, --log-file TEXT</td>
<td>The log file to send runtime logs to.</td>
</tr>
<tr>
<td>--layer-cache-basedir DIRECTORY</td>
<td>Specifies the location basedir where the layers your template uses are downloaded to.</td>
</tr>
<tr>
<td>--skip-pull-image</td>
<td>Specifies whether the CLI should skip pulling down the latest Docker image for the Lambda runtime.</td>
</tr>
<tr>
<td>--force-image-build</td>
<td>Specify whether the CLI should rebuild the image used for invoking functions with layers.</td>
</tr>
<tr>
<td>--profile TEXT</td>
<td>The specific profile from your credential file that gets AWS credentials.</td>
</tr>
<tr>
<td>--region TEXT</td>
<td>The AWS Region to deploy to. For example, us-east-1.</td>
</tr>
<tr>
<td>--config-file PATH</td>
<td>The path and file name of the configuration file containing default parameter values to use. The default value is &quot;samconfig.toml&quot; in the root of the project directory. For more information about configuration files, see AWS SAM CLI configuration file (p. 259).</td>
</tr>
<tr>
<td>--config-env TEXT</td>
<td>The environment name specifying the default parameter values in the configuration file to use. The default value is &quot;default&quot;. For more information about configuration files, see AWS SAM CLI configuration file (p. 259).</td>
</tr>
<tr>
<td>--debug</td>
<td>Turns on debug logging to print debug message generated by the AWS SAM CLI and display timestamps.</td>
</tr>
<tr>
<td>--help</td>
<td>Shows this message and exits.</td>
</tr>
</tbody>
</table>

## sam logs

Fetches logs that are generated by your Lambda function.

When your functions are a part of an AWS CloudFormation stack, you can fetch logs by using the function's logical ID when you specify the stack name.

**Usage:**

```
sam logs [OPTIONS]
```

**Examples:**

```
sam logs -n HelloWorldFunction --stack-name mystack

# Or, you can fetch logs using the function's name.
sam logs -n mystack-HelloWorldFunction-1FJ8PD36GML2Q

# You can view logs for a specific time range using the -s (--start-time) and -e (--end-time) options
sam logs -n HelloWorldFunction --stack-name mystack -s '10min ago' -e '2min ago'

# You can also add the --tail option to wait for new logs and see them as they arrive.
sam logs -n HelloWorldFunction --stack-name mystack --tail

# Use the --filter option to quickly find logs that match terms, phrases or values in your log events.
```
### Options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-n, --name TEXT</code></td>
<td>The name of your Lambda function. If this function is part of an AWS CloudFormation stack, this can be the logical ID of the function resource in the AWS CloudFormation/AWS SAM template. [required]</td>
</tr>
<tr>
<td><code>--stack-name TEXT</code></td>
<td>The name of the AWS CloudFormation stack that the function is a part of.</td>
</tr>
<tr>
<td><code>--filter TEXT</code></td>
<td>Lets you specify an expression to quickly find logs that match terms, phrases, or values in your log events. This can be a simple keyword (for example, &quot;error&quot;) or a pattern that's supported by Amazon CloudWatch Logs. For the syntax, see the Amazon CloudWatch Logs documentation.</td>
</tr>
<tr>
<td><code>-s, --start-time TEXT</code></td>
<td>Fetches logs starting at this time. The time can be relative values like '5mins ago', 'yesterday', or a formatted timestamp like '2018-01-01 10:10:10'. It defaults to '10mins ago'.</td>
</tr>
<tr>
<td><code>-e, --end-time TEXT</code></td>
<td>Fetches logs up to this time. The time can be relative values like '5mins ago', 'tomorrow', or a formatted timestamp like '2018-01-01 10:10:10'.</td>
</tr>
<tr>
<td><code>-t, --tail</code></td>
<td>Tails the log output. This ignores the end time argument and continues to fetch logs as they become available.</td>
</tr>
<tr>
<td><code>--profile TEXT</code></td>
<td>The specific profile from your credential file that gets AWS credentials.</td>
</tr>
<tr>
<td><code>--region TEXT</code></td>
<td>The AWS Region to deploy to. For example, us-east-1.</td>
</tr>
<tr>
<td><code>--config-file PATH</code></td>
<td>The path and file name of the configuration file containing default parameter values to use. The default value is &quot;samconfig.toml&quot; in the root of the project directory. For more information about configuration files, see AWS SAM CLI configuration file (p. 259).</td>
</tr>
<tr>
<td><code>--config-env TEXT</code></td>
<td>The environment name specifying the default parameter values in the configuration file to use. The default value is &quot;default&quot;. For more information about configuration files, see AWS SAM CLI configuration file (p. 259).</td>
</tr>
<tr>
<td><code>--debug</code></td>
<td>Turns on debug logging to print debug message generated by the AWS SAM CLI and display timestamps.</td>
</tr>
<tr>
<td><code>--help</code></td>
<td>Shows this message and exits.</td>
</tr>
</tbody>
</table>

### sam package

Packages an AWS SAM application. This command creates a .zip file of your code and dependencies, and uploads the file to Amazon Simple Storage Service (Amazon S3). It then returns a copy of your AWS SAM template, replacing references to local artifacts with the Amazon S3 location where the command uploaded the artifacts.

**Note**

`sam deploy (p. 241)` now implicitly performs the functionality of `sam package`. You can use the `sam deploy (p. 241)` command directly to package and deploy your application.
# Usage:

```bash
sam package [OPTIONS] [ARGS]...
```

## Options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-t, --template-file</code>, <code>--template PATH</code></td>
<td>The path and file name where your AWS SAM template is located. Default: `template.[yaml</td>
</tr>
<tr>
<td><code>--s3-bucket TEXT</code></td>
<td>(Required) The name of the Amazon S3 bucket where this command uploads your AWS CloudFormation template.</td>
</tr>
<tr>
<td><code>--s3-prefix TEXT</code></td>
<td>Prefix added to the artifacts name that are uploaded to the Amazon S3 bucket. The prefix name is a path name (folder name) for the Amazon S3 bucket. Only applies for functions declared with <code>Zip</code> package type.</td>
</tr>
<tr>
<td><code>--image-repository TEXT</code></td>
<td>The URI of the Amazon Elastic Container Registry (Amazon ECR) repository where this command uploads your function's image. Required for functions declared with the <code>Image</code> package type.</td>
</tr>
<tr>
<td><code>--kms-key-id TEXT</code></td>
<td>The ID of an AWS Key Management Service (AWS KMS) key used to encrypt artifacts that are at rest in the Amazon S3 bucket.</td>
</tr>
<tr>
<td><code>--signing-profiles LIST</code></td>
<td>(Optional) The list of signing profiles to sign your deployment packages with. This parameter takes a list of key-value pairs, where the key is the name of the function or layer to sign, and the value is the signing profile, with an optional profile owner delimited with <code>:</code>. For example, <code>FunctionNameToSign=SigningProfileName1:SigningProfileOwner,LayerNameToSign=SigningProfileName2:SigningProfileOwner</code>.</td>
</tr>
<tr>
<td><code>--output-template-file PATH</code></td>
<td>The path to the file where the command writes the packaged template. If you don't specify a path, the command writes the template to the standard output.</td>
</tr>
<tr>
<td><code>--use-json</code></td>
<td>Output JSON for the AWS CloudFormation template. YAML is used by default.</td>
</tr>
<tr>
<td><code>--force-upload</code></td>
<td>Override existing files in the Amazon S3 bucket. Specify this flag to upload artifacts even if they match existing artifacts in the Amazon S3 bucket.</td>
</tr>
<tr>
<td><code>--metadata</code></td>
<td>(Optional) A map of metadata to attach to all artifacts that are referenced in your template.</td>
</tr>
<tr>
<td><code>--profile TEXT</code></td>
<td>The specific profile from your credential file that gets AWS credentials.</td>
</tr>
<tr>
<td><code>--region TEXT</code></td>
<td>The AWS Region to deploy to. For example, us-east-1.</td>
</tr>
<tr>
<td><code>--config-file PATH</code></td>
<td>The path and file name of the configuration file containing default parameter values to use. The default value is &quot;samconfig.toml&quot; in the root of the project directory. For more information about configuration files, see AWS SAM CLI configuration file (p. 259).</td>
</tr>
<tr>
<td><code>--config-env TEXT</code></td>
<td>The environment name specifying the default parameter values in the configuration file to use. The default value is &quot;default&quot;. For more information about configuration files, see AWS SAM CLI configuration file (p. 259).</td>
</tr>
</tbody>
</table>
## Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--no-progressbar</td>
<td>Do not display a progress bar when uploading artifacts to Amazon S3.</td>
</tr>
<tr>
<td>--debug</td>
<td>Turns on debug logging to print debug message generated by the AWS SAM CLI and display timestamps.</td>
</tr>
<tr>
<td>--help</td>
<td>Shows this message and exits.</td>
</tr>
</tbody>
</table>

### Note

If the AWS SAM template contains a Metadata section for ServerlessRepo, and the LicenseUrl or ReadmeUrl properties contain references to local files, you must update AWS CLI to version 1.16.77 or later. For more information about the Metadata section of AWS SAM templates and publishing applications with AWS SAM CLI, see Publishing serverless applications using the AWS SAM CLI (p. 225).

### sam publish

Publish an AWS SAM application to the AWS Serverless Application Repository. Takes a packaged AWS SAM template and publishes the application to the specified AWS Region.

The `sam publish` command expects the AWS SAM template to include a Metadata section that contains application metadata required for publishing. In the Metadata section, the LicenseUrl and ReadmeUrl properties must refer to Amazon Simple Storage Service (Amazon S3) buckets, not local files. For more information about the Metadata section of the AWS SAM template, see Publishing serverless applications using the AWS SAM CLI (p. 225).

By default, `sam publish` creates the application as private. Before other AWS accounts are allowed to view and deploy your application, you must share it. For information on sharing applications, see AWS Serverless Application Repository Resource-Based Policy Examples in the AWS Serverless Application Repository Developer Guide.

### Note

Currently `sam publish` doesn't support publishing nested applications that are specified locally. If your application contains nested applications, you must publish them separately to the AWS Serverless Application Repository before publishing your parent application.

### Usage:

```
sam publish [OPTIONS]
```

### Examples:

```
# To publish an application
sam publish --template packaged.yaml --region us-east-1
```

### Options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-t, --template PATH</td>
<td>The path of AWS SAM template file [default: template.[yaml</td>
</tr>
<tr>
<td>--semantic-version TEXT</td>
<td>(Optional) Use this option to provide a semantic version of your application that overrides the SemanticVersion property in the</td>
</tr>
</tbody>
</table>
Option      | Description                                                                                                                                 |
-------------|---------------------------------------------------------------------------------------------------------------------------------------------|
Metadata     | Metadata section of the template file. For more information about semantic versioning, see the Semantic Versioning specification.          |
--profile TEXT | The specific profile from your credential file that gets AWS credentials.                                                                  |
--region TEXT   | The AWS Region to deploy to. For example, us-east-1.                                                                                          |
--config-file PATH | The path and file name of the configuration file containing default parameter values to use. The default value is "samconfig.toml" in the root of the project directory. For more information about configuration files, see AWS SAM CLI configuration file (p. 259). |
--config-env TEXT | The environment name specifying the default parameter values in the configuration file to use. The default value is "default". For more information about configuration files, see AWS SAM CLI configuration file (p. 259). |
--debug     | Turns on debug logging to print debug messages that the AWS SAM CLI generates, and to display timestamps.                                 |
--help       | Shows this message and exits.                                                                                                                                 |

**sam validate**

Verifies whether an AWS SAM template file is valid.

**Usage:**

```
sam validate [OPTIONS]
```

**Options:**

Option      | Description                                                                                                                                 |
-------------|---------------------------------------------------------------------------------------------------------------------------------------------|
-t, --template, --template-file PATH | The AWS SAM template file [default: template.[yaml|yml]].                                                                                     |
--profile TEXT   | The specific profile from your credential file that gets AWS credentials.                                                                  |
--region TEXT   | The AWS Region to deploy to. For example, us-east-1.                                                                                          |
--config-file PATH | The path and file name of the configuration file containing default parameter values to use. The default value is "samconfig.toml" in the root of the project directory. For more information about configuration files, see AWS SAM CLI configuration file (p. 259). |
--config-env TEXT | The environment name specifying the default parameter values in the configuration file to use. The default value is "default". For more information about configuration files, see AWS SAM CLI configuration file (p. 259). |
--debug     | Turns on debug logging to print debug message generated by the AWS SAM CLI and display timestamps.                                 |
AWS SAM CLI configuration file

The AWS SAM CLI supports a project-level configuration file that stores default parameters for its commands. This configuration file is in the TOML file format. The default file name is `samconfig.toml`, and the file's default location is your project's root directory, which contains your project's AWS SAM template file.

You can manually edit this file to set default parameters for any AWS SAM CLI command. In addition, the `sam deploy --guided` command writes a subset of parameters to your configuration file. For more information about this command, see Writing configurations with `sam deploy --guided` (p. 260) later in this topic.

Example

Here's an example configuration file that contains parameters for the `deploy` and `build` commands for the default environment:

```toml
version=0.1
[default.deploy.parameters]
stack_name = "my-app-stack"
s3_bucket = "my-source-bucket"
s3_prefix = "my-s3-prefix"
image_repositories = ["my-function-1=image-repo-1", "my-function-2=image-repo-2"]
region = "us-west-2"
confirm_changeset = true
capabilities = "CAPABILITY_IAM"
tags = "project="my-application" stage="production"

[default.build.parameters]
container_env_var = ["Function1.GITHUB_TOKEN=TOKEN1", "Function2.GITHUB_TOKEN=TOKEN2"]
container_env_var_file = "env.json"
```

Configuration file rules

The AWS SAM CLI applies the following rules to configuration files:

File name and location

- The default configuration file is named `samconfig.toml` and is located in your project's root directory.
- You can override the default file name and location using the `--config-file` parameter.

Tables

- The AWS SAM CLI uses TOML tables to group configuration entries by environment and command.
- For commands, the format of the table header is `[environment.command.parameters]`. For example, for the `sam deploy` command, the configuration table header is `[default.deploy.parameters]`.
- For subcommands, the format of the table header is `[environment.command_subcommand.parameters]`. That is, delimit the command and subcommand with `_` (underscore). For example, for the `sam local invoke` command, the configuration table header is `[default.local_invoke.parameters]`.

259
• If any command or subcommand contains a - (hyphen) character, replace it with _ (underscore). For example, for the sam local start-api command, the configuration table header is [default.local_start_api.parameters].
• The default environment name is default. You can override the default environment name using the --config-env parameter.
• A single configuration file can contain tables for multiple environments and multiple commands and subcommands.

Configuration entries

• Each configuration entry is a TOML key-value pair.
• The configuration key is the long-form parameter name with the - (hyphen) character replaced with _ (underscore). For the list of available parameters for each command, see the AWS SAM CLI command reference (p. 237), or run sam command --help.
• The configuration value can take the following forms:
  • For toggle parameters, the value can be true or false (no quotation marks). For example, confirm_changeset = true.
  • For parameters that take a single argument, the value is the argument surrounded by " " (quotation marks). For example, region = "us-west-2".
  • For parameters that take a list of arguments, the arguments are space-delimited within " " (quotation marks). For example, capabilities = "CAPABILITY_IAM CAPABILITY_NAMED_IAM".
  • To specify a list of key-value pairs, the pairs are space-delimited, and the value of each pair is surrounded by encoded " " (quotation marks). For example, tags = "project=\"my-application\" stage=\"production\"".
  • For parameters that you can specify multiple times, the value is an array of arguments. For example, image_repositories = ["my-function-1=image-repo-1", "my-function-2=image-repo-2"].

Precedence

• Parameter values that you provide on the command line take precedence over corresponding entries in the configuration file. For example, if your configuration file contains the entry stack_name = "DefaultStack" and you run the command sam deploy --stack-name MyCustomStack, then the deployed stack name is MyCustomStack.
• For the parameter_overrides entry, both the parameter values that you provide on the command line and entries in the configuration file take precedence over corresponding objects declared in the Parameters section of the template file.

Writing configurations with sam deploy --guided

When you run the sam deploy --guided command, the AWS SAM CLI guides you through the deployment with a series of prompts.

These prompts include the question "Save arguments to samconfig.toml [Y/n]:". If you respond y to this prompt, the AWS SAM CLI updates the configuration file with values for the following parameters:
• stack_name
• s3_bucket
• s3_prefix
• image_repository
• region
• confirm_changeset
• capabilities
• signing_profiles
• parameter_overrides

If you've previously written a configuration file, the AWS SAM CLI reads it and uses those parameter values as the default values for the corresponding prompts. If you specify values on the command line for any of these parameters, the AWS SAM CLI uses those values as the default values.

To set any parameter other than those previously listed, you must manually edit the configuration file.

AWS SAM policy templates

AWS SAM allows you to choose from a list of policy templates to scope the permissions of your Lambda functions to the resources that are used by your application.

AWS SAM applications in the AWS Serverless Application Repository that use policy templates don't require any special customer acknowledgments to deploy the application from the AWS Serverless Application Repository.

If you want to request a new policy template to be added, do the following:

1. Submit a pull request against the policy_templates.json source file in the develop branch of the AWS SAM GitHub project. You can find the source file in policy_templates.json on the GitHub website.
2. Submit an issue in the AWS SAM GitHub project that includes the reasons for your pull request and a link to the request. Use this link to submit a new issue: AWS Serverless Application Model: Issues.

Syntax

For every policy template you specify in your AWS SAM template file, you must always specify an object containing the policy template's placeholder values. If a policy template does not require any placeholder values, you must specify an empty object.

YAML

```
MyFunction:
  Type: AWS::Serverless::Function
  Properties:
    Policies:
    - PolicyTemplateName1:        # Policy template with placeholder value
      Key1: Value1
    - PolicyTemplateName2: {}     # Policy template with no placeholder value
```

Examples

Example 1: Policy template with placeholder values

The following example shows that the SQSPollerPolicy (p. 267) policy template expects a QueueName as a resource. The AWS SAM template retrieves the name of the "MyQueue" Amazon SQS queue, which you can create in the same application or requested as a parameter to the application.
Example 2: Policy template with no placeholder values

The following example contains the CloudWatchPutMetricPolicy (p. 268) policy template, which has no placeholder values.

**Note**

Even though there are no placeholder values, you must specify an empty object, otherwise an error will result.

<table>
<thead>
<tr>
<th>Policy Template</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQSPollerPolicy</td>
<td>Gives permission to poll an Amazon Simple Queue Service (Amazon SQS) queue.</td>
</tr>
<tr>
<td>LambdaInvokePolicy</td>
<td>Gives permission to invoke an AWS Lambda function, alias, or version.</td>
</tr>
<tr>
<td>CloudWatchDescribeAlarmHistory</td>
<td>Gives permission to describe CloudWatch alarm history.</td>
</tr>
<tr>
<td>CloudWatchPutMetricPolicy</td>
<td>Gives permission to send metrics to CloudWatch.</td>
</tr>
<tr>
<td>EC2DescribePolicy</td>
<td>Gives permission to describe Amazon Elastic Compute Cloud (Amazon EC2) instances.</td>
</tr>
<tr>
<td>DynamoDBCrudPolicy</td>
<td>Gives create, read, update, and delete permissions to an Amazon DynamoDB table.</td>
</tr>
<tr>
<td>DynamoDBReadPolicy</td>
<td>Gives read-only permission to a DynamoDB table.</td>
</tr>
<tr>
<td>Policy Template</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DynamoDBWritePolicy</td>
<td>Gives write-only permission to a DynamoDB table.</td>
</tr>
<tr>
<td>DynamoDBReconfigurePolicy</td>
<td>Gives permission to reconfigure a DynamoDB table.</td>
</tr>
<tr>
<td>SESSendBouncePolicy</td>
<td>Gives SendBounce permission to an Amazon Simple Email Service (Amazon SES) identity.</td>
</tr>
<tr>
<td>ElasticsearchHttpPostPolicy</td>
<td>Gives POST permission to Amazon Elasticsearch Service.</td>
</tr>
<tr>
<td>S3ReadPolicy (p. 272)</td>
<td>Gives read-only permission to read objects in an Amazon Simple Storage Service (Amazon S3) bucket.</td>
</tr>
<tr>
<td>S3WritePolicy (p. 272)</td>
<td>Gives write permission to write objects into an Amazon S3 bucket.</td>
</tr>
<tr>
<td>S3CrudPolicy (p. 272)</td>
<td>Gives create, read, update, and delete permission to act on the objects in an Amazon S3 bucket.</td>
</tr>
<tr>
<td>AMIDescribePolicy</td>
<td>Gives permission to describe Amazon Machine Images (AMIs).</td>
</tr>
<tr>
<td>CloudFormationDescribeStacksPolicy</td>
<td>Gives permission to describe AWS CloudFormation stacks.</td>
</tr>
<tr>
<td>RekognitionDetectOnlyPolicy</td>
<td>Gives permission to detect faces, labels, and text.</td>
</tr>
<tr>
<td>RekognitionNoDataAccessPolicy</td>
<td>Gives permission to compare and detect faces and labels.</td>
</tr>
<tr>
<td>RekognitionReadPolicy</td>
<td>Gives permission to list and search faces.</td>
</tr>
<tr>
<td>RekognitionWriteOnlyAccessPolicy</td>
<td>Gives permission to create collection and index faces.</td>
</tr>
<tr>
<td>SQSSendMessagePolicy</td>
<td>Gives permission to send message to an Amazon SQS queue.</td>
</tr>
<tr>
<td>SNSPublishMessagePolicy</td>
<td>Gives permission to publish a message to an Amazon Simple Notification Service (Amazon SNS) topic.</td>
</tr>
<tr>
<td>VPCAccessPolicy (p. 277)</td>
<td>Gives access to create, delete, describe, and detach elastic network interfaces.</td>
</tr>
<tr>
<td>DynamoDBStreamReadPolicy</td>
<td>Gives permission to describe and read DynamoDB streams and records.</td>
</tr>
<tr>
<td>KinesisStreamReadPolicy</td>
<td>Gives permission to list and read an Amazon Kinesis stream.</td>
</tr>
<tr>
<td>SESCrudPolicy (p. 279)</td>
<td>Gives permission to send email and verify identity.</td>
</tr>
<tr>
<td>Policy Template</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SNSCrudPolicy (p. 279)</td>
<td>Gives permission to create, publish, and subscribe to Amazon SNS topics.</td>
</tr>
<tr>
<td>KinesisCrudPolicy (p. 280)</td>
<td>Gives permission to create, publish, and delete an Amazon Kinesis stream.</td>
</tr>
<tr>
<td>KMSDecryptPolicy</td>
<td>Gives permission to decrypt with an AWS Key Management Service (AWS KMS) key.</td>
</tr>
<tr>
<td>KMSEncryptPolicy</td>
<td>Gives permission to encrypt with an AWS Key Management Service (AWS KMS) key.</td>
</tr>
<tr>
<td>PollyFullAccessPolicy</td>
<td>Gives full access permission to Amazon Polly lexicon resources.</td>
</tr>
<tr>
<td>S3FullAccessPolicy</td>
<td>Gives full access permission to act on the objects in an Amazon S3 bucket.</td>
</tr>
<tr>
<td>CodePipelineLambdaExecutionPolicy</td>
<td>Gives permission for a Lambda function invoked by CodePipeline to report the status of the job.</td>
</tr>
<tr>
<td>ServerlessRepoReadWriteAccessPolicy</td>
<td>Gives permission to create and list applications in the AWS Serverless Application Repository service.</td>
</tr>
<tr>
<td>EC2CopyImagePolicy</td>
<td>Gives permission to copy Amazon EC2 images.</td>
</tr>
<tr>
<td>AWSSecretsManagerWritePolicy</td>
<td>Gives permission to write a secret in AWS Secrets Manager.</td>
</tr>
<tr>
<td>AWSSecretsManagerReadPolicy</td>
<td>Gives permission to get the secret value for the specified AWS Secrets Manager secret.</td>
</tr>
<tr>
<td>CodePipelineReadOnlyPolicy</td>
<td>Gives read permission to get details about a CodePipeline pipeline.</td>
</tr>
<tr>
<td>CloudWatchDashboardPolicy</td>
<td>Gives permissions to put metrics to operate on CloudWatch dashboards.</td>
</tr>
<tr>
<td>RekognitionFacesManagementPolicy</td>
<td>Gives permission to add, delete, and search faces in an Amazon Rekognition collection.</td>
</tr>
<tr>
<td>RekognitionFacesPolicy</td>
<td>Gives permission to compare and detect faces and labels.</td>
</tr>
<tr>
<td>RekognitionLabelsPolicy</td>
<td>Gives permission to detect object and moderation labels.</td>
</tr>
<tr>
<td>DynamoDBBackupFullAccessPolicy</td>
<td>Gives read and write permission to DynamoDB on-demand backups for a table.</td>
</tr>
<tr>
<td>DynamoDBRestorePolicy</td>
<td>Gives permission to restore a DynamoDB table from backup.</td>
</tr>
<tr>
<td>ComprehendBasicAccessPolicy</td>
<td>Gives permission for detecting entities, key phrases, languages, and sentiments.</td>
</tr>
<tr>
<td>Policy Template</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MobileAnalyticsWriteOnlyAccessPolicy</td>
<td>Gives write-only permission to put event data for all application resources.</td>
</tr>
<tr>
<td>PinpointEndpointAccessPolicy</td>
<td>Gives permission to get and update endpoints for an Amazon Pinpoint application.</td>
</tr>
<tr>
<td>FirehoseWritePolicy</td>
<td>Gives permission to write to a Kinesis Data Firehose delivery stream.</td>
</tr>
<tr>
<td>FirehoseCrudPolicy</td>
<td>Gives permission to create, write, update, and delete a Kinesis Data Firehose delivery stream.</td>
</tr>
<tr>
<td>EKSDescribePolicy</td>
<td>Gives permission to describe or list Amazon EKS clusters.</td>
</tr>
<tr>
<td>CostExplorerReadOnlyPolicy</td>
<td>Gives read-only permission to the read-only Cost Explorer APIs for billing history.</td>
</tr>
<tr>
<td>OrganizationsListAccountsPolicy</td>
<td>Gives read-only permission to list child account names and IDs.</td>
</tr>
<tr>
<td>SESBulkTemplatedCrudPolicy</td>
<td>Gives permission to send email, templated email, templated bulk emails and verify identity.</td>
</tr>
<tr>
<td>SESEmailTemplateCrudPolicy</td>
<td>Gives permission to create, get, list, update and delete Amazon SES email templates.</td>
</tr>
<tr>
<td>FilterLogEventsPolicy</td>
<td>Gives permission to filter CloudWatch Logs events from a specified log group.</td>
</tr>
<tr>
<td>SSMParameterReadPolicy</td>
<td>Gives permission to access a parameters from an Amazon EC2 Systems Manager (SSM) parameter store to load secrets in this account.</td>
</tr>
<tr>
<td>StepFunctionsExecutionPolicy</td>
<td>Gives permission to start a Step Functions state machine execution.</td>
</tr>
<tr>
<td>CodeCommitCrudPolicy</td>
<td>Gives permissions to create/read/update/delete objects within a specific CodeCommit repository.</td>
</tr>
<tr>
<td>CodeCommitReadPolicy</td>
<td>Gives permissions to read objects within a specific CodeCommit repository.</td>
</tr>
<tr>
<td>AthenaQueryPolicy</td>
<td>Gives permission to execute Athena queries.</td>
</tr>
<tr>
<td>TextractPolicy (p. 265)</td>
<td>Gives full access to Amazon Textract.</td>
</tr>
<tr>
<td>TextractDetectAnalyzePolicy</td>
<td>Gives access to detect and analyze documents with Amazon Textract.</td>
</tr>
<tr>
<td>TextractGetResultPolicy</td>
<td>Gives access to get detected and analyzed documents from Amazon Textract.</td>
</tr>
</tbody>
</table>
Troubleshooting

**SAM CLI error: "Must specify valid parameter values for policy template '<policy-template-name>'"**

When executing `sam build`, you see the following error:

"Must specify valid parameter values for policy template '<policy-template-name>'"

This means that you did not pass an empty object when declaring a policy template that does not have any placeholder values.

To fix this, declare the policy like the following example for `CloudWatchPutMetricPolicy` (p. 268).

```
MyFunction:
  Policies:
    - CloudWatchPutMetricPolicy: {}
```
Policy template list

The following are the available policy templates, along with the permissions that are applied to each one. AWS Serverless Application Model (AWS SAM) automatically populates the placeholder items (such as AWS Region and account ID) with the appropriate information.

SQSPollerPolicy

Gives permission to poll an Amazon Simple Queue Service (Amazon SQS) queue.

```
"Statement": [  
  {  
    "Effect": "Allow",  
    "Action": [  
      "sqs:ChangeMessageVisibility",  
      "sqs:ChangeMessageVisibilityBatch",  
      "sqs:DeleteMessage",  
      "sqs:DeleteMessageBatch",  
      "sqs:GetQueueAttributes",  
      "sqs:ReceiveMessage"  
    ],  
    "Resource": {  
      "Fn::Sub": [  
        "arn:${AWS::Partition}:sqs:${AWS::Region}:${AWS::AccountId}:${queueName}",  
        {  
          "queueName": {  
            "Ref": "QueueName"  
          }  
        }  
      ]  
    }  
  }  
]
```

LambdaInvokePolicy

Gives permission to invoke an AWS Lambda function, alias, or version.

```
"Statement": [  
  {  
    "Effect": "Allow",  
    "Action": [  
      "lambda:InvokeFunction"  
    ],  
    "Resource": {  
      "Fn::Sub": [  
        "arn:${AWS::Partition}:lambda:${AWS::Region}:${AWS::AccountId}:function:  
        ${functionName}**",  
        {  
          "functionName": {  
            "Ref": "FunctionName"  
          }  
        }  
      ]  
    }  
  }  
]
```
**CloudWatchDescribeAlarmHistoryPolicy**

Gives permission to describe Amazon CloudWatch alarm history.

```json
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "cloudwatch:DescribeAlarmHistory"
    ],
    "Resource": "*"
  }
]
```

**CloudWatchPutMetricPolicy**

Gives permission to send metrics to CloudWatch.

```json
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "cloudwatch:PutMetricData"
    ],
    "Resource": "*"
  }
]
```

**EC2DescribePolicy**

Gives permission to describe Amazon Elastic Compute Cloud (Amazon EC2) instances.

```json
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ec2:DescribeRegions",
      "ec2:DescribeInstances"
    ],
    "Resource": "*"
  }
]
```

**DynamoDBCrudPolicy**

Gives create, read, update, and delete permissions to an Amazon DynamoDB table.

```json
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
```

268
"dynamodb:GetItem",
"dynamodb:DeleteItem",
"dynamodb:PutItem",
"dynamodb:Scan",
"dynamodb:Query",
"dynamodb:UpdateItem",
"dynamodb:BatchWriteItem",
"dynamodb:BatchGetItem",
"dynamodb:DescribeTable",
"dynamodb:ConditionCheckItem"
],
"Resource": [
{
"Fn::Sub": [
  "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
  ${tableName}",
  {
    "tableName": {
      "Ref": "TableName"
    }
  }
],
"Fn::Sub": [
  "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
  ${tableName}/index/*",
  {
    "tableName": {
      "Ref": "TableName"
    }
  }
],
"Fn::Sub": [
  "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
  ${tableName}",
  {
    "tableName": {
      "Ref": "TableName"
    }
  }
]}
]

DynamoDBReadPolicy
Gives read-only permission to a DynamoDB table.

"Statement": [
{
  "Effect": "Allow",
  "Action": [
    "dynamodb:GetItem",
    "dynamodb:Scan",
    "dynamodb:Query",
    "dynamodb:BatchGetItem",
    "dynamodb:DescribeTable"
  ],
  "Resource": [
    "Fn::Sub": [
      "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
      ${tableName}",
      {
        "tableName": {
          "Ref": "TableName"
        }
      }
    ]
  ]
]
DynamoDBWritePolicy

Gives write-only permission to a DynamoDB table.

DynamoDBReconfigurePolicy

Gives permission to reconfigure a DynamoDB table.
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "dynamodb:UpdateTable"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
        ${tableName}",
        {
          "tableName": {
            "Ref": "TableName"
          }
        }
      ]
    }
  }
]

SESSendBouncePolicy

Gives SendBounce permission to an Amazon Simple Email Service (Amazon SES) identity.

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ses:SendBounce"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:ses:${AWS::Region}:${AWS::AccountId}:identity/
        ${identityName}",
        {
          "identityName": {
            "Ref": "IdentityName"
          }
        }
      ]
    }
  }
]

ElasticsearchHttpPostPolicy

Gives POST and PUT permission to Amazon Elasticsearch Service.

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "es:ESHttpPost",
      "es:ESHttpPut"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:es:${AWS::Region}:${AWS::AccountId}:domain/
        ${domainName}",
        "TableName": {
          "Ref": "TableName"
        }
      ]
    }
  }
]
"Fn::Sub": [
  "arn:${AWS::Partition}:es:${AWS::Region}:${AWS::AccountId}:domain/
${domainName}/**",
  {
    "domainName": {
      "Ref": "DomainName"
    }
  }
]

S3ReadPolicy
Gives read-only permission to read objects in an Amazon Simple Storage Service (Amazon S3) bucket.

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
      "s3:ListBucket",
      "s3:GetBucketLocation",
      "s3:GetObjectVersion",
      "s3:GetLifecycleConfiguration"
    ],
    "Resource": [
      {
        "Fn::Sub": [
          "arn:${AWS::Partition}:s3:::${bucketName}/*",
          {
            "bucketName": {
              "Ref": "BucketName"
            }
          }
        ],
        {
          "Fn::Sub": [
            "arn:${AWS::Partition}:s3:::${bucketName}/*",
            {
              "bucketName": {
                "Ref": "BucketName"
              }
            }
          ]
        }
      ]
    ]
  }
]

S3WritePolicy
Gives write permission to write objects into an Amazon S3 bucket.

"Statement": [
  {

"Effect": "Allow",
"Action": [
  "s3:PutObject",
  "s3:PutObjectAcl",
  "s3:PutLifecycleConfiguration"
],
"Resource": [
  {
    "Fn::Sub": [
      "arn:${AWS::Partition}:s3:::${bucketName}",
      {
        "bucketName": {
          "Ref": "BucketName"
        }
      }
    ],
    "Fn::Sub": [
      "arn:${AWS::Partition}:s3:::${bucketName}/*",
      {
        "bucketName": {
          "Ref": "BucketName"
        }
      }
    ]
  }
]

S3CrudPolicy

Gives create, read, update, and delete permission to act on the objects in an Amazon S3 bucket.

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
      "s3:ListBucket",
      "s3:GetBucketLocation",
      "s3:GetObjectVersion",
      "s3:PutObject",
      "s3:PutObjectAcl",
      "s3:GetLifecycleConfiguration",
      "s3:PutLifecycleConfiguration",
      "s3:DeleteObject"
    ],
    "Resource": [
      {
        "Fn::Sub": [
          "arn:${AWS::Partition}:s3:::${bucketName}",
          {
            "bucketName": {
              "Ref": "BucketName"
            }
          }
        ],
        "Fn::Sub": [
          "arn:${AWS::Partition}:s3:::${bucketName}/*",
          {
            "bucketName": {
              "Ref": "BucketName"
            }
          }
        ]
      }
    ]
  }
]
Policy template list

```
"arn:${AWS::Partition}:s3:${bucketName}/*",
{
  "bucketName": {
    "Ref": "BucketName"
  }
}
```

**AMIDescribePolicy**

Gives permission to describe Amazon Machine Images (AMIs).

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ec2:DescribeImages"
    ],
    "Resource": {
      "Fn::Sub": "arn:${AWS::Partition}:ec2:${AWS::Region}:${AWS::AccountId}:image/*"
    }
  }
]
```

**CloudFormationDescribeStacksPolicy**

Gives permission to describe AWS CloudFormation stacks.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "cloudformation:DescribeStacks"
    ],
    "Resource": {
      "Fn::Sub": "arn:${AWS::Partition}:cloudformation:${AWS::Region}:${AWS::AccountId}:stack/*"
    }
  }
]
```

**RekognitionDetectOnlyPolicy**

Gives permission to detect faces, labels, and text.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
```
"rekognition:DetectFaces",
"rekognition:DetectLabels",
"rekognition:DetectModerationLabels",
"rekognition:DetectText"
],
"Resource": "*"
}
]

RekognitionNoDataAccessPolicy

Gives permission to compare and detect faces and labels.

"Statement": [
{
  "Effect": "Allow",
  "Action": [
  "rekognition:CompareFaces",
  "rekognition:DetectFaces",
  "rekognition:DetectLabels",
  "rekognition:DetectModerationLabels"
  ],
  "Resource": { "Fn::Sub": [
  "arn:${AWS::Partition}:rekognition:${AWS::Region}:
  ${AWS::AccountId}:collection/${collectionId}",
  { "collectionId": { "Ref": "CollectionId" } }
  ]
  }
}
]

RekognitionReadPolicy

Gives permission to list and search faces.

"Statement": [
{
  "Effect": "Allow",
  "Action": [
  "rekognition:ListCollections",
  "rekognition:ListFaces",
  "rekognition:SearchFaces",
  "rekognition:SearchFacesByImage"
  ],
  "Resource": { "Fn::Sub": [
  "arn:${AWS::Partition}:rekognition:${AWS::Region}:
  ${AWS::AccountId}:collection/${collectionId}",
  { "collectionId": { "Ref": "CollectionId" } }
  ]
  }
}
RekognitionWriteOnlyAccessPolicy

Gives permission to create collection and index faces.

```
"Statement": [  
  {   
    "Effect": "Allow",  
    "Action": [   
      "rekognition:CreateCollection",  
      "rekognition:IndexFaces"  
    ],  
    "Resource": {   
      "Fn::Sub": [  
        "arn:${AWS::Partition}:rekognition:${AWS::Region}:#{AWS::AccountId}:collection/${collectionId}"  
      ],  
      "collectionId": {  
        "Ref": "CollectionId"  
      }  
    }  
  }  
]
```

SQSSendMessagePolicy

Gives permission to send message to an Amazon SQS queue.

```
"Statement": [  
  {   
    "Effect": "Allow",  
    "Action": [   
      "sqs:SendMessage*"  
    ],  
    "Resource": {   
      "Fn::Sub": [  
        "arn:${AWS::Partition}:sqs:${AWS::Region}:${AWS::AccountId}:${queueName}"  
      ],  
      "queueName": {  
        "Ref": "QueueName"  
      }  
    }  
  }  
]
```

SNSPublishMessagePolicy

Gives permission to publish a message to an Amazon Simple Notification Service (Amazon SNS) topic.

```
"Statement": [  
  {   
    "Effect": "Allow",  
    "Action": [   
      "sqs:SendMessage*"  
    ],  
    "Resource": {   
      "Fn::Sub": [  
        "arn:${AWS::Partition}:sqs:${AWS::Region}:${AWS::AccountId}:${queueName}"  
      ],  
      "queueName": {  
        "Ref": "QueueName"  
      }  
    }  
  }  
]
```
"Statement": [  
  {  
    "Effect": "Allow",  
    "Action": [  
      "sns:Publish"  
    ],  
    "Resource": {  
      "Fn::Sub": [  
        "arn:${AWS::Partition}:sns:${AWS::Region}:${AWS::AccountId}:${topicName}",  
        {  
          "topicName": {  
            "Ref": "TopicName"  
          }  
        }  
      ]  
    }  
  }  
],

VPCAccessPolicy

Gives access to create, delete, describe, and detach elastic network interfaces.

"Statement": [  
  {  
    "Effect": "Allow",  
    "Action": [  
      "ec2:CreateNetworkInterface",  
      "ec2:DeleteNetworkInterface",  
      "ec2:DescribeNetworkInterfaces",  
      "ec2:DetachNetworkInterface"  
    ],  
    "Resource": "*"  
  }  
]

DynamoDBStreamReadPolicy

Gives permission to describe and read DynamoDB streams and records.

"Statement": [  
  {  
    "Effect": "Allow",  
    "Action": [  
      "dynamodb:DescribeStream",  
      "dynamodb:GetRecords",  
      "dynamodb:GetShardIterator"  
    ],  
    "Resource": {  
      "Fn::Sub": [  
        "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/${tableName}/stream/${streamName}",  
        {  
          "tableName": {  
            "Ref": "TableName"  
          }  
        }  
      ]  
    }  
  }  
]
"streamName": {  
  "Ref": "StreamName"  
}
"Action": [  
  "dynamodb:ListStreams"  
],
"Resource": {  
  "Fn::Sub": [  
    "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
    *{tableName}/stream/*",  
    {  
      "tableName": {  
        "Ref": "TableName"  
      }  
    }  
  ]
}
}
},
{  
  "Effect": "Allow",
  "Action": [  
    "kinesis:ListStreams",
    "kinesis:DescribeLimits"  
],
  "Resource": {  
    "Fn::Sub": "arn:${AWS::Partition}:kinesis:${AWS::Region}:
    *{AWS::AccountId}:stream/*"  
  }
},
{  
  "Effect": "Allow",
  "Action": [  
    "kinesis:DescribeStream",
    "kinesis:DescribeStreamSummary",
    "kinesis:GetRecords",
    "kinesis:GetShardIterator"  
],
  "Resource": {  
    "Fn::Sub": [  
      "arn:${AWS::Partition}:kinesis:${AWS::Region}:${AWS::AccountId}:stream/
      *{streamName}"
    ]
  }
}

KinesisStreamReadPolicy

Gives permission to list and read an Amazon Kinesis stream.
SESCrudPolicy

Gives permission to send email and verify identity.

"Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ses:GetIdentityVerificationAttributes",
        "ses:SendEmail",
        "ses:SendRawEmail",
        "ses:VerifyEmailIdentity"
      ],
      "Resource": {
        "Fn::Sub": [
          "arn:${AWS::Partition}:ses:${AWS::Region}:${AWS::AccountId}:identity/
          ${identityName}",
          {
            "identityName": {
              "Ref": "IdentityName"
            }
          }
        ]
      }
    }
  ]

SNSCrudPolicy

Gives permission to create, publish, and subscribe to Amazon SNS topics.

"Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sns:ListSubscriptionsByTopic",
        "sns:CreateTopic",
        "sns:SetTopicAttributes",
        "sns:Subscribe",
        "sns:Publish"
      ],
      "Resource": {
        "Fn::Sub": [
          "arn:${AWS::Partition}:sns:${AWS::Region}:${AWS::AccountId}:${topicName}*",
          {
            "topicName": {
              "Ref": "TopicName"
            }
          }
        ]
      }
    }
  ]
**KinesisCrudPolicy**

Gives permission to create, publish, and delete an Amazon Kinesis stream.

```json
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "kinesis:AddTagsToStream",
      "kinesis:CreateStream",
      "kinesis:DecreaseStreamRetentionPeriod",
      "kinesis:DeleteStream",
      "kinesis:DescribeStream",
      "kinesis:DescribeStreamSummary",
      "kinesis:GetShardIterator",
      "kinesis:IncreaseStreamRetentionPeriod",
      "kinesis:ListTagsForStream",
      "kinesis:MergeShards",
      "kinesis:PutRecord",
      "kinesis:PutRecords",
      "kinesis:SplitShard",
      "kinesis:RemoveTagsFromStream"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:kinesis:${AWS::Region}:${AWS::AccountId}:stream/
        ${streamName}",
        {
          "streamName": {
            "Ref": "StreamName"
          }
        }
      ]
    }
  }
]
```

**KMSDecryptPolicy**

Gives permission to decrypt with an AWS Key Management Service (AWS KMS) key. Note that keyId must be an AWS KMS key ID, and not a key alias.

```json
"Statement": [
  {
    "Action": "kms:Decrypt",
    "Effect": "Allow",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:kms:${AWS::Region}:${AWS::AccountId}:key/${keyId}",
        {
          "keyId": {
            "Ref": "KeyId"
          }
        }
      ]
    }
  }
]
```
KMSEncryptPolicy

Gives permission to encrypt with an AWS KMS key. Note that keyid must be an AWS KMS key ID, and not a key alias.

```
"Statement": [
  {
    "Action": "kms:Encrypt",
    "Effect": "Allow",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:kms:${AWS::Region}:${AWS::AccountId}:key/${keyId}",
        {
          "keyId": {
            "Ref": "KeyId"
          }
        }
      ]
    }
  }
]
```

PollyFullAccessPolicy

Gives full access permission to Amazon Polly lexicon resources.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "polly:GetLexicon",
      "polly:DeleteLexicon"
    ],
    "Resource": [
      {
        "Fn::Sub": [
          "arn:${AWS::Partition}:polly:${AWS::Region}:${AWS::AccountId}:lexicon/
          ${lexiconName}",
          {
            "lexiconName": {
              "Ref": "LexiconName"
            }
          }
        ]
      }
    ],
    "Condition": {
      " ArnEquals": [
        " Arn:: Partition:: Region:: AccountId:: lexicon/
        *",
        " Arn:: Partition:: Region:: AccountId:: lexicon/*
      ]
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "polly:DescribeVoices",
      "polly:ListLexicons",
      "polly:PutLexicon",
      "polly:SynthesizeSpeech"
    ],
    "Resource": [
      {
        "Fn::Sub": "arn:${AWS::Partition}:polly:${AWS::Region}:
        ${AWS::AccountId}:lexicon/*"
      }
    ]
  }
]
```
S3FullAccessPolicy

Gives full access permission to act on the objects in an Amazon S3 bucket.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
      "s3:GetObjectAcl",
      "s3:GetObjectVersion",
      "s3:PutObject",
      "s3:PutObjectAcl",
      "s3:DeleteObject",
      "s3:DeleteObjectTagging",
      "s3:DeleteObjectVersionTagging",
      "s3:GetObjectTagging",
      "s3:GetObjectVersionTagging",
      "s3:GetObjectLocation",
      "s3:GetObjectTagging",
      "s3:GetObjectVersionTagging",
      "s3:GetObjectLocation",
      "s3:GetObjectVersionLocation"
    ],
    "Resource": [
      {
        "Fn::Sub": [
          "arn:${AWS::Partition}:s3:::${bucketName}/*",
          {
            "bucketName": {
              "Ref": "BucketName"
            }
          }
        ]
      }
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "s3:ListBucket",
      "s3:GetBucketLocation",
      "s3:GetLifecycleConfiguration",
      "s3:PutLifecycleConfiguration"
    ],
    "Resource": [
      {
        "Fn::Sub": [
          "arn:${AWS::Partition}:s3:::${bucketName}";
          {
            "bucketName": {
              "Ref": "BucketName"
            }
          }
        ]
      }
    ]
  }
]
```
**CodePipelineLambdaExecutionPolicy**

Gives permission for a Lambda function invoked by AWS CodePipeline to report the status of the job.

```
"Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codepipeline:PutJobSuccessResult",
        "codepipeline:PutJobFailureResult"
      ],
      "Resource": "*"
    }
  ]
```

**ServerlessRepoReadWriteAccessPolicy**

Gives permission to create and list applications in the AWS Serverless Application Repository (AWS SAM) service.

```
"Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "serverlessrepo:CreateApplication",
        "serverlessrepo:CreateApplicationVersion",
        "serverlessrepo:GetApplication",
        "serverlessrepo:ListApplications",
        "serverlessrepo:ListApplicationVersions"
      ],
      "Resource": [
        { "Fn::Sub": "arn:${AWS::Partition}:serverlessrepo:${AWS::Region}:${AWS::AccountId}:applications/*" }
      ]
    }
  ]
```

**EC2CopyImagePolicy**

Gives permission to copy Amazon EC2 images.

```
"Statement": [
    {
      "Effect": "Allow",
      "Action": ["ec2:CopyImage"],
      "Resource": {
        "Fn::Sub": ["arn:${AWS::Partition}:ec2:${AWS::Region}:${AWS::AccountId}:image/${imageId}"
      },
      "imageId": {
```
AWSSecretsManagerRotationPolicy

Gives permission to rotate a secret in AWS Secrets Manager.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "secretsmanager:DescribeSecret",
      "secretsmanager:GetSecretValue",
      "secretsmanager:PutSecretValue",
      "secretsmanager:UpdateSecretVersionStage"
    ],
    "Resource": {
      "Fn::Sub": "arn:${AWS::Partition}:secretsmanager:${AWS::Region}:#{AWS::AccountId}:secret:*"
    },
    "Condition": {
      "StringEquals": {
        "secretsmanager:resource/AllowRotationLambdaArn": {
          "Fn::Sub": {
            "arn:${AWS::Partition}:lambda:${AWS::Region}:#{AWS::AccountId}:function:${functionName}"
          },
        "functionName": {
          "Ref": "FunctionName"
        }
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "secretsmanager:GetRandomPassword"
    ],
    "Resource": "*"
  }
]
```

AWSSecretsManagerGetSecretValuePolicy

Gives permission to get the secret value for the specified AWS Secrets Manager secret.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "secretsmanager:GetSecretValue"
    ],
    "Resource": "*"
  }
]
```
"Action": [
  "secretsmanager:GetSecretValue"
],
"Resource": {
  "Fn::Sub": [
    "${secretArn}",
    {
      "secretArn": {
        "Ref": "SecretArn"
      }
    }
  ]
}
}

**CodePipelineReadOnlyPolicy**

Gives read permission to get details about a CodePipeline pipeline.

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "codepipeline:ListPipelineExecutions"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:codepipeline:${AWS::Region}:${AWS::AccountId}:${pipelinename}"
      ]
    }
  }
]

**CloudWatchDashboardPolicy**

Gives permissions to put metrics to operate on CloudWatch dashboards.

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "cloudwatch:GetDashboard",
      "cloudwatch:ListDashboards",
      "cloudwatch:PutDashboard",
      "cloudwatch:ListMetrics"
    ],
    "Resource": "*"
  }
]
RekognitionFacesManagementPolicy

Gives permission to add, delete, and search faces in an Amazon Rekognition collection.

```
"Statement": [{
  "Effect": "Allow",
  "Action": [
    "rekognition:IndexFaces",
    "rekognition:DeleteFaces",
    "rekognition:SearchFaces",
    "rekognition:SearchFacesByImage",
    "rekognition:ListFaces"
  ],
  "Resource": {
    "Fn::Sub": [
      "arn:${AWS::Partition}:rekognition:${AWS::Region}:
      ${AWS::AccountId}:collection/${collectionId}"
    ]
  }
}
```

RekognitionFacesPolicy

Gives permission to compare and detect faces and labels.

```
"Statement": [{
  "Effect": "Allow",
  "Action": [
    "rekognition:CompareFaces",
    "rekognition:DetectFaces"
  ],
  "Resource": "*"
}
```

RekognitionLabelsPolicy

Gives permission to detect object and moderation labels.

```
"Statement": [{
  "Effect": "Allow",
  "Action": [
    "rekognition:DetectLabels",
    "rekognition:DetectModerationLabels"
  ],
  "Resource": "*"
}
```
**DynamoDBBackupFullAccessPolicy**

Gives read and write permission to DynamoDB on-demand backups for a table.

```
"Statement": [{
  "Effect": "Allow",
  "Action": [
    "dynamodb:CreateBackup",
    "dynamodb:DescribeContinuousBackups"
  ],
  "Resource": {
    "Fn::Sub": [
      "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
      ${tableName}",
      {
        "tableName": {
          "Ref": "TableName"
        }
      }
    ]
  }
},
{
  "Effect": "Allow",
  "Action": [
    "dynamodb:DeleteBackup",
    "dynamodb:DescribeBackup",
    "dynamodb:ListBackups"
  ],
  "Resource": {
    "Fn::Sub": [
      "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
      ${tableName}/backup/*",
      {
        "tableName": {
          "Ref": "TableName"
        }
      }
    ]
  }
}
]
```

**DynamoDBRestoreFromBackupPolicy**

Gives permission to restore a DynamoDB table from backup.

```
"Statement": [{
  "Effect": "Allow",
  "Action": [
    "dynamodb:RestoreTableFromBackup"
  ],
  "Resource": {
    "Fn::Sub": [
      "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
      ${tableName}/backup/*",
      {
        "tableName": {
          "Ref": "TableName"
        }
      }
    ]
  }
}
]```
ComprehendBasicAccessPolicy

Gives permission for detecting entities, key phrases, languages, and sentiments.

```
"Statement": [{
   "Effect": "Allow",
   "Action": [
      "comprehend:BatchDetectKeyPhrases",
      "comprehend:DetectDominantLanguage",
      "comprehend:DetectEntities",
      "comprehend:BatchDetectEntities",
      "comprehend:DetectKeyPhrases",
      "comprehend:DetectSentiment",
      "comprehend:BatchDetectDominantLanguage",
      "comprehend:BatchDetectSentiment"
   ],
   "Resource": "*"
}
]
```

MobileAnalyticsWriteOnlyAccessPolicy

Gives write-only permission to put event data for all application resources.

```
"Statement": [
   {
      "Effect": "Allow",
      "Action": [
```
Policy template list

"mobileanalytics:PutEvents"
],
"Resource": "**"
}
]

PinpointEndpointAccessPolicy

Gives permission to get and update endpoints for an Amazon Pinpoint application.

"Statement": [
{
"Effect": "Allow",
"Action": [
"mobiletargeting:GetEndpoint",
"mobiletargeting:UpdateEndpoint",
"mobiletargeting:UpdateEndpointsBatch"
],
"Resource": {
"Fn::Sub": [
"arn:${AWS::Partition}:mobiletargeting:${AWS::Region}:
${AWS::AccountId}:apps/${pinpointApplicationId}/endpoints/*",
{
"pinpointApplicationId": {
"Ref": "PinpointApplicationId"
}
}
]
}
]

FirehoseWritePolicy

Gives permission to write to a Kinesis Data Firehose delivery stream.

"Statement": [
{
"Effect": "Allow",
"Action": [
"firehose:PutRecord",
"firehose:PutRecordBatch"
],
"Resource": {
"Fn::Sub": [
"arn:${AWS::Partition}:firehose:${AWS::Region}:
${AWS::AccountId}:deliverystream/${deliveryStreamName}"
,
{
"deliveryStreamName": {
"Ref": "DeliveryStreamName"
}
}
]
}
]
FirehoseCrudPolicy

Gives permission to create, write, update, and delete a Kinesis Data Firehose delivery stream.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "firehose:CreateDeliveryStream",
      "firehose:DeleteDeliveryStream",
      "firehose:DescribeDeliveryStream",
      "firehose:PutRecord",
      "firehose:PutRecordBatch",
      "firehose:UpdateDestination"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:firehose:${AWS::Region}:
        ${AWS::AccountId}:deliverystream/${deliveryStreamName}",
        
        "deliveryStreamName": {
          "Ref": "DeliveryStreamName"
        }
      ]
    }
  }
]
```

EKSDescribePolicy

Gives permission to describe or list Amazon Elastic Kubernetes Service (Amazon EKS) clusters.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "eks:DescribeCluster",
      "eks:ListClusters"
    ],
    "Resource": "*"
  }
]
```

CostExplorerReadOnlyPolicy

Gives read-only permission to the read-only AWS Cost Explorer (Cost Explorer) APIs for billing history.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ce:GetCostAndUsage",
      "ce:GetDimensionValues",
      "ce:GetReservationCoverage",
      "ce:GetReservationPurchaseRecommendation"
    ],
    "Resource": "*"
  }
]
```
OrganizationsListAccountsPolicy

Gives read-only permission to list child account names and IDs.

```
"Statement": [{
  "Effect": "Allow",
  "Action": ["organizations:ListAccounts"],
  "Resource": "*"
}]
```

SESBulkTemplatedCrudPolicy

Gives permission to send Amazon SES email, templated email, and templated bulk emails and to verify identity.

```
"Statement": [
  { "Effect": "Allow",
    "Action": ["ses:GetIdentityVerificationAttributes",
                "ses:SendEmail",
                "ses:SendRawEmail",
                "ses:SendTemplatedEmail",
                "ses:SendBulkTemplatedEmail",
                "ses:VerifyEmailIdentity"],
    "Resource": {
      "Fn::Sub": ["arn:${AWS::Partition}:ses:${AWS::Region}:${AWS::AccountId}:identity/${identityName}",
                   {"identityName": {"Ref": "IdentityName"}}]
    }
  }
]
```

SESEmailTemplateCrudPolicy

Gives permission to create, get, list, update, and delete Amazon SES email templates.

```
"Statement": [{
  "Effect": "Allow",
  "Action": ["ses:GetEmailTemplate",
```
"Effect": "Allow",
"Action": [
  "ses:CreateTemplate",
  "ses:GetTemplate",
  "ses:ListTemplates",
  "ses:UpdateTemplate",
  "ses:DeleteTemplate",
  "ses:TestRenderTemplate"
],
"Resource": "*
}

**FilterLogEventsPolicy**

Gives permission to filter CloudWatch Logs events from a specified log group.

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "logs:FilterLogEvents"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:logs:${AWS::Region}:${AWS::AccountId}:log-group:${logGroupName}:log-stream:*",
        {
          "logGroupName": {
            "Ref": "LogGroupName"
          }
        }
      ]
    }
  }
]

**SSMParameterReadPolicy**

Gives permission to access a parameters from an Amazon EC2 Systems Manager (SSM) parameter store to load secrets in this account.

**Note**

If you are not using default key, you will also need the **KMSDecryptPolicy** policy.

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ssm:DescribeParameters"
    ],
    "Resource": "*
  },
  {
    "Effect": "Allow",
    "Action": [
      "ssm:GetParameters",
      "ssm:GetParameter",
      292
"ssm:GetParametersByPath"
],
"Resource": {
  "Fn::Sub": [
    "arn:${AWS::Partition}:ssm:${AWS::Region}:${AWS::AccountId}:parameter/
    ${parameterName}",
    { "parameterName": { "Ref": "ParameterName" }
  }
  ]
}
]
}
]

**StepFunctionsExecutionPolicy**

Gives permission to start a Step Functions state machine execution.

"Statement": [
  { "Effect": "Allow", "Action": [ "states:StartExecution" ],
  "Resource": {
    "Fn::Sub": [
      "arn:${AWS::Partition}:states:${AWS::Region}:
      ${AWS::AccountId}:stateMachine:${stateMachineName}",
      { "stateMachineName": { "Ref": "StateMachineName" }
    }
  ]
  }
]

**CodeCommitCrudPolicy**

Gives permissions to create, read, update, and delete objects within a specific CodeCommit repository.

"Statement": [
  { "Effect": "Allow", "Action": [ "codecommit:GitPull",
    "codecommit:GitPush",
    "codecommit:CreateBranch",
    "codecommit:DeleteBranch",
    "codecommit:GetBranch",
    "codecommit:ListBranches",
    "codecommit:MergeBranchesByFastForward",
    "codecommit:MergeBranchesBySquash",
    "codecommit:MergeBranchesByThreeWay",
    "codecommit:UpdateDefaultBranch",
    "$ssm:GetParametersByPath"
  ]
  ]
]
"codecommit:BatchDescribeMergeConflicts",
"codecommit:CreateUnreferencedMergeCommit",
"codecommit:DescribeMergeConflicts",
"codecommit:GetMergeCommit",
"codecommit:GetMergeOptions",
"codecommit:BatchGetPullRequests",
"codecommit:CreatePullRequest",
"codecommit:DescribePullRequestEvents",
"codecommit:GetCommentsForPullRequest",
"codecommit:GetCommitsFromMergeBase",
"codecommit:GetMergeConflicts",
"codecommit:GetPullRequest",
"codecommit:ListPullRequests",
"codecommit:MergePullRequestByFastForward",
"codecommit:MergePullRequestBySquash",
"codecommit:MergePullRequestByThreeWay",
"codecommit:PostCommentForPullRequest",
"codecommit:UpdatePullRequestDescription",
"codecommit:UpdatePullRequestStatus",
"codecommit:UpdatePullRequestTitle",
"codecommit:DeleteFile",
"codecommit:GetBlob",
"codecommit:GetFile",
"codecommit:GetFolder",
"codecommit:PutFile",
"codecommit:DeleteCommentContent",
"codecommit:GetComment",
"codecommit:GetCommentsForComparedCommit",
"codecommit:PostCommentForComparedCommit",
"codecommit:PostCommentReply",
"codecommit:UpdateComment",
"codecommit:BatchGetCommits",
"codecommit:CreateCommit",
"codecommit:GetCommit",
"codecommit:GetCommitHistory",
"codecommit:GetDifferences",
"codecommit:GetObjectIdentifier",
"codecommit:GetReferences",
"codecommit:GetTree",
"codecommit:GetRepository",
"codecommit:UpdateRepositoryDescription",
"codecommit:ListTagsForResource",
"codecommit:TagResource",
"codecommit:UntagResource",
"codecommit:GetRepositoryTriggers",
"codecommit:PutRepositoryTriggers",
"codecommit:TestRepositoryTriggers",
"codecommit:GetBranch",
"codecommit:GetCommit",
"codecommit:UploadArchive",
"codecommit:GetUploadArchiveStatus",
"codecommit:CancelUploadArchive"
],
"Resource": {
  "Fn::Sub": [
    "arn:${AWS::Partition}:codecommit:${AWS::Region}:${AWS::AccountId}:
    {repositoryName}",
    {
      "repositoryName": {
        "Ref": "RepositoryName"
      }
    }
  ]}
**CodeCommitReadPolicy**

Gives permissions to read objects within a specific CodeCommit repository.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "codecommit:GitPull",
      "codecommit:GetBranch",
      "codecommit:ListBranches",
      "codecommit:BatchDescribeMergeConflicts",
      "codecommit:DescribeMergeConflicts",
      "codecommit:GetMergeCommit",
      "codecommit:GetMergeOptions",
      "codecommit:BatchGetPullRequests",
      "codecommit:DescribePullRequestEvents",
      "codecommit:GetCommentsForPullRequest",
      "codecommit:GetCommitsFromMergeBase",
      "codecommit:GetMergeConflicts",
      "codecommit:GetPullRequest",
      "codecommit:ListPullRequests",
      "codecommit:GetBlob",
      "codecommit:GetFile",
      "codecommit:GetFolder",
      "codecommit:GetComment",
      "codecommit:GetCommentsForComparedCommit",
      "codecommit:BatchGetCommits",
      "codecommit:GetCommit",
      "codecommit:GetCommitHistory",
      "codecommit:GetDifferences",
      "codecommit:GetObjectIdentifier",
      "codecommit:GetReferences",
      "codecommit:GetTree",
      "codecommit:GetRepository",
      "codecommit:ListTagsForResource",
      "codecommit:GetRepositoryTriggers",
      "codecommit:TestRepositoryTriggers",
      "codecommit:GetBranch",
      "codecommit:GetCommit",
      "codecommit:GetUploadArchiveStatus"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:codecommit:${AWS::Region}:${AWS::AccountId}:${repositoryName}",
        {
          "repositoryName": {
            "Ref": "RepositoryName"
          }
        }
      ]
    }
  }
]
```

**AthenaQueryPolicy**

Gives permissions to execute Athena queries.
"Statement": [  
  
  "Effect": "Allow",  
  "Action": [  
    "athena:ListWorkGroups",  
    "athena:GetExecutionEngine",  
    "athena:GetExecutionEngines",  
    "athena:GetNamespace",  
    "athena:GetCatalogs",  
    "athena:GetNamespaces",  
    "athena:GetTables",  
    "athena:GetTable"  
  ],  
  "Resource": "*"  
],  

"Effect": "Allow",  
"Action": [  
  "athena:StartQueryExecution",  
  "athena:GetQueryResults",  
  "athena:DeleteNamedQuery",  
  "athena:GetNamedQuery",  
  "athena:ListQueryExecutions",  
  "athena:StopQueryExecution",  
  "athena:GetQueryResultsStream",  
  "athena:ListNamedQueries",  
  "athena:CreateNamedQuery",  
  "athena:GetQueryExecution",  
  "athena:BatchGetNamedQuery",  
  "athena:BatchGetQueryExecution",  
  "athena:GetWorkGroup"  
],  

"Resource": {  
  "Fn::Sub": [  
    "arn:${AWS::Partition}:athena:${AWS::Region}:${AWS::AccountId}:workgroup/  
    #{'workgroupName}',  
    {  
      "workgroupName": {  
        "Ref": "WorkGroupName"  
      }  
    }  
  ]  
}  
]  

---

**TextractPolicy**

Gives full access to Amazon Textract.

"Statement": [  
  
  "Effect": "Allow",  
  "Action": [  
    "textract:*"  
  ],  
  "Resource": "*"  
]
**TextractDetectAnalyzePolicy**

Gives access to detect and analyze documents with Amazon Textract.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "textract:DetectDocumentText",
      "textract:StartDocumentTextDetection",
      "textract:StartDocumentAnalysis",
      "textract:AnalyzeDocument"
    ],
    "Resource": "*"
  }
]
```

**TextractGetResultPolicy**

Gives access to get detected and analyzed documents from Amazon Textract.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "textract:GetDocumentTextDetection",
      "textract:GetDocumentAnalysis"
    ],
    "Resource": "*"
  }
]
```

**EventBridgePutEventsPolicy**

Gives permissions to send events to Amazon EventBridge.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": "events:PutEvents",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:events:${AWS::Region}:${AWS::AccountId}:event-bus/
        ${eventBusName}",
        {
          "eventBusName": {  
            "Ref": "EventBusName"
          }
        }
      ]
    }
  }
]
```
**ElasticMapReduceModifyInstanceFleetPolicy**

Gives permission to list details and modify capacities for instance fleets within a cluster.

```json

"Statement": [ 

  { 
    "Action": [ 
      "elasticmapreduce:ModifyInstanceFleet",
      "elasticmapreduce:ListInstanceFleets"
    ],
    "Resource": { 
      "Fn::Sub": [ 
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:${AWS::AccountId}:cluster/${clusterId}",
        { 
          "clusterId": { 
            "Ref": "ClusterId"
          }
        }
      ],
      "Effect": "Allow"
    }
  }
]

```

**ElasticMapReduceSetTerminationProtectionPolicy**

Gives permission to set termination protection for a cluster.

```json

"Statement": [ 

  { 
    "Action": "elasticmapreduce:SetTerminationProtection",
    "Resource": { 
      "Fn::Sub": [ 
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:${AWS::AccountId}:cluster/${clusterId}",
        { 
          "clusterId": { 
            "Ref": "ClusterId"
          }
        }
      ],
      "Effect": "Allow"
    }
  }
]

```

**ElasticMapReduceModifyInstanceGroupsPolicy**

Gives permission to list details and modify settings for instance groups within a cluster.

```json

"Statement": [ 

  { 
    "Action": [ 
      "elasticmapreduce:ModifyInstanceGroups",
      "elasticmapreduce:ListInstanceGroups"
    ],
    "Resource": { 
      "Fn::Sub": [ 
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:${AWS::AccountId}:cluster/${clusterId}",
        { 
          "clusterId": { 
            "Ref": "ClusterId"
          }
        }
      ],
      "Effect": "Allow"
    }
  }
]

```
Policy template list

```
{
  "Resource": {
    "Fn::Sub": [
      "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:${AWS::AccountId}:cluster/${clusterId}",
      {
        "clusterId": {
          "Ref": "ClusterId"
        }
      }
    ],
    "Effect": "Allow"
  }
}
```

**ElasticMapReduceCancelStepsPolicy**

Gives permission to cancel a pending step or steps in a running cluster.

```
"Statement": [
  {
    "Action": "elasticmapreduce:CancelSteps",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:${AWS::AccountId}:cluster/${clusterId}",
        {
          "clusterId": {
            "Ref": "ClusterId"
          }
        }
      ],
      "Effect": "Allow"
    }
  }
]
```

**ElasticMapReduceTerminateJobFlowsPolicy**

Gives permission to shut down a cluster.

```
"Statement": [
  {
    "Action": "elasticmapreduce:TerminateJobFlows",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:${AWS::AccountId}:cluster/${clusterId}",
        {
          "clusterId": {
            "Ref": "ClusterId"
          }
        }
      ],
      "Effect": "Allow"
    }
  }
]
```
ElasticMapReduceAddJobFlowStepsPolicy

Gives permission to add new steps to a running cluster.

```
"Statement": [
  {
    "Action": "elasticmapreduce:AddJobFlowSteps",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:${AWS::AccountId}:cluster/${clusterId}",
        {
          "clusterId": {
            "Ref": "ClusterId"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

SageMakerCreateEndpointPolicy

Gives permission to create an endpoint in SageMaker.

```
"Statement": [
  {
    "Action": [
      "sagemaker:CreateEndpoint"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:sagemaker:${AWS::Region}:${AWS::AccountId}:endpoint/${endpointName}",
        {
          "endpointName": {
            "Ref": "EndpointName"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

SageMakerCreateEndpointConfigPolicy

Gives permission to create an endpoint configuration in SageMaker.

```
"Statement": [
  {
    "Action": [
      "sagemaker:CreateEndpointConfig"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:sagemaker:${AWS::Region}:${AWS::AccountId}:endpointconfig/${endpointConfigName}",
        {
          "endpointConfigName": {
            "Ref": "EndpointConfigName"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```
EcsRunTaskPolicy

Gives permission to start a new task for a task definition.

EFSWriteAccessPolicy

Gives permission to mount an Amazon EFS file system with write access.
Telemetry in the AWS SAM CLI

At AWS, we develop and launch services based on what we learn from interactions with customers. We use customer feedback to iterate on our product. Telemetry is additional information that helps us to better understand our customers’ needs, diagnose issues, and deliver features that improve the customer experience.

The AWS SAM CLI collects telemetry, such as generic usage metrics, system and environment information, and errors. For details of the types of telemetry collected, see Types of information collected (p. 303).

The AWS SAM CLI does not collect personal information, such as usernames or email addresses. It also does not extract sensitive project-level information.

Customers control whether telemetry is enabled, and can change their settings at any point of time. If telemetry remains enabled, the AWS SAM CLI sends telemetry data in the background without requiring any additional customer interaction.

Disabling telemetry for a session

In macOS and Linux operating systems, you can disable telemetry for a single session. To disable telemetry for your current session, run the following command to set the environment variable SAM_CLI_TELEMETRY to false. You must repeat the command for each new terminal or session.

```
export SAM_CLI_TELEMETRY=0
```

Disabling telemetry for your profile in all sessions

Run the following commands to disable telemetry for all sessions when you're running the AWS SAM CLI on your operating system.

To disable telemetry in Linux

1. Run:
Types of information collected

- Usage information – The generic commands and subcommands that are run.
- Errors and diagnostic information – The status and duration of commands that are run, including exit codes, internal exception names, and failures when connecting to Docker.
- System and environment information – The Python version, operating system (Windows, Linux, or macOS), and environment in which the AWS SAM CLI is executed (for example, AWS CodeBuild, an AWS IDE toolkit, or a terminal).

Learn more

The telemetry data that’s collected adheres to the AWS data privacy policies. For more information, see the following:

- AWS Service Terms
- Data Privacy

Permissions

To control access to AWS resources, AWS SAM uses the same mechanisms as AWS CloudFormation. For more information, see Controlling access with AWS Identity and Access Management in the AWS CloudFormation User Guide.
There are three main options for granting a user permission to manage serverless applications. Each option provides users with different levels of access control.

- Grant administrator permissions.
- Attach necessary AWS managed policies.
- Grant specific AWS Identity and Access Management (IAM) permissions.

Depending on which option you choose, users can manage only serverless applications containing AWS resources that they have permission to access.

The following sections describe each option in more detail.

**Grant administrator permissions**

If you grant administrator permissions to a user, they can manage serverless applications that contain any combination of AWS resources. This is the simplest option, but it also grants users the broadest set of permissions, which therefore enables them to perform actions with the highest impact.

For more information about granting administrator permissions to a user, see Creating your first IAM admin user and group in the *IAM User Guide*.

**Attach necessary AWS managed policies**

You can grant users a subset of permissions using AWS managed policies, rather than granting full administrator permissions. If you use this option, make sure that the set of AWS managed policies covers all of the actions and resources required for the serverless applications that the users manage.

For example, the following AWS managed policies are sufficient to deploy the sample Hello World application (p. 14):

- AWSCloudFormationFullAccess
- IAMFullAccess
- AWSLambda_FullAccess
- AmazonAPIGatewayAdministrator
- AmazonS3FullAccess
- AmazonEC2ContainerRegistryFullAccess

For information about attaching policies to an IAM user, see Changing permissions for an IAM user in the *IAM User Guide*.

**Grant specific IAM permissions**

For the most granular level of access control, you can grant specific IAM permissions to users using policy statements. If you use this option, make sure that the policy statement includes all of the actions and resources required for the serverless applications that the users manage.

The best practice with this option is to deny users the permission to create roles, including Lambda execution roles, so they can't grant themselves escalated permissions. So, you as the administrator must first create a Lambda execution role that will be specified in the serverless applications that users will manage. For information about creating Lambda execution roles, see Creating an execution role in the IAM console.

For the sample Hello World application (p. 14) the *AWSLambdaBasicExecutionRole* is sufficient to run the application. After you've created a Lambda execution role, modify the AWS SAM template file of the...
sample Hello World application to add the following property to the AWS::Serverless::Function resource:

| Role: lambda-execution-role-arn |

With the modified Hello World application in place, the following policy statement grants sufficient permissions for users to deploy, update, and delete the application:

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CloudFormationTemplate",
            "Effect": "Allow",
            "Action": [
                "cloudformation:CreateChangeSet"
            ],
            "Resource": [
                "arn:aws:cloudformation:*:aws:transform/Serverless-2016-10-31"
            ]
        },
        {
            "Sid": "CloudFormationStack",
            "Effect": "Allow",
            "Action": [
                "cloudformation:CreateChangeSet",
                "cloudformation:DeleteStack",
                "cloudformation:DescribeChangeSet",
                "cloudformation:DescribeStackEvents",
                "cloudformation:DescribeStacks",
                "cloudformation:ExecuteChangeSet",
                "cloudformation:GetTemplateSummary"
            ],
            "Resource": [
                "arn:aws:cloudformation:*:111122223333:stack/*"
            ]
        },
        {
            "Sid": "S3",
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:305/*"
            ]
        },
        {
            "Sid": "ECRRepository",
            "Effect": "Allow",
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:BatchGetImage",
                "ecr:CompleteLayerUpload",
                "ecr:DescribeImages",
                "ecr:DescribeRepositories",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetRepositoryPolicy",
                "ecr:InitiateLayerUpload",
                "ecr:ListImages",
                "ecr:PutImage",
                "ecr:SetRepositoryPolicy"
            ]
        }
    ]
}
```
Grant specific IAM permissions

```
"ecr:UploadLayerPart"
],
"Resource": [  
  "arn:aws:ecr::*:111122223333:repository/*"
],
"Sid": "ECRAuthToken",
"Effect": "Allow",
"Action": [  
  "ecr:GetAuthorizationToken"
],
"Resource": [  
  "*
]
},
{
"Sid": "Lambda",
"Effect": "Allow",
"Action": [  
  "lambda:AddPermission",
  "lambda:CreateFunction",
  "lambda:DeleteFunction",
  "lambda:GetFunction",
  "lambda:GetFunctionConfiguration",
  "lambda:ListTags",
  "lambda:RemovePermission",
  "lambda:TagResource",
  "lambda:UntagResource",
  "lambda:UpdateFunctionCode",
  "lambda:UpdateFunctionConfiguration"
],
"Resource": [  
  "arn:aws:lambda::*:111122223333:function:*"
]
},
{
"Sid": "IAM",
"Effect": "Allow",
"Action": [  
  "iam:AttachRolePolicy",
  "iam:DeleteRole",
  "iam:DetachRolePolicy",
  "iam:GetRole",
  "iam:PassRole",
  "iam:TagRole"
],
"Resource": [  
  "arn:aws:iam::*:111122223333:role/*"
]
},
{
"Sid": "APIGateway",
"Effect": "Allow",
"Action": [  
  "apigateway:DELETE",
  "apigateway:GET",
  "apigateway:PATCH",
  "apigateway:POST",
  "apigateway:PUT"
],
"Resource": [  
  "arn:aws:apigateway:*
]
}
```

306
Important notes

This section contains important notes and known issues for AWS Serverless Application Model.

Installing AWS SAM CLI on 32-bit Windows

Support for AWS SAM CLI on 32-bit Windows will soon be deprecated. If you operate on a 32-bit system, we recommend that you upgrade to a 64-bit system and follow the instructions found in Installing the AWS SAM CLI on Windows (p. 9).

If you cannot upgrade to a 64-bit system, you can use the Legacy Docker Toolbox with AWS SAM CLI on a 32-bit system. However, this will cause you to encounter certain limitations with the AWS SAM CLI. For example, you cannot run 64-bit Docker containers on a 32-bit system. So, if your Lambda function depends on a 64-bit natively compiled container, you will not be able to test it locally on a 32-bit system.

To install AWS SAM CLI on a 32-bit system, execute the following command:

```
$ pip install aws-sam-cli
```

**Important**

Although the `pip install aws-sam-cli` command also works on 64-bit Windows, we recommend that you use the 64-bit MSI to install AWS SAM CLI on 64-bit systems.
# Document history for AWS SAM

The following table describes the important changes in each release of the *AWS Serverless Application Model Developer Guide*. For notifications about updates to this documentation, you can subscribe to an RSS feed.

- **Latest documentation update:** March 4, 2021

<table>
<thead>
<tr>
<th>update-history-change</th>
<th>update-history-description</th>
<th>update-history-date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build container environment variables support (p. 308)</td>
<td>You can now pass environment variables to build containers. For more information, see the <code>--container-env-var</code> and <code>--container-env-var-file</code> options in <code>sam build</code>.</td>
<td>March 4, 2021</td>
</tr>
<tr>
<td>New Linux installation process (p. 308)</td>
<td>You can now install the AWS SAM CLI using a native Linux installer. For more information, see Installing the AWS SAM CLI on Linux.</td>
<td>February 10, 2021</td>
</tr>
<tr>
<td>Support for dead-letter queues for EventBridge (p. 308)</td>
<td>Added support for dead-letter queues for EventBridge and Schedule event sources for serverless functions and state machines. For more information, see the <code>DeadLetterConfig</code> property of the <code>EventBridgeRule</code> and <code>Schedule</code> event sources, for both the <code>AWS::Serverless::Function</code> and <code>AWS::Serverless::StateMachine</code> resource types.</td>
<td>January 29, 2021</td>
</tr>
<tr>
<td>Support for custom checkpoints (p. 308)</td>
<td>Added support for custom checkpoints for DynamoDB and Kinesis event sources for serverless functions. For more information, see the <code>FunctionResponseTypes</code> property of the <code>Kinesis</code> and <code>DynamoDB</code> data types of the <code>AWS::Serverless::Function</code> resource type.</td>
<td>January 29, 2021</td>
</tr>
<tr>
<td>Support for tumbling windows (p. 308)</td>
<td>Added support for tumbling windows for DynamoDB and Kinesis event sources for serverless functions. For more information, see the</td>
<td>December 17, 2020</td>
</tr>
<tr>
<td>Features</td>
<td>Details</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>Support for warm containers (p. 308)</td>
<td>Added support for warm containers when testing locally using the AWS SAM CLI commands <code>sam local start-api</code> and <code>sam local start-lambda</code>. For more information, see the <code>--warm-containers</code> option for those commands.</td>
<td></td>
</tr>
<tr>
<td>Support for Lambda container images (p. 308)</td>
<td>Added support for Lambda container images. For more information, see Building applications.</td>
<td></td>
</tr>
<tr>
<td>Support for code signing (p. 308)</td>
<td>Added support for code signing and trusted deployments of serverless application code. For more information, see Configuring code signing for AWS SAM applications.</td>
<td></td>
</tr>
<tr>
<td>Support for parallel and cached builds (p. 308)</td>
<td>Improved performance of serverless application builds by adding two options to the <code>sam build</code> command: <code>--parallel</code>, which builds functions and layers in parallel rather than sequentially, and <code>--cached</code>, which uses build artifacts from previous builds when no changes have been made that requires a rebuild.</td>
<td></td>
</tr>
<tr>
<td>Feature</td>
<td>Added Support Description</td>
<td>Date</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Support for Amazon MQ, and mutual TLS authentication (p. 308)</td>
<td>Added support for Amazon MQ as an event source for serverless functions. For more information, see the EventSource and MQ data types of the AWS::Serverless::Function resource type. Also added support for mutual Transport Layer Security (TLS) authentication for API Gateway APIs and HTTP APIs. For more information, see the DomainConfiguration data type of the AWS::Serverless::Api resource type, or the HttpApiDomainConfiguration data type of the AWS::Serverless::HttpApi resource type.</td>
<td>November 5, 2020</td>
</tr>
<tr>
<td>Support for Lambda authorizers for HTTP APIs (p. 308)</td>
<td>Added support for Lambda authorizers for the AWS::Serverless::HttpApi resource type. For more information, see Lambda authorizer example (AWS::Serverless::HttpApi).</td>
<td>October 27, 2020</td>
</tr>
<tr>
<td>Support for multiple configuration files and environments (p. 308)</td>
<td>Added support for multiple configuration files and environments to store default parameter values for AWS SAM CLI commands. For more information, see AWS SAM CLI configuration file.</td>
<td>September 24, 2020</td>
</tr>
<tr>
<td>Support for X-Ray with Step Functions, and references when controlling access to APIs (p. 308)</td>
<td>Added support for X-Ray as an event source for serverless state machines. For more information, see the Tracing property of the AWS::Serverless::StateMachine resource type. Also added support for references when controlling access to APIs. For more information, see the ResourcePolicyStatement data type.</td>
<td>September 17, 2020</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
<td>Date</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Support for Amazon MSK (p. 308)</td>
<td>Added support for Amazon MSK as an event source for serverless functions. This allows records in an Amazon MSK topic to trigger your Lambda function. For more information, see the EventSource and MSK data types of the AWS::Serverless::Function resource type.</td>
<td>August 13, 2020</td>
</tr>
<tr>
<td>Support for Amazon EFS (p. 308)</td>
<td>Added support for mounting Amazon EFS file systems to local directories. This allows your Lambda function code to access and modify shared resources. For more information, see the FileSystemConfigs property of the AWS::Serverless::Function resource type.</td>
<td>June 16, 2020</td>
</tr>
<tr>
<td>Orchestrating serverless applications (p. 308)</td>
<td>Added support for orchestrating applications by creating Step Functions state machines using AWS SAM. For more information, see Orchestrating AWS resources with AWS Step Functions and the AWS::Serverless::StateMachine resource type.</td>
<td>May 27, 2020</td>
</tr>
<tr>
<td>Building custom runtimes (p. 308)</td>
<td>Added the ability to build custom runtimes. For more information, see Building custom runtimes.</td>
<td>May 21, 2020</td>
</tr>
<tr>
<td>Building layers (p. 308)</td>
<td>Added the ability to build individual LayerVersion resources. For more information, see Building layers.</td>
<td>May 19, 2020</td>
</tr>
<tr>
<td>Generated AWS CloudFormation resources (p. 308)</td>
<td>Provided details about the AWS CloudFormation resources that AWS SAM generates and how to reference them. For more information, see Generated AWS CloudFormation resources.</td>
<td>April 8, 2020</td>
</tr>
<tr>
<td>Setting up AWS credentials (p. 308)</td>
<td>Added instructions for setting up AWS credentials in case you haven't already set them to use with other AWS tools, such as one of the AWS SDKs or the AWS CLI. For more information, see Setting up AWS credentials.</td>
<td>January 17, 2020</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
<td>Date</td>
</tr>
<tr>
<td>-----------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>AWS SAM specification and AWS SAM CLI updates</td>
<td>Migrated the AWS SAM specification from GitHub. For more information, see AWS SAM specification. Also updated the deployment workflow with changes to the <code>sam deploy</code> command.</td>
<td>November 25, 2019</td>
</tr>
<tr>
<td>New options for controlling access to API Gateway APIs and policy template updates</td>
<td>Added new options for controlling access to API Gateway APIs: IAM permissions, API keys, and resource policies. For more information, see Controlling access to API Gateway APIs. Also updated two policy templates: RekognitionFacesPolicy and ElasticsearchHttpPostPolicy. For more information, see AWS SAM policy templates.</td>
<td>August 29, 2019</td>
</tr>
<tr>
<td>Getting started updates</td>
<td>Updated the getting started chapter with improved installation instructions for the AWS SAM CLI and the Hello World tutorial. For more information, see Getting started with AWS SAM.</td>
<td>July 25, 2019</td>
</tr>
<tr>
<td>Controlling access to API Gateway APIs</td>
<td>Added support for controlling access to API Gateway APIs. For more information, see Controlling access to API Gateway APIs.</td>
<td>March 21, 2019</td>
</tr>
<tr>
<td>Added <code>sam publish</code> to the AWS SAM CLI</td>
<td>The new <code>sam publish</code> command in the AWS SAM CLI simplifies the process for publishing serverless applications in the AWS Serverless Application Repository. For more information, see Publishing serverless applications using the AWS SAM CLI.</td>
<td>December 21, 2018</td>
</tr>
<tr>
<td>Nested applications and layers support</td>
<td>Added support for nested applications and layers. For more information, see Using nested applications and Working with layers.</td>
<td>November 29, 2018</td>
</tr>
<tr>
<td>Added sam build to the AWS SAM CLI (p. 308)</td>
<td>The new <code>sam build</code> command in the AWS SAM CLI simplifies the process for compiling serverless applications with dependencies so that you can locally test and deploy these applications. For more information, see Building applications.</td>
<td>November 19, 2018</td>
</tr>
<tr>
<td>Added new installation options for the AWS SAM CLI (p. 308)</td>
<td>Added Linuxbrew (Linux), MSI (Windows), and Homebrew (macOS) installation options for the AWS SAM CLI. For more information, see Installing the AWS SAM CLI.</td>
<td>November 7, 2018</td>
</tr>
<tr>
<td>New guide (p. 308)</td>
<td>This is the first release of the AWS Serverless Application Model Developer Guide.</td>
<td>October 17, 2018</td>
</tr>
</tbody>
</table>