AWS Serverless Application Model
Developer Guide
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is AWS SAM?</td>
<td>1</td>
</tr>
<tr>
<td>Benefits of using AWS SAM</td>
<td>1</td>
</tr>
<tr>
<td>Next step</td>
<td>2</td>
</tr>
<tr>
<td>Getting started</td>
<td>3</td>
</tr>
<tr>
<td>Installing the AWS SAM CLI</td>
<td>3</td>
</tr>
<tr>
<td>Linux</td>
<td>3</td>
</tr>
<tr>
<td>Windows</td>
<td>7</td>
</tr>
<tr>
<td>macOS</td>
<td>8</td>
</tr>
<tr>
<td>Setting up AWS credentials</td>
<td>11</td>
</tr>
<tr>
<td>Using the AWS CLI</td>
<td>11</td>
</tr>
<tr>
<td>Not using the AWS CLI</td>
<td>11</td>
</tr>
<tr>
<td>Tutorial: Deploying a Hello World application</td>
<td>12</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>12</td>
</tr>
<tr>
<td>Step 1: Download a sample AWS SAM application</td>
<td>13</td>
</tr>
<tr>
<td>Step 2: Build your application</td>
<td>14</td>
</tr>
<tr>
<td>Step 3: Deploy your application to the AWS Cloud</td>
<td>14</td>
</tr>
<tr>
<td>Step 4: Testing your application locally (optional)</td>
<td>17</td>
</tr>
<tr>
<td>Troubleshooting</td>
<td>19</td>
</tr>
<tr>
<td>Clean up</td>
<td>21</td>
</tr>
<tr>
<td>Conclusion</td>
<td>21</td>
</tr>
<tr>
<td>Next steps</td>
<td>21</td>
</tr>
<tr>
<td>AWS SAM specification</td>
<td>23</td>
</tr>
<tr>
<td>Template anatomy</td>
<td>23</td>
</tr>
<tr>
<td>YAML</td>
<td>23</td>
</tr>
<tr>
<td>Template sections</td>
<td>24</td>
</tr>
<tr>
<td>Next steps</td>
<td>25</td>
</tr>
<tr>
<td>Globals</td>
<td>25</td>
</tr>
<tr>
<td>Resource and property reference</td>
<td>29</td>
</tr>
<tr>
<td>AWS::Serverless::Api</td>
<td>29</td>
</tr>
<tr>
<td>AWS::Serverless::Application</td>
<td>60</td>
</tr>
<tr>
<td>AWS::Serverless::Function</td>
<td>63</td>
</tr>
<tr>
<td>AWS::Serverless::HttpApi</td>
<td>114</td>
</tr>
<tr>
<td>AWS::Serverless::LayerVersion</td>
<td>129</td>
</tr>
<tr>
<td>AWS::Serverless::SimpleTable</td>
<td>132</td>
</tr>
<tr>
<td>AWS::Serverless::StateMachine</td>
<td>135</td>
</tr>
<tr>
<td>Resource attributes</td>
<td>151</td>
</tr>
<tr>
<td>Intrinsic functions</td>
<td>152</td>
</tr>
<tr>
<td>Generated resources</td>
<td>152</td>
</tr>
<tr>
<td>Referencing generated AWS CloudFormation resources</td>
<td>152</td>
</tr>
<tr>
<td>Generated AWS CloudFormation resource reference</td>
<td>153</td>
</tr>
<tr>
<td>AWS::Serverless::Api</td>
<td>154</td>
</tr>
<tr>
<td>AWS::Serverless::Function</td>
<td>155</td>
</tr>
<tr>
<td>AWS::Serverless::HttpApi</td>
<td>157</td>
</tr>
<tr>
<td>API Gateway extensions</td>
<td>158</td>
</tr>
<tr>
<td>Authoring</td>
<td>160</td>
</tr>
<tr>
<td>Validating AWS SAM template files</td>
<td>160</td>
</tr>
<tr>
<td>Working with layers</td>
<td>160</td>
</tr>
<tr>
<td>Using nested applications</td>
<td>162</td>
</tr>
<tr>
<td>Defining a nested application from the AWS Serverless Application Repository</td>
<td>162</td>
</tr>
<tr>
<td>Defining a nested application from the local file system</td>
<td>163</td>
</tr>
<tr>
<td>Deploying nested applications</td>
<td>164</td>
</tr>
<tr>
<td>Controlling access to APIs</td>
<td>164</td>
</tr>
<tr>
<td>Choosing a mechanism to control access</td>
<td>165</td>
</tr>
<tr>
<td>Example: Defining Lambda token authorizers</td>
<td>165</td>
</tr>
<tr>
<td>Page</td>
<td>Section</td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
</tr>
<tr>
<td>166</td>
<td>Example: Defining Lambda request authorizers</td>
</tr>
<tr>
<td>167</td>
<td>Example: Defining Amazon Cognito user pools</td>
</tr>
<tr>
<td>168</td>
<td>Example: Defining IAM permissions</td>
</tr>
<tr>
<td>168</td>
<td>Example: Defining API keys</td>
</tr>
<tr>
<td>168</td>
<td>Example: Defining resource policies</td>
</tr>
<tr>
<td>169</td>
<td>Example: Defining customized responses</td>
</tr>
<tr>
<td>170</td>
<td>Orchestrating applications</td>
</tr>
<tr>
<td>170</td>
<td>Example</td>
</tr>
<tr>
<td>171</td>
<td>More information</td>
</tr>
<tr>
<td>172</td>
<td>Building</td>
</tr>
<tr>
<td>172</td>
<td>Building applications</td>
</tr>
<tr>
<td>172</td>
<td>Examples</td>
</tr>
<tr>
<td>173</td>
<td>Building layers</td>
</tr>
<tr>
<td>172</td>
<td>Examples</td>
</tr>
<tr>
<td>174</td>
<td>Building custom runtimes</td>
</tr>
<tr>
<td>174</td>
<td>Examples</td>
</tr>
<tr>
<td>176</td>
<td>Testing and debugging</td>
</tr>
<tr>
<td>176</td>
<td>Invoking Functions Locally</td>
</tr>
<tr>
<td>177</td>
<td>Environment Variable File</td>
</tr>
<tr>
<td>178</td>
<td>Layers</td>
</tr>
<tr>
<td>178</td>
<td>Running API Gateway locally</td>
</tr>
<tr>
<td>180</td>
<td>Layers</td>
</tr>
<tr>
<td>180</td>
<td>Integrating with automated tests</td>
</tr>
<tr>
<td>182</td>
<td>Generating sample event payloads</td>
</tr>
<tr>
<td>182</td>
<td>Step-through debugging Lambda functions locally</td>
</tr>
<tr>
<td>183</td>
<td>Using AWS Toolkits</td>
</tr>
<tr>
<td>183</td>
<td>Running AWS SAM locally</td>
</tr>
<tr>
<td>185</td>
<td>Node.js</td>
</tr>
<tr>
<td>187</td>
<td>Python</td>
</tr>
<tr>
<td>187</td>
<td>Golang</td>
</tr>
<tr>
<td>188</td>
<td>Passing additional runtime debug arguments</td>
</tr>
<tr>
<td>189</td>
<td>Deploying</td>
</tr>
<tr>
<td>189</td>
<td>Deploying using the AWS SAM CLI</td>
</tr>
<tr>
<td>189</td>
<td>Publishing serverless applications</td>
</tr>
<tr>
<td>189</td>
<td>Automating deployments</td>
</tr>
<tr>
<td>190</td>
<td>Troubleshooting</td>
</tr>
<tr>
<td>19</td>
<td>SAM CLI error: “Security Constraints Not Satisfied”</td>
</tr>
<tr>
<td>190</td>
<td>Deploying gradually</td>
</tr>
<tr>
<td>193</td>
<td>Monitoring</td>
</tr>
<tr>
<td>193</td>
<td>Working with logs</td>
</tr>
<tr>
<td>193</td>
<td>Fetching logs by AWS CloudFormation stack</td>
</tr>
<tr>
<td>193</td>
<td>Fetching logs by Lambda function name</td>
</tr>
<tr>
<td>193</td>
<td>Tailing logs</td>
</tr>
<tr>
<td>193</td>
<td>Viewing logs for a specific time range</td>
</tr>
<tr>
<td>193</td>
<td>Filtering logs</td>
</tr>
<tr>
<td>194</td>
<td>Error highlighting</td>
</tr>
<tr>
<td>194</td>
<td>JSON pretty printing</td>
</tr>
<tr>
<td>195</td>
<td>Publishing</td>
</tr>
<tr>
<td>195</td>
<td>Prerequisites</td>
</tr>
<tr>
<td>196</td>
<td>Publishing a new application</td>
</tr>
<tr>
<td>196</td>
<td>Step 1: Add a Metadata section to the AWS SAM template</td>
</tr>
<tr>
<td>196</td>
<td>Step 2: Package the application</td>
</tr>
<tr>
<td>197</td>
<td>Step 3: Publish the application</td>
</tr>
<tr>
<td>197</td>
<td>Publishing a new version of an existing application</td>
</tr>
<tr>
<td>197</td>
<td>Additional topics</td>
</tr>
<tr>
<td>198</td>
<td>Metadata section properties</td>
</tr>
<tr>
<td>198</td>
<td>Properties</td>
</tr>
</tbody>
</table>
Use cases .......................................................................................................................... 199
Example ........................................................................................................................... 200
Example applications ........................................................................................................... 201
Process DynamoDB events ................................................................................................. 201
  Before you begin ........................................................................................................... 201
  Step 1: Initialize the application .................................................................................. 201
  Step 2: Test the application locally ............................................................................... 201
  Step 3: Package the application ................................................................................... 202
  Step 4: Deploy the application ..................................................................................... 202
  Next steps ...................................................................................................................... 203
Process Amazon S3 events ................................................................................................. 203
  Before you begin .......................................................................................................... 203
  Step 1: Initialize the application .................................................................................. 203
  Step 2: Package the application ................................................................................... 204
  Step 3: Deploy the application ..................................................................................... 204
  Step 4: Test the application locally ............................................................................... 205
  Next steps ...................................................................................................................... 205
AWS SAM reference ........................................................................................................... 206
  AWS SAM specification ................................................................................................. 206
  AWS SAM CLI command reference ............................................................................... 206
  AWS SAM policy templates ........................................................................................... 206
AWS SAM CLI command reference ......................................................................................... 207
  sam build ...................................................................................................................... 207
  sam deploy ................................................................................................................... 209
  sam init ......................................................................................................................... 211
  sam local generate-event .............................................................................................. 213
  sam local invoke .......................................................................................................... 215
  sam local start-api ....................................................................................................... 216
  sam local start-lambda ................................................................................................. 218
  sam logs ....................................................................................................................... 220
  sam package ................................................................................................................. 221
  sam publish .................................................................................................................. 222
  sam validate .................................................................................................................. 223
AWS SAM CLI configuration file .......................................................................................... 224
  Example ......................................................................................................................... 224
  Writing configurations with sam deploy --guided ....................................................... 225
AWS SAM policy templates ................................................................................................. 226
  Syntax .......................................................................................................................... 226
  Examples ....................................................................................................................... 226
  Policy template table ..................................................................................................... 227
  Troubleshooting ............................................................................................................ 231
  Policy template list ......................................................................................................... 231
AWS SAM CLI telemetry ....................................................................................................... 267
  Disabling telemetry for a session .................................................................................. 267
  Disabling telemetry for your profile in all sessions ...................................................... 267
  Types of information collected ...................................................................................... 268
  Learn more .................................................................................................................... 268
Important notes .................................................................................................................. 268
  Installing AWS SAM CLI on 32-bit Windows ............................................................... 269
Document history ................................................................................................................. 270
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. 23).

- **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. 206).

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. 12) 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. 11)
- Tutorial: Deploying a Hello World application (p. 12)

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. 7)
- Installing the AWS SAM CLI on macOS (p. 8)

**Installing the AWS SAM CLI on Linux**

The following steps help you to install and configure the required prerequisites for using the AWS SAM CLI on your Linux host:

1. Create an AWS account.
2. Configure IAM permissions.
3. Install Docker. Note: Docker is only a prerequisite for testing your application locally.
4. Install Homebrew.
5. Install the AWS SAM CLI.

**Note**

These instructions cause your environment's default Python version to be the one installed by Homebrew. The change in default Python version occurs in Step 4: Install Homebrew (p. 5).

**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 Create and Activate an AWS Account.
Step 2: Create an IAM user with administrator permissions

If you don't already have an IAM user with administrator permissions, see Creating Your First IAM Admin User and Group in the IAM User Guide.

In addition, you must set up AWS credentials to enable the AWS SAM CLI to make AWS service calls. For example, the AWS SAM CLI makes calls to Amazon S3 and AWS CloudFormation. For more information about setting up AWS credentials, see Setting up AWS credentials (p. 11).

Step 3: Install Docker

Note
Docker is only a prerequisite for testing your application locally and to build deployment packages using the --use-container flag. You may skip this section or install Docker at a later time if you do not plan to use these features initially.

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.

You must have Docker installed and working to be able to run serverless projects and functions locally with the AWS SAM CLI. 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, like CentOS, Debian, Ubuntu, etc. For more information about how to install Docker on your particular operating system, go to the Docker installation guide.

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

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

   ```
   sudo yum update -y
   ```

2. Install the most recent Docker Community Edition package.

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

3. Start the Docker service.

   ```
   sudo service docker start
   ```

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

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

5. Log out and log back in again to pick up the new docker group permissions. You can accomplish this by closing your current SSH terminal window and reconnecting to your instance in a new one. Your new SSH session will have the appropriate docker group permissions.

6. Verify that the ec2-user can run Docker commands without sudo.

   ```
   docker ps
   ```

You should see the following output, showing Docker is installed and running:
Note
In some cases, you may need to reboot your instance to provide permissions for the `ec2-user` to access the Docker daemon. Try rebooting your instance if you see the following error:

```
Cannot connect to the Docker daemon. Is the docker daemon running on this host?
```

If you run into issues installing Docker, see the Troubleshooting (p. 6) section later in this guide, or the Troubleshooting section of the Docker installation guide for additional troubleshooting tips.

## Step 4: Install Homebrew

**Note**
This step causes your environment's default Python version to be the one installed by Homebrew.

The recommended approach for installing the AWS SAM CLI on Linux 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 most modern Linux distributions. 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:

```
/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/Ubuntu or `~/.bash_profile` on CentOS/Fedora/RedHat:

```
test -d ~/.linuxbrew && eval $(~/.linuxbrew/bin/brew shellenv)
test -d /home/linuxbrew/.linuxbrew && eval $(/home/linuxbrew/.linuxbrew/bin/brew shellenv)
test -r ~/.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
```

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

```
Homebrew 2.1.6
Homebrew/homebrew-core (git revision ef21; last commit 2019-06-19)
```

## Step 5: Install the AWS SAM CLI

Follow these steps to install the AWS SAM CLI using Homebrew:
brew tap aws/tap
brew install aws-sam-cli

Verify the installation:

sam --version

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

SAM CLI, version 1.3.0

You're now ready to start development.

Upgrading

To upgrade the AWS SAM CLI, you still use Homebrew, but replace install with upgrade as follows:

brew upgrade aws-sam-cli

Troubleshooting

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

In some cases, you may need to reboot your instance to provide permissionst for the ec2-user to access the Docker daemon. If you receive this error, try rebooting your instance.

Shell error: "command not found"

Your shell is not able to locate the AWS SAM CLI executable in the path. If you receive this error, verify the location of directory where the AWS SAM CLI executable was installed, and verify that directory is on your path.

For example, if you used the instructions in this topic to both 1) Install Homebrew, and 2) Use Homebrew to install the AWS SAM CLI, then the AWS SAM CLI executable will be installed to the following location:

/home/linuxbrew/.linuxbrew/bin/sam

Error: "AWS SAM CLI no longer supports installations on Python 2.7."

Your system is using an old installation of Python that is not compatible with the AWS SAM CLI. If you receive this error, follow the instructions in Installing the AWS SAM CLI on Linux (p. 3) to install a version of Python compatible with 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. 12) – 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 Windows

The following steps help you to install and configure the required prerequisites for using the AWS SAM CLI on your Windows host:

1. Create an AWS account.
2. Configure IAM permissions.
3. Install Docker. Note: Docker is only a prerequisite for testing your application locally.
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: Create an IAM user with administrator permissions

If you don’t already have an IAM user with administrator permissions, see *Creating Your First IAM Admin User and Group* in the [IAM User Guide](http://aws.amazon.com).

In addition, you must set up AWS credentials to enable the AWS SAM CLI to make AWS service calls. For example, the AWS SAM CLI makes calls to Amazon S3 and AWS CloudFormation. For more information about setting up AWS credentials, see *Setting up AWS credentials (p. 11)*.

### Step 3: Install Docker

**Note**

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

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.

You must have Docker installed and working to be able to run serverless projects and functions locally with the AWS SAM CLI. 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](http://aws.amazon.com).
   • 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 AWS SAM CLI (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. 269).

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:

   ```text
   SAM CLI, version 1.3.0
   ```

You’re now ready to start development.

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. 12) – 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

The following steps help you to install and configure the required prerequisites for using the AWS SAM CLI on your macOS host:

1. Create an AWS account.
2. Configure IAM permissions.
3. Install Docker. Note: Docker is only a prerequisite for testing your application locally.
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 Create and Activate an AWS Account.

**Step 2: Create an IAM user with administrator permissions**

If you don't already have an IAM user with administrator permissions, see Creating Your First IAM Admin User and Group in the IAM User Guide.

In addition, you must set up AWS credentials to enable the AWS SAM CLI to make AWS service calls. For example, the AWS SAM CLI makes calls to Amazon S3 and AWS CloudFormation. For more information about setting up AWS credentials, see Setting up AWS credentials (p. 11).

**Step 3: Install Docker**

**Note**

Docker is only a prerequisite for testing your application locally and to build deployment packages using the --use-container flag. You may skip this section or install Docker at a later time if you do not plan to use these features initially.

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.

You must have Docker installed and working to be able to run serverless projects and functions locally with the AWS SAM CLI. 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 AWS SAM CLI (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 2.4.11
Homebrew/homebrew-core (git revision 54246b; last commit 2020-08-13)
Homebrew/homebrew-cask (git revision 4fd7ce; last commit 2020-08-14)
```

**Step 5: Install the AWS SAM CLI**

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

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

Verify the installation:

```
sam --version
```

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

```
SAM CLI, version 1.3.0
```

You're now ready to start development.

**Upgrading**

To upgrade the AWS SAM CLI, you still use Homebrew, but replace `install` with `upgrade` as follows:

```
brew upgrade 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. 12) – 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.

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:

```bash
$ 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.

### 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:

  ```bash
  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:
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 simple API backend. It consists of an API Gateway endpoint and a 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:

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

| #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 in the Installing the AWS SAM CLI (p. 3) for your OS. It assumes that you’ve done the following:

1. Created an AWS account.
2. Configured IAM permissions.
3. Installed Docker. Note: Docker is only a prerequisite for testing your application locally.
4. Installed Homebrew. Note: Homebrew is only a prerequisite for Linux and macOS.
5. Installed the AWS SAM CLI. Note: Make sure you have version 1.0.0 or later. You can check which version you have by executing the command `sam --version`.

**Step 1: Download a sample AWS SAM application**

**Command to run:**

```
sam init
```

Follow the on-screen prompts. For this tutorial we recommend you choose AWS Quick Start Templates, 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 you provided as the project name. The contents of the project directory are similar to the following (these contents are created when one of the Python runtimes and the Hello World Example are chosen):

```
sam-app/
#### README.md
#### events/
  #  #### event.json
#### hello_world/
  #  #### __init__.py
  #  #### app.py #Contains your AWS Lambda handler logic.
  #  #### requirements.txt #Contains any Python dependencies the application requires, used for sam build
#### template.yaml #Contains the AWS SAM template defining your application’s AWS resources.
#### tests/
  #### unit/
    #### __init__.py
    #### test_handler.py
```

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 (that is, the directory where the template.yaml file for the sample application is located; by default 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. You can just respond with `Enter` to accept the default options provided in the interactive experience.

**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 configuring authorization, see Controlling access to API Gateway APIs (p. 164).

Example output:

```
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>HelloWorldFunctionHelloWorldPermissionProd</td>
<td>AWS::Lambda::Permission</td>
</tr>
<tr>
<td>+ Add</td>
<td>HelloWorldFunction</td>
<td>AWS::Lambda::Function</td>
</tr>
<tr>
<td>+ Add</td>
<td>HelloWorldFunctionRole</td>
<td>AWS::IAM::Role</td>
</tr>
<tr>
<td>+ Add</td>
<td>ServerlessRestApiDeployment</td>
<td>AWS::ApiGateway::Deployment</td>
</tr>
<tr>
<td>+ Add</td>
<td>ServerlessRestApiProdStage</td>
<td>AWS::ApiGateway::Deployment</td>
</tr>
<tr>
<td>+ Add</td>
<td>ServerlessRestApi</td>
<td>AWS::ApiGateway::RestApi</td>
</tr>
<tr>
<td>+ Add</td>
<td>HelloWorldFunction</td>
<td>AWS::Lambda::Function</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>ResourceStatus</th>
<th>LogicalResourceId</th>
<th>ResourceType</th>
<th>ResourceStatusReason</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPDATE_IN_PROGRESS</td>
<td>HelloWorldFunctionRole</td>
<td>AWS::IAM::Role</td>
<td>-</td>
</tr>
<tr>
<td>UPDATE_COMPLETE</td>
<td>HelloWorldFunctionRole</td>
<td>AWS::IAM::Role</td>
<td>-</td>
</tr>
<tr>
<td>UPDATE_IN_PROGRESS</td>
<td>ServerlessRestApiDeployment</td>
<td>AWS::Lambda::Function</td>
<td>-</td>
</tr>
<tr>
<td>UPDATE_COMPLETE</td>
<td>ServerlessRestApiDeployment</td>
<td>AWS::Lambda::Function</td>
<td>-</td>
</tr>
<tr>
<td>CREATE_IN_PROGRESS</td>
<td>ServerlessRestApi</td>
<td>AWS::ApiGateway::RestApi</td>
<td>-</td>
</tr>
<tr>
<td>CREATE_IN_PROGRESS</td>
<td>ServerlessRestApiDeployment</td>
<td>AWS::ApiGateway::Deployment</td>
<td>-</td>
</tr>
<tr>
<td>CREATE_IN_PROGRESS</td>
<td>HelloWorldFunction</td>
<td>AWS::ApiGateway::RestApi</td>
<td>-</td>
</tr>
<tr>
<td>CREATE_COMPLETE</td>
<td>HelloWorldFunction</td>
<td>AWS::ApiGateway::RestApi</td>
<td>-</td>
</tr>
<tr>
<td>CREATE_IN_PROGRESS</td>
<td>ServerlessRestApiDeployment47fc2d5</td>
<td>AWS::Lambda::Permission</td>
<td>-</td>
</tr>
</tbody>
</table>
```

15
Step 3: Deploy your application to the AWS Cloud

CREATE_IN_PROGRESS      AWS::Lambda::Permission  sionProd
HelloWorldFunctionHelloWorldPermis -
CREATE_IN_PROGRESS      AWS::ApiGateway::Deployment sionProd
ServerlessRestApiDeployment47fc2d5 -
CREATE_COMPLETE         AWS::ApiGateway::Deployment f9d
ServerlessRestApiDeployment47fc2d5 -
CREATE_IN_PROGRESS      AWS::ApiGateway::Stage f9d
ServerlessRestApiProdStage -
CREATE_IN_PROGRESS      AWS::ApiGateway::Stage f9d
ServerlessRestApiProdStage -
CREATE_COMPLETE         AWS::ApiGateway::Stage
ServerlessRestApiProdStage -
CREATE_COMPLETE         AWS::Lambda::Permission
HelloWorldFunctionHelloWorldPermis -
UPDATE_COMPLETE_CLEANUP_IN_PROGRESS AWS::CloudFormation::Stack sionProd
Sam-app
S
UPDATE_COMPLETE         AWS::CloudFormation::Stack sam-app

Stack sam-app outputs:

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

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 you build with the `sam build` command, packages and uploads them to an Amazon S3 bucket created by AWS SAM CLI, and deploys the application using AWS CloudFormation. In the output of the deploy command you can see the changes being made to your AWS CloudFormation stack.

If your application created a HTTP endpoint, the Outputs generated by `sam deploy` 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/

You should see output like the following after successfully deploying your application:
Step 4: Testing your application locally (optional)

When you're developing your application, you might also 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
FETCHING lambci/lambda:python3.7 Docker container image......................................................................................................................................................................................
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 might 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......


START RequestId: 52fdfe07-2182-154f-163f-5f0f9a621d72 Version: $LATEST
END RequestId: 52fdfe07-2182-154f-163f-5f0f9a621d72

REPORT RequestId: 52fdfe07-2182-154f-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 locally in. The end result is the same output that you saw when you called your function in the AWS Cloud.

Making one-off invocations

Command to run:

```bash
sam local invoke "HelloWorldFunction" -e events/event.json
```

Example output:

```bash
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.....................................................................................................................


START RequestId: 52fdfe07-2182-154f-163f-5f0f9a621d72 Version: $LATEST
END RequestId: 52fdfe07-2182-154f-163f-5f0f9a621d72

REPORT RequestId: 52fdfe07-2182-154f-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's provided by the sample application.

Your initialized application came with a default aws-proxy event for API Gateway. A number of values are prepopulated 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 be substituted in to the request to simulate what you would expect from an actual request. This following is an example of generating your own input event and comparing the output with the default event.json object:

```bash
sam local generate-event apigateway aws-proxy --body "" --path "hello" --method GET > api-event.json
diff api-event.json event.json
```

Example output:
Troubleshooting

**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 this is not OK.

To fix this, you have the following options:

- Configure your application with authorization. For information configuring authorization, see Controlling access to API Gateway APIs (p. 164).
- 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.

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

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

```
Error: no such option: --app-template
```
AWS Serverless Application Model Developer Guide
Troubleshooting

This means that you are using an older version of the AWS SAM CLI that does not support the --apptemplate 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.

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.

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. 11).

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.
For instructions on installing Docker on your development host, go to Installing the AWS SAM
CLI (p. 3), choose the appropriate platform, and 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.

20


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 was deployed successfully by using the AWS CloudFormation console or AWS CLI, and that your `curl` command is correct.

Clean up

If you no longer need the AWS resources 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 created with this tutorial 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 stack you created).
4. Choose **Delete**.

When done, the status of the of the stack will change to **DELETE_COMPLETE**.

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

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

Verify deleted stack

For both methods of deleting the AWS CloudFormation stack, you can verify it was deleted by going to the https://console.aws.amazon.com/cloudformation, choosing **Stacks** in the left navigation pane, and choosing **Deleted** in the dropdown to the right of the search text box. You should see your stack name **sam-app** (or the name of the stack you created) 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 with AWS SAM.
2. Tested your application locally by 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 SAM GitHub repository. To access this repository, see AWS SAM example applications.
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. 23)
- AWS SAM resource and property reference (p. 29)
- Resource attributes (p. 151)
- Intrinsic functions (p. 152)
- Generated AWS CloudFormation resources (p. 152)
- API Gateway extensions (p. 158)

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. 25).

- 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. 29).

- 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. 209) 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. 25)**

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 the Globals 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.
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. 12).

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 the Globals 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:**

```markdown
Globals:
```
The Runtime for MyFunction is set to python3.6.

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:

```json
{
  "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. 29)
- AWS::Serverless::Application (p. 60)
- AWS::Serverless::Function (p. 63)
- AWS::Serverless::HttpApi (p. 114)
- AWS::Serverless::LayerVersion (p. 129)
- AWS::Serverless::SimpleTable (p. 132)
- AWS::Serverless::StateMachine (p. 135)

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. 29)` 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. 63)` resources defined in the template that do not refer to an `AWS::Serverless::Api (p. 29)` resource.

An `AWS::Serverless::Api (p. 29)` 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. 36)
  BinaryMediaTypes: List
  CacheClusterEnabled: Boolean
  CacheClusterSize: String
  CanarySetting: CanarySetting
  Cors: String | CorsConfiguration (p. 53)
  DefinitionBody: String
```
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. 164) in the AWS Serverless Application Model Developer Guide.

_Type:_ ApiAuth (p. 36)

_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 Amazon API Gateway Developer Guide.

Type: String | CorsConfiguration (p. 53)

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. 52)

_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.

**Domain**

Configures a custom domain for this API Gateway API.

_Type_: DomainConfiguration (p. 55)

_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. 59)

_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).

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
      Cors: "'www.example.com'"
      DefinitionBody: # Pull in an OpenApi definition from S3
                        'Fn::Transform':
                            Name: 'AWS::Include'
                            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

```yaml
Resources:
  ApiGatewayApi:
    Type: AWS::Serverless::Api
    Properties:
```
**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**

```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. 164) in the AWS Serverless Application Model Developer Guide.

**Syntax**

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

**YAML**

```yaml
AddDefaultAuthorizerToCorsPreflight: Boolean
ApiKeyRequired: Boolean
```
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. 164) in the AWS Serverless Application Model Developer Guide.

_Type_: CognitoAuthorizer (p. 41) | LambdaTokenAuthorizer (p. 47) | LambdaRequestAuthorizer (p. 43)

_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
**InvokeRole**

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

**Supported values:** CALLER_CREDENTIALS, NONE, IAM Role Arn.

**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. 49)

**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. 84). 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. 155). For general information about generated AWS CloudFormation resources, see Generated AWS CloudFormation resources (p. 152).

**Type:** ApiUsagePlan (p. 39)

**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: String
Description: String
Quota: QuotaSettings
Tags: List
Throttle: ThrottleSettings
UsagePlanName: String
```

**Properties**

**CreateUsagePlan**

Determine 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 theGlobals section of the AWS SAM template (p. 25).

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

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. 164) in the AWS Serverless Application Model Developer Guide.

Syntax

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

YAML

AuthorizationScopes: List
Identity: CognitoAuthorizationIdentity (p. 42)
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. 42)

**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:
        - scope1
        - scope2
      UserPoolArn:
        Fn::GetAtt:
          - MyCognitoUserPool
          - Arn
      Identity:
        Header: MyAuthorizationHeader
        ValidationExpression: myauthvalidationexpression
```

**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](https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-authorizer.html).

### Syntax

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

**YAML**

```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

Required: No

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

Examples

CognitoAuthIdentity

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. 164) in the AWS Serverless Application Model Developer Guide.

Syntax

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

YAML

AuthorizationScopes: List
**FunctionArn:** String
**FunctionInvokeRole:** String
**FunctionPayloadType:** String
**Identity:** LambdaRequestAuthorizationIdentity (p. 45)

<table>
<thead>
<tr>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AuthorizationScopes</strong></td>
</tr>
<tr>
<td>List of authorization scopes for this authorizer.</td>
</tr>
<tr>
<td><strong>Type:</strong> List</td>
</tr>
<tr>
<td><strong>Required:</strong> No</td>
</tr>
<tr>
<td><em>AWS CloudFormation compatibility:</em> This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.</td>
</tr>
<tr>
<td><strong>FunctionArn</strong></td>
</tr>
<tr>
<td>Specify the function arn of the Lambda function which provides authorization for the API.</td>
</tr>
<tr>
<td><strong>Type:</strong> String</td>
</tr>
<tr>
<td><strong>Required:</strong> Yes</td>
</tr>
<tr>
<td><em>AWS CloudFormation compatibility:</em> This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.</td>
</tr>
<tr>
<td><strong>FunctionInvokeRole</strong></td>
</tr>
<tr>
<td>Adds authorizer credentials to the OpenApi definition of the Lambda authorizer.</td>
</tr>
<tr>
<td><strong>Type:</strong> String</td>
</tr>
<tr>
<td><strong>Required:</strong> No</td>
</tr>
<tr>
<td><em>AWS CloudFormation compatibility:</em> This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.</td>
</tr>
<tr>
<td><strong>FunctionPayloadType</strong></td>
</tr>
<tr>
<td>This property can be used to define the type of Lambda Authorizer for an API.</td>
</tr>
<tr>
<td><strong>Supported values:</strong> TOKEN and REQUEST.</td>
</tr>
<tr>
<td><strong>Type:</strong> String</td>
</tr>
<tr>
<td><strong>Required:</strong> No</td>
</tr>
<tr>
<td><strong>Default:</strong> TOKEN</td>
</tr>
<tr>
<td><em>AWS CloudFormation compatibility:</em> This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.</td>
</tr>
<tr>
<td><strong>Identity</strong></td>
</tr>
<tr>
<td>This property can be used to specify an IdentitySource in an incoming request for an authorizer</td>
</tr>
<tr>
<td><strong>Type:</strong> LambdaRequestAuthorizationIdentity (p. 45)</td>
</tr>
<tr>
<td><strong>Required:</strong> No</td>
</tr>
</tbody>
</table>
**AWS CloudFormation compatibility**: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.

### Examples

**LambdaRequestAuth**

**YAML**

```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

```
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. 164) in the AWS Serverless Application Model Developer Guide.

Syntax

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

**YAML**

```
AuthorizationScopes: List
FunctionArn: String
FunctionInvokeRole: String
FunctionPayloadType: String
Identity: LambdaTokenAuthorizationIdentity (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.

Supported values: TOKEN and REQUEST.

*Type: String*
**Identity**

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

*Type:* LambdaTokenAuthorizationIdentity (p. 48)

*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](https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-authorizer-resource-policies.html) 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
```
Properties

AwsAccountBlacklist

The AWS accounts to block.

_Type_: List

_Restricted_: 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

_Restricted_: 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

_Restricted_: 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

_Restricted_: 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
```
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 Amazon 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**

```
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

*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
Route53: Route53Configuration (p. 57)
```

### Properties

**BasePath**

List of basepaths to be configured with the 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. SAM will create multiple `AWS::ApiGateway::BasePathMapping` resources, one per `BasePath` specified in this property.

CertificateArn

The reference to an AWS-managed certificate for use by the endpoint for this domain name. 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, the certificate must be created in the `us-east-1` 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. 155). For general information about generated AWS CloudFormation resources, see Generated AWS CloudFormation resources (p. 152).

Type: String

Required: Yes

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

EndpointConfiguration

Property to define the type of API Gateway endpoint to be mapped to the custom domain. The value of this property controls how the `CertificateArn` property gets mapped in AWS CloudFormation. See CertificateArn above.

Valid values are `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.

Route53

Property that adds Route53 configuration based on the values defined.

Type: `Route53Configuration` (p. 57)

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

## Examples

### DomainName

**DomainName example**

**YAML**

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

```yaml
Domain:
  DomainName: www.example.com
  CertificateArn: arn-example
```
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

```yaml
Type: String
VPCEndpointIds: 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

```yaml
EndpointConfiguration:
  Type: EDGE
  VPCEndpointIds:
    - vpce-123a123a
```
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. 60) 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. 62)
  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. 62) 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. 62)

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. 62) 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**

```
Type: AWS::Serverless::Application
Properties:
  Location:
    ApplicationId: 'arn:aws:serverlessrepo:us-east-1:012345678901:applications/my-application'
    SemanticVersion: 1.0.0
  Parameters:
    StringParameter: parameter-value
    IntegerParameter: 2
```

Normal-Application

Application from an S3 url

**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**

```
  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

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

AWS::Serverless::Function

Creates a Lambda function, IAM execution role, and event source mappings that trigger the function.

The AWS::Serverless::Function resource also supports the Metadata resource attribute, so you can instruct AWS SAM to build custom runtimes required by your application. For more information about building custom runtimes, see Building custom runtimes.

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
  CodeUri: String | FunctionCode
  DeadLetterQueue: Map | DeadLetterQueue
  DeploymentPreference: DeploymentPreference
  Description: String
  Environment: Environment
  EventInvokeConfig: EventInvokeConfiguration
  Events: EventSource
  FileSystemConfigs: List
  FunctionName: String
  Handler: String
  InlineCode: String
```
Properties

AssumeRolePolicyDocument

Add 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 ARN is provided for this function, this property does nothing.

AutoPublishAlias

Name of the Lambda alias. For more information about Lambda aliases, see AWS Lambda Function Aliases. For examples that use this property, see Deploying serverless applications gradually (p. 190).

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. 156). For general information about generated AWS CloudFormation resources, see Generated AWS CloudFormation resources (p. 152).

*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 if 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), 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 might occur when the deployment package stored in an 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, you must provide a unique value for AutoPublishCodeSha256 to trigger the gradual deployment successfully.

_Type_: String

_Required_: No

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

**CodeUri**

The Amazon S3 URI, local file path, or FunctionCode (p. 113) object of the function code.

If an Amazon S3 URI or FunctionCode (p. 113) object is provided, the Amazon S3 object referenced must be a valid Lambda deployment package.

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 code to be transformed properly.

**Note**: Either CodeUri or InlineCode is required.

_Type_: String | FunctionCode (p. 113)

_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 SNS topic or 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.

_Type_: Map | DeadLetterQueue (p. 71)

_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. 72)

_Required_: No

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

**See also**: See the Deploying serverless applications gradually (p. 190) documentation for more information about this property.
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. 75)

_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. 80)

_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 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
**Handler**

The function within your code that is called to begin execution.

*Type:* String

*Required:* Yes

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

**InlineCode**

The Lambda function code that is written directly in the template.

**Note:** Either `CodeUri` or `InlineCode` is required.

*Type:* String

*Required:* Conditional

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

**KmsKeyArn**

The Amazon Resource Name (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 should be used by this function. The order specified here is the order that 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 allocated per invocation of the function in MB.

*Type:* Integer

*Required:* No

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

**PermissionsBoundary**

The ARN of a permissions boundary to use for this function'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 function needs. They will be appended to the default role for this function.

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.

For more information about AWS managed IAM policies, see [AWS Managed Policies](<link>). For more information about AWS SAM policy templates, see [AWS SAM policy templates (p. 226)](<link>). For more information about inline policies, see [Inline Policies](<link>).

**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 only accepts JSON policy documents.

### ProvisionedConcurrencyConfig

The provisioned concurrency configuration of a function’s alias.

**Note:** ProvisionedConcurrencyConfig can only be specified 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](<link>) 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 Resource 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.

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 required by this function. For more information about building custom runtimes, see Building custom runtimes (p. 174).

Type: String

Required: Yes

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 IAM 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 IAM execution role.

Timeout

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

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.

Supported values: Active and PassThrough.

Type: String

Required: No
AWS CloudFormation compatibility: This property is similar to the TracingConfig property of an AWS::Lambda::Function resource. If Tracing is set to Active, then AWS SAM adds the arn:aws:iam::aws:policy/AWSXrayWriteOnlyAccess policy to the Lambda role.

**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 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 Amazon Resource Name (ARN) of the underlying Lambda function.

**Examples**

**Simple Function**

The following is a base case example of an AWS::Serverless::Function resource.

**YAML**

```yaml
Type: AWS::Serverless::Function
Properties:
  Handler: index.handler
  Runtime: python3.6
```
Function Properties Example

The following is an example of an AWS::Serverless::Function that uses InlineCode, Tracing, Policies, Layers, EFS, and an Api event source.

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
```

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

```
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.

*Supported 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**

```
DeadLetterQueue:
  Type: SNS
```

**DeploymentPreference**

Specifies the configurations to enable gradual Lambda deployments. For more information about configuring gradual Lambda deployments, see Deploying serverless applications gradually (p. 190).

*Note*: You must specify an `AutoPublishAlias` in your `AWS::Serverless::Function` (p. 63) 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. 74)
```
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. 74)

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. 190).

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**

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

**Properties**

**PostTraffic**

Lambda function that is run after traffic shifting.

Type: String
**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**

```
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**

```
DestinationConfig: EventInvokeDestinationConfiguration (p. 76)
MaximumEventAgeInSeconds: Integer
MaximumRetryAttempts: Integer
```

**Properties**

**DestinationConfig**

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

*Type: EventInvokeDestinationConfiguration (p. 76)*

*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**

```yaml
OnFailure: OnFailure (p. 77)
OnSuccess: OnSuccess (p. 78)
```

**Properties**

**OnFailure**

A destination for events that failed processing.
Type: **OnFailure** (p. 77)

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. 78)

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.

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:
    OnFailure:
      Type: Lambda
      Destination: !GetAtt 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
      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. 106) | SNS (p. 109) | Kinesis (p. 102) | DynamoDB (p. 93) | SQS (p. 112) | Api (p. 82) | Schedule (p. 108) | CloudWatchEvent (p. 90) | EventBridgeRule (p. 96) | CloudWatchLogs (p. 92) | IoTRule (p. 102) | AlexaSkill (p. 81) | Cognito (p. 93) | HttpApi (p. 98) | MSK (p. 105)
Type: String
```

Properties

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

```
Type: S3 (p. 106) | SNS (p. 109) | Kinesis (p. 102) | DynamoDB (p. 93) | SQS (p. 112) | Api (p. 82) | Schedule (p. 108) | CloudWatchEvent (p. 90) | EventBridgeRule (p. 96) | CloudWatchLogs (p. 92) | IoTRule (p. 102) | AlexaSkill (p. 81) | Cognito (p. 93) | HttpApi (p. 98) | MSK (p. 105)
Required: Yes
```

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

Type

The event type.
Supported values: S3, SNS, Kinesis, DynamoDB, SQS, Api, Schedule, CloudWatchEvent, CloudWatchLogs, IoTRule, AlexaSkill, Cognito, EventBridgeRule, HttpApi, MSK.

_Type_: String

_Required_: Yes

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

**Examples**

**API-Event**

Example of using an API Event

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

```
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 (p. 29) 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 (p. 29) 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. 84)
Method: String
Path: String
RequestModel: RequestModel (p. 88)
RequestParameters: String / RequestParameter (p. 89)
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. 84)*

*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. 29) resource.

Type: RequestModel (p. 88)

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. 89)

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. 29) resource defined in this template.

If you don't define this property, AWS SAM creates a default AWS::Serverless::Api (p. 29) 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 (p. 29) 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:
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

```yaml
ApiKeyRequired: Boolean
AuthorizationScopes: List
Authorizer: String
InvokeRole: 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 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
**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. 85)

**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**

```
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
```
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*
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
```

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

```yaml
  Model: String
  Required: Boolean
```
Properties

Model

Name of a model defined in the Models property of the AWS::Serverless::Api (p. 29).

_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. 96) 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

```
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**

```
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.

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
MaximumBatchingWindowInSeconds: Integer
MaximumRecordAgeInSeconds: Integer
MaximumRetryAttempts: Integer
ParallelizationFactor: Integer
StartingPosition: String
Stream: String
```

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 SQS queue or 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.

MaximumBatchingWindowIn Seconds
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 MaximumBatchingWindowIn Seconds 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.

Supported values: TRIM_HORIZON, LATEST.

Type: String

Required: Yes

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

Stream
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.

Examples

DynamoDB Event for Existing DynamoDB Table

DynamoDB Event 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.

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
EventBusName: String
Input: String
InputPath: String
```
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

EventBridgeRule

The following is an example of an `EventBridgeRule` event source type.

YAML

EBRule:
**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. 101)
Method: String
Path: String
PayloadFormatVersion: String
RouteSettings: RouteSettings
TimeoutInMillis: Integer
```

**Properties**

**ApiId**

Identifier of an `AWS::Serverless::HttpApi (p. 114)` resource defined in this template.

If not defined, a default `AWS::Serverless::HttpApi (p. 114)` 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. 114)` 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. 101)`

**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.

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:
      Auth:
        Authorizer: OpenIdAuth
        AuthorizationScopes:
          - scope1
```
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**

```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 in the AWS IoT Developer Guide.

*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**

```yaml
IoTRule:
  Type: IoTRule
  Properties:
    Sql: SELECT * FROM 'topic/test'
```

Kinesis

The object describing a Kinesis event source type.

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
BatchSize: Integer  
BisectBatchOnFunctionError: Boolean  
DestinationConfig: DestinationConfig  
Enabled: Boolean  
MaximumBatchingWindowInSeconds: Integer  
MaximumRecordAgeInSeconds: Integer  
MaximumRetryAttempts: Integer  
ParallelizationFactor: Integer  
StartingPosition: String  
Stream: String
```

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 SQS queue or 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
AWS CloudFormation compatibility: This property is passed directly to the `Enabled` property of an `AWS::Lambda::EventSourceMapping` resource.

**MaximumBatchingWindowInSeconds**

The maximum amount of time to gather records before invoking the function, in seconds.

*Type:* Integer

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

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

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

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.

Supported values: `TRIM_HORIZON`, `LATEST`, `AT_TIMESTAMP`.

*Type:* String

AWS CloudFormation compatibility: This property is passed directly to the `StartingPosition` property of an `AWS::Lambda::EventSourceMapping` resource.

**Stream**

The 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.

Examples

Kinesis Event Source

Kinesis Event Source

YAML

```yaml
Events:
  KinesisEvent:
    Type: Kinesis
    Properties:
      StartingPosition: TRIM_HORIZON
      BatchSize: 10
      Enabled: false
```

MSK

The object describing an MSK event source type.

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.

Supported values: TRIM_HORIZON, LATEST, AT_TIMESTAMP.

Type: String

Required: Yes

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

Stream

The 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:
        - MyTopic
```

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

```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

```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](https://docs.aws.amazon.com/AmazonS3/latest/API/s3-events.html) 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
```
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.

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
Description: String
Enabled: Boolean
Input: String
Name: String
Schedule: String
```

Properties

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.

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
```

**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. 111)
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. 111)

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

Properties:
  SqsSubscription: True
  FilterPolicy:
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:
  Fn::GetAtt: MyCustomQueue.Arn
QueueUrl:
  Ref: MyCustomQueue
BatchSize: 5
```

SQS

The object describing an SQS event source type.

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
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: 10

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.

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

SQS Event

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

```
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 API Gateway HTTP API, which enables you to create RESTful APIs with lower latency and lower costs than REST APIs. For more information about HTTP APIs see HTTP API 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
Type: AWS::Serverless::HttpApi
Properties:
  AccessLogSettings: AccessLogSettings
  Auth: HttpApiAuth (p. 120)
  CorsConfiguration: String | HttpApiCorsConfiguration (p. 123)
  DefaultRouteSettings: RouteSettings
  DefinitionBody: String
  DefinitionUri: String | HttpApiDefinition (p. 125)
  Domain: HttpApiDomainConfiguration (p. 126)
  FailOnWarnings: Boolean
  RouteSettings: RouteSettings
  StageName: String
  StageVariables: Json
  Tags: Map
```

Properties

AccessLogSettings

Settings for logging access 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

Configure authorization to control access to your API Gateway API.

For more information about configuring access see JWT Authorizers in the API Gateway Developer Guide.

*Type: HttpApiAuth (p. 120)

*Required: No

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

CorsConfiguration

Manage Cross-origin resource sharing (CORS) for all your HTTP 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 inline OpenApi defined with DefinitionBody.

For more information about CORS, see Configuring CORS for an HTTP API in the Amazon 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.

Note: If this property is set to True then all origins are allowed.
Type: String | HttpApiCorsConfiguration (p. 123)

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

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::ApiGatewayV2::Api resource. If certain properties are provided, content may be inserted or modified into the DefinitionBody before being passed to CloudFormation. Properties include Auth and an EventSource of type HttpApi 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 | HttpApiDefinition (p. 125)

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.

Domain

Configures a custom domain for this API Gateway API.

Type: HttpApiDomainConfiguration (p. 126)

Required: No

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

Specifies whether to rollback the 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::ApiGatewayV2::Api resource.

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.

Type: RouteSettings

Required: No

AWS CloudFormation compatibility: This property is passed directly to the RouteSettings property of an AWS::ApiGatewayV2::Stage resource.

StageName

The name of the API stage. If a name is not given, SAM will use the $default stage from Api Gateway.

Type: String

Required: No

Default: $default

AWS CloudFormation compatibility: This property is passed directly to the StageName property of an AWS::ApiGatewayV2::Stage resource.

StageVariables

A map that defines the stage variables for a Stage. Variable names can have alphanumeric and underscore characters, and 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 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. NOTE: Tags requires AWS SAM to modify your OpenAPI definition. So, it works only if inline OpenApi is defined with DefinitionBody.

Type: Map

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent.
Additional Notes: Because Tags requires AWS SAM to modify your OpenAPI definition, they will only be added if the DefinitionBody property is specified—no tags will be added if the DefinitionUri property is provided. AWS SAM automatically adds a httpapi:createdBy:SAM tag. Tags will also be added to AWS::ApiGatewayV2::Stage and AWS::ApiGatewayV2::DomainName (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, such as a1bcdef2gh.

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

Examples

Simple Http Api

Bare minimum needed to set up an HttpApi endpoint backed by a Lambda function. This uses the default HTTP API that 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
```

Http Api with Auth

Example of how to set up authorization on API endpoints.

YAML

```
Properties:
  FailOnWarnings: true
  Auth:
    DefaultAuthorizer: OAuth2
    Authorizers:
      OAuth2:
        AuthorizationScopes:
        - scope4
        JwtConfiguration:
          issuer: "https://www.example.com/v1/connect/oauth2"
          audience:
            - MyApi
```
Http Api with OpenApi Document

Shows how to add OpenApi to the document.

Note that SAM will fill in any missing lambda integrations for HttpApi events that reference this API. SAM will also add any missing paths that HttpApi events reference.

YAML

```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
```

Http Api with Configuration Settings

Shows how to add 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:
  Type: AWS::Logs::LogGroup

Outputs:
  HttpApiUrl:
    Description: URL of your API endpoint
    Value:
      Fn::Sub: 'https://${HttpApi}.execute-api.${AWS::Region}.${AWS::URLSuffix}/
      ${StageName}/'
  HttpApiId:
    Description: Api id of HttpApi
    Value:
      Ref: HttpApi

HttpApiAuth

Configure authorization to control access to your API Gateway API.

For more information about configuring access see 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**

```
Authorsizers: OAuth2Authorizer (p. 122)
DefaultAuthorizer: String
```

Properties

**Authorsizers**

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

- **Type**: OAuth2Authorizer (p. 122)
- **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.

- **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

**OAuth2 Authorizer**

**OAuth2 Authorizer Example**

**YAML**

```
Auth:
  Authorizers:
    OAuth2Authorizer:
      AuthorizationScopes:
        - scope1
        - scope2
      JwtConfiguration:
        issuer: "https://www.example.com/v1/connect/oauth2"
        audience:
          - MyApi
        IdentitySource: "$request.querystring.param"
```
DefaultAuthorizer: OAuth2Authorizer

**OAuth2Authorizer**

Definition for an OAuth 2.0 authorizer.

For more information, see the [API Gateway documentation](https://docs.aws.amazon.com/apigateway/latest/developerguide/authorizers-oauth2.html).

**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**

JSON Web Token (JWT) configuration for this authorizer.

This is passed through to the `jwtConfiguration` section of a `x-amazon-apigateway-authorizer` in the `securitySchemes` section of an OpenAPI document.

*Type*: Map

*Required*: No

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

**Examples**

**OAuth2 Authorizer**

**OAuth2 Authorizer Example**
YAML

Auth:
  Authorizers:
    OAuth2Authorizer:
      AuthorizationScopes:
        - scope1
      JwtConfiguration:
        issuer: "https://www.example.com/v1/connect/oauth2"
        audience:
          - MyApi
        IdentitySource: "$request.querystring.param"
    DefaultAuthorizer: OAuth2Authorizer

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 Amazon 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**

```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::ApiBodyS3Location` 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::ApiBodyS3Location` 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::ApiBodyS3Location` data type.

Examples

**Definition Uri example**

**API Definition example**

**YAML**

```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

```yaml
BaseUrl: List
CertificateArn: String
DomainName: String
EndpointConfiguration: String
Route53: Route53Configuration (p. 127)
```

Properties

BaseUrl

List of basepaths to be configured with the API Gateway Domain Name.

Type: List

Required: No

Default: /

AWS CloudFormation compatibility: This property is similar to the `BaseUrl` property of an `AWS::ApiGatewayV2::ApiMapping` resource. SAM will create multiple `AWS::ApiGatewayV2::ApiMapping` resources, one per `BaseUrl` specified in this property.

Additional Notes: SAM will create multiple `AWS::ApiGatewayV2::ApiMapping` resources, one per `BaseUrl` specified in this property.

CertificateArn

The reference to an AWS-managed certificate for use by the endpoint for this domain name. 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. 158). For general information about generated AWS CloudFormation resources, see Generated AWS CloudFormation resources (p. 152).
**Type:** String

**Required:** Yes

*AWS CloudFormation compatibility:* This property is passed directly to the `DomainName` property of an `AWS::ApiGateway2::DomainName` resource.

**EndpointConfiguration**

Property to define the type of API Gateway endpoint to be mapped to the custom domain. The value of this property controls how the `CertificateArn` property gets mapped in AWS CloudFormation. See `CertificateArn` above.

Valid value for HttpApis is only `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.

**Route53**

Property that adds Route53 configuration based on the values defined.

**Type:** `Route53Configuration (p. 127)`

**Required:** No

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

**Examples**

**DomainName**

DomainName example

**YAML**

```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
**AWS::Serverless::LayerVersion**

Creates a Lambda LayerVersion that contains library or runtime code needed by a Lambda Function.

The **AWS::Serverless::LayerVersion** 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. 173).

**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. 129).

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**

```
Type: AWS::Serverless::LayerVersion
Properties:
  CompatibleRuntimes: List
```
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. 131)
 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.

Supported values: Retain and Delete.

Type: String

Required: No

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

Additional Notes: When you specify Retain, AWS SAM adds a Resource Attribute of DeletionPolicy: Retain to the transformed AWS::Lambda::LayerVersion resource.

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

<table>
<thead>
<tr>
<th>Properties:</th>
</tr>
</thead>
<tbody>
<tr>
<td>LayerName: MyLayer</td>
</tr>
<tr>
<td>Description: Layer description</td>
</tr>
<tr>
<td>ContentUri: 's3://my-bucket/my-layer.zip'</td>
</tr>
<tr>
<td>CompatibleRuntimes:</td>
</tr>
<tr>
<td>- nodejs10.x</td>
</tr>
<tr>
<td>- nodejs12.x</td>
</tr>
<tr>
<td>LicenseInfo: 'Available under the MIT-0 license.'</td>
</tr>
<tr>
<td>RetentionPolicy: Retain</td>
</tr>
</tbody>
</table>

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**

```yaml
Type: AWS::Serverless::SimpleTable
Properties:
  PrimaryKey: PrimaryKeyObject (p. 134)
  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. 134)

*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

*Required:* No

*AWS CloudFormation compatibility:* This property is passed directly to the `ProvisionedThroughput` property of an `AWS::DynamoDB::Table` resource.

**SSESpecification**

Specifies the settings to enable server-side encryption.

*Type:* SSESpecification

*Required:* No

*AWS CloudFormation compatibility:* This property is passed directly to the `SSESpecification` property of an `AWS::DynamoDB::Table` resource.

**TableName**

Name for the DynamoDB Table.
**Type:** String

**Required:** No

*AWS CloudFormation compatibility:* This property is passed directly to the `TableName` property of an `AWS::DynamoDB::Table` resource.

**Tags**

A map (string to string) that specifies the tags to be added to this SimpleTable. 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::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**

```
Properties:
  TableName: my-table
  Tags:
    Department: Engineering
    AppType: Serverless
```

**PrimaryKeyObject**

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**

```
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.

_Supported 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

PrimaryKey

Primary key example.

YAML

```
Properties:
  PrimaryKey:
    Name: MyPrimaryKey
    Type: String
```

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](https://docs.aws.amazon.com/stepfunctions/latest/dg/).

**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

```
Type: AWS::Serverless::StateMachine
Properties:
  Definition: Map
  DefinitionSubstitutions: Map
  DefinitionUri: String | S3Location
  Events: EventSource (p. 139)
  Logging: LoggingConfiguration
  Name: 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. 139).

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.

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
### AWS::Serverless::StateMachine

**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. 139)

**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.

#### 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 Amazon Resource Name (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 IAM 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 \texttt{Fn::GetAtt}, see \texttt{Fn::GetAtt} in the \textit{AWS CloudFormation User Guide}.

Name

Returns the name of the state machine, such as \texttt{HelloWorld-StateMachine}.

Examples

State Machine Definition File

The following is an example of a state machine defined with a definition file. The \texttt{my\_state\_machine.asl.json} file must be written in the \textit{Amazon States Language}.

In this example, the \texttt{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.Arn
      MyDDBTable: !Ref TransactionTable
```

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**

```yaml
Properties: Schedule (p. 142) | CloudWatchEvent (p. 149) | EventBridgeRule (p. 140) | Api (p. 143)
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. 142) | CloudWatchEvent (p. 149) | EventBridgeRule (p. 140) | Api (p. 143)*

*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
```

**EventBridgeRule**

The object describing an EventBridgeRule event source type.
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
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

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
```

Schedule

The object describing a Schedule event source type.

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
Description: String
Enabled: Boolean
Input: String
Name: String
Schedule: String
```

Properties

**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.

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

```yaml
CWSchedule:
  Type: Schedule
  Properties:
    Schedule: 'rate(1 minute)'
    Name: TestSchedule
    Description: test schedule
    Enabled: False
```

Api

The object describing an Api event source type. If an AWS::Serverless::Api (p. 29) 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. 145)
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. 145)*

*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. 29)` resource that is defined in this template.

If you don't define this property, AWS SAM creates a default `AWS::Serverless::Api (p. 29)` 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. 29)` 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
```

ApiStateMachineAuth

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. 146)

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

```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
```
### 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.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Required</th>
<th>AWS CloudFormation compatibility</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AwsAccountBlacklist</code></td>
<td>List</td>
<td>No</td>
<td>Unique to AWS SAM</td>
</tr>
<tr>
<td><code>AwsAccountWhitelist</code></td>
<td>List</td>
<td>No</td>
<td>Unique to AWS SAM</td>
</tr>
<tr>
<td><code>CustomStatements</code></td>
<td>List</td>
<td>No</td>
<td>Unique to AWS SAM</td>
</tr>
<tr>
<td><code>IntrinsicVpcBlacklist</code></td>
<td>List</td>
<td>No</td>
<td>Unique to AWS SAM</td>
</tr>
<tr>
<td><code>IntrinsicVpcWhitelist</code></td>
<td>List</td>
<td>No</td>
<td>Unique to AWS SAM</td>
</tr>
</tbody>
</table>
**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
```

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. 140) 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 in the Amazon EventBridge User Guide

*Type: EventPattern*
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. 29)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AWS::Serverless::Application (p. 60)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AWS::Serverless::Function (p. 63)</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>(p. 174)*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AWS::Serverless::HttpApi (p. 114)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AWS::Serverless::LayerVersion (p. 129)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>(p. 173)**</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AWS::Serverless::SimpleTable (p. 132)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AWS::Serverless::StateMachine (p. 135)</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. 174).
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. 152)
- Generated AWS CloudFormation resource reference (p. 153)
- AWS CloudFormation resources generated when AWS::Serverless::Api is specified (p. 154)
- AWS CloudFormation resources generated when AWS::Serverless::Function is specified (p. 155)
- AWS CloudFormation resources generated when AWS::Serverless::HttpApi is specified (p. 157)

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. 154)</td>
<td>- DomainName property is specified (p. 155)</td>
</tr>
<tr>
<td></td>
<td>- UsagePlan property is specified (p. 155)</td>
</tr>
<tr>
<td>AWS::Serverless::Function (p. 155)</td>
<td>- AutoPublishAlias property is specified (p. 156)</td>
</tr>
<tr>
<td></td>
<td>- Role property is not specified (p. 156)</td>
</tr>
<tr>
<td></td>
<td>- OnSuccess (or OnFailure) property is specified for Amazon SNS events (p. 156)</td>
</tr>
<tr>
<td></td>
<td>- OnSuccess (or OnFailure) property is specified for Amazon SQS events (p. 157)</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. 155)
- UsagePlan property is specified (p. 155)
**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. 156)
- Role property is not specified (p. 156)
- OnSuccess (or OnFailure) property is specified for Amazon SNS events (p. 156)
- OnSuccess (or OnFailure) property is specified for Amazon SQS events (p. 157)

**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<alias-name>`

*<alias-name>* 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. 158)
- StageName property is not specified (p. 158)
- DomainName property is specified (p. 158)
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>
</tbody>
</table>
### API Gateway extensions

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<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 how to validate your AWS SAM template, and how to build your application with dependencies. It also contains topics for how to use AWS SAM for certain use cases like working with Lambda layers, using nested applications, and controlling access to API Gateway APIs.

Topics
- Validating AWS SAM template files (p. 160)
- Working with layers (p. 160)
- Using nested applications (p. 162)
- Controlling access to API Gateway APIs (p. 164)
- Orchestrating AWS resources with AWS Step Functions (p. 170)

Validating AWS SAM template files

Validate your templates with `sam validate` (p. 223). 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

The AWS SAM CLI supports applications that include layers. For more information about layers, see AWS Lambda layers.

The 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>

For more information about including layers in your application, see AWS::Serverless::Function (p. 63).

When you invoke your function using one of the sam local CLI subcommands, the layers package of your function is downloaded and cached on your local host. See the following chart for default cache directory locations. 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:**

```
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:

```
```

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:

```
myLayer-1-926eeb5ff1-mySecondLayer-1-6bc1022bdf = 2dd7ac5fffb30d515926aef
```

Then combine this value with the function's runtime, with a delimiter of '-':

```
2dd7ac5fffb30d515926aef-Python3.7 = 2dd7ac5fffb30d515926aef-Python3.7
```
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 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 applications, see Application Deployment Permissions and Publishing Applications in the AWS Serverless Application Repository Developer Guide.

### 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
```
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. 189).

**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

You can use AWS SAM to control who can access your API Gateway APIs by enabling authorization within your AWS SAM template.

AWS SAM supports several mechanisms for controlling access to your API Gateway APIs:

- **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 is provided by the client application. The Lambda function returns a policy document that specifies the operations that the caller is authorized to perform, if any. For more information about Lambda authorizers, see Use API Gateway Lambda Authorizers in the API Gateway Developer Guide. For examples of Lambda authorizers, see Example: Defining Lambda token authorizers (p. 165) and Example: Defining Lambda request authorizers (p. 166) later in this topic.

- **Amazon Cognito user pools.** Amazon Cognito user pools are user directories in Amazon Cognito. A client of your API must first sign a user into the user pool, and obtain an identity or access token for the user. Then your API is called with one of the returned tokens. The API call succeeds only if the required token is valid. For more information about Amazon Cognito user pools, see Control Access to REST API Using Amazon Cognito User Pools as Authorizer in the API Gateway Developer Guide. For an example of Amazon Cognito user pools, see Example: Defining Amazon Cognito user pools (p. 167) later in this topic.

- **IAM permissions.** You can control who can invoke your API using IAM permissions. Users calling your API must be authenticated with IAM credentials. Calls to your API only succeed 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 is assumed by the user. For more information about IAM permissions, see Control Access to an API with IAM Permissions in the API Gateway Developer Guide. For an example of IAM permissions, see Example: Defining IAM permissions (p. 168) later in this topic.

- **API keys.** API keys are alphanumeric string values that you distribute to application developer customers to grant access to your API. For more information about API keys, see Create and Use Usage Plans with API Keys in the API Gateway Developer Guide. For an example of API keys, see Example: Defining API keys (p. 168) later in this topic.

- **Resource policies.** Resource policies are JSON policy documents that you can attach to an API Gateway API to control whether a specified principal (typically an IAM user or role) can invoke the API. For more information about resource policies, see Control Access to an API with Amazon API Gateway Resource Policies in the API Gateway Developer Guide. For an example of resource policies, see Example: Defining resource policies (p. 169) later in this topic.

In addition, you can use AWS SAM to customize the content of some API Gateway error responses. For more information about customizing API Gateway error responses, see Set Up Gateway Responses to Customize Error Responses. For an example of customized responses, see Example: Defining customized responses (p. 169) later in this topic.

Choosing a mechanism to control access

The mechanism that you choose to control access to your API Gateway APIs depends on a few factors. For example, if you have a greenfield project that doesn't have either authorization or access control set up yet, then Amazon Cognito user pools might be your best option. This is because when you set up user pools, you also set up both authentication and access control automatically.

However, if your application already has authentication set up, then using Lambda authorizers might be the 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.

After you've decided which mechanism to use, see the corresponding section in this topic to see how to use AWS SAM to configure your application to use that mechanism.

Example: Defining Lambda token authorizers

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. 36) 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:
```
Example: Defining Lambda request authorizers

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. 36) data type.

The following is an example AWS SAM template section for a Lambda Request authorizer:

```yaml
Resources:
  MyApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      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
```

For more information about API Gateway Lambda authorizers, see Use API Gateway Lambda Authorizers in the *API Gateway Developer Guide*.
Example: Defining Amazon Cognito user pools

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. 36) 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.
Example: Defining IAM permissions

You can control access to your APIs by defining IAM permissions within your AWS SAM template. To do this, you use the ApiAuth (p. 36) 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 Using IAM Permissions in the API Gateway Developer Guide.

Example: Defining API keys

You can control access to your APIs by requiring API keys within your AWS SAM template. To do this, you use the ApiAuth (p. 36) 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 Create and Use Usage Plans with API Keys in the API Gateway Developer Guide.

Example: Defining resource policies

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. 36) 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 Control Access to an API with Amazon API Gateway Resource Policies in the API Gateway Developer Guide.

Example: Defining customized responses

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:

```
Resources:
  MyApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      GatewayResponses:
        DEFAULT_4xx:
          ResponseParameters:
            Headers:
```
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

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:
    GetResource:
      Type: Api
      Properties:
        Path: /error
        Method: get
        RestApiId: !Ref MyApi
```
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. 135)
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` 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. 172)
- Building layers (p. 173)
- Building custom runtimes (p. 174)

Building applications

To build your serverless application, use the `sam build` command.

`sam build` also gathers the build artifacts of your application’s dependencies, and places them in the proper format and location 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), or by using the `Layers` property of a function resource. The `Layers` property contains a list of Lambda layer resources that the function depends on.

If your AWS Lambda function depends on packages that have natively compiled programs, use the `--use-container` flag. The `--use-container` flag locally compiles your functions in a Docker container that functions like a Lambda environment, so they are in the right format when you deploy them to the AWS Cloud.

**Examples**

The following `sam build` commands build applications.

```bash
# 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
```
Building layers

To build layers that you have declared in your AWS Serverless Application Model (AWS SAM) template file, 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 following is an example Metadata resource attribute section.

```yaml
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. 130) property of the `AWS::Serverless::LayerVersion` resource type.

When you include the Metadata resource attribute section, you can use the `sam build` (p. 207) 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.

```yaml
Resources:
MyLayer:
    Type: AWS::Serverless::LayerVersion
    Properties:
        ContentUri: my_layer
        CompatibleRuntimes:
            - python3.6
    Metadata:
        BuildMethod: python3.6  # Required to have AWS SAM build this layer
```

**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:

MyLayer:
  Type: AWS::Serverless::LayerVersion
  Properties:
    ContentUri: my_layer
    CompatibleRuntimes:
      - python3.8
  Metadata:
    BuildMethod: makefile

The following makefile contains the build target and commands that will be executed.

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
sam build

Building custom runtimes

You can use the sam build (p. 207) 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.

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:
Examples

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.

```
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.

```
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.

Topics
- Invoking Functions Locally (p. 176)
- Running API Gateway locally (p. 178)
- Integrating with automated tests (p. 180)
- Generating sample event payloads (p. 182)
- Step-through debugging Lambda functions locally (p. 182)
- Passing additional runtime debug arguments (p. 188)

Invoking Functions Locally

You can invoke your function locally by using the `sam local invoke` 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:

```
# 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
```

This animation shows invoking a Lambda function locally using Microsoft Visual Studio Code:
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. Structure the file as follows:


```json
{
    "MyFunction1": {
        "TABLE_NAME": "localtable",
        "BUCKET_NAME": "testBucket"
    },
    "MyFunction2": {
        "TABLE_NAME": "localtable",
        "STAGE": "dev"
    }
}
```

For example, if you save this content in a file named `env.json`, then 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. 160) for more information about how to debug layers issues on your local host.

**Running API Gateway locally**

Use the `sam local start-api (p. 216)` 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.

**Example:**

```
sam local start-api
```

AWS SAM automatically finds any functions within your AWS SAM template that have `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 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

By default, AWS SAM uses **Proxy Integration** and expects the response from your Lambda function to include one or more of the following: statusCode, headers, or body.

For example:

```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"
  });
}
```

For examples in other AWS Lambda languages, see **Proxy Integration**.

**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. Structure the file as follows:

```json
{
  "MyFunction1": {
    "TABLE_NAME": "localtable",
    "BUCKET_NAME": "testBucket"
  },
  "MyFunction2": {
    "TABLE_NAME": "localtable",
    "STAGE": "dev"
  }
}
```

For example, if you save this content in a file named `env.json`, then 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. 160)** 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:

   ```
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
   # locally if it is configured to do so
   response = lambda_client.invoke(FunctionName="HelloWorldFunction")

   # Verify the response
   assert response == "Hello World"
   ```
You can use this code to test deployed Lambda functions by setting `running_locally` to `False`. This sets up the AWS SDK to connect to AWS Lambda in the AWS Cloud.

### 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:

- AWS Toolkit for JetBrains
- AWS Toolkit for PyCharm
- AWS Toolkit for IntelliJ
- AWS Toolkit for Visual Studio Code
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. 183)
- Step-through debugging Python functions locally (p. 185)
- Step-through debugging Golang functions locally (p. 187)

**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": [
```
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:

```
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:

```
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. 188) `-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:

```bash
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?.

You can deploy your application by using AWS SAM command line interface (CLI) commands. You can also use other AWS services that integrate with AWS SAM to automate your deployments.

The standard input to deploying serverless applications is the build artifacts created using the `sam build` command. For more information about the `sam build` command, see Building serverless applications.

Deploying using the AWS SAM CLI

After you develop and test your serverless application locally, you can deploy your application by 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 Amazon Simple Storage Service (Amazon S3), and deploys your application to the AWS Cloud.

Example:

```bash
# 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 have been published by others. For more information, see What Is the AWS Serverless Application Repository?

Automating deployments

You can use AWS SAM with a number of other AWS services to automate the deployment process of your serverless application.

- **CodeBuild**: You use CodeBuild to build, locally test, and package your serverless application. For more information, see What Is CodeBuild?.
- **CodeDeploy**: You use CodeDeploy to gradually deploy updates to your serverless applications. For more information, see Deploying serverless applications gradually (p. 190).
- **CodePipeline**: You use CodePipeline to model, visualize, and automate the steps that are required to release your serverless application. For more information, see What Is CodePipeline?
Troubleshooting

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 this is not OK.

To fix this, you have the following options:

- Configure your application with authorization. For information configuring authorization, see Controlling access to API Gateway APIs (p. 164).
- 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:
```
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/versioning.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>
</tbody>
</table>
Deploying gradually

### Deployment Preference Type

<table>
<thead>
<tr>
<th>Linear10PercentEvery2Minutes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear10PercentEvery3Minutes</td>
</tr>
<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. 193)

Working with logs

To simplify troubleshooting, the AWS SAM CLI has a command called `sam logs` (p. 220). 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

You can use the AWS SAM CLI to publish your application to the AWS Serverless Application Repository to make it available for others to find and deploy.

The application that you want to publish must be one that you've defined using AWS SAM. You also need to have tested it locally and/or in the AWS Cloud. The application's deployment package and AWS SAM template are the inputs to the following procedure steps.

The following instructions either create a new application, create a new version of an existing application, or update the metadata of an existing application. This 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 that's used to publish applications, see AWS SAM template Metadata section properties (p. 198).

Prerequisites

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

- A valid AWS account with an IAM user that has administrator permissions. See Set Up an AWS Account.
- Version 1.16.77 or later of the AWS CLI installed. See Installing the AWS Command Line Interface. If you have the AWS CLI installed, you can get the version by running the following command:

  ```
  aws --version
  ```

- The AWS SAM CLI (command line interface) installed. See Installing the AWS SAM CLI. You can determine whether the AWS SAM CLI is installed by running the following command:

  ```
  sam --version
  ```

- A valid AWS Serverless Application Model (AWS SAM) template.
- Your application code and dependencies referenced by the AWS SAM template.
- A semantic version for your application (required 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 to share your application publicly).
- A valid Amazon S3 bucket policy that grants the service read permissions for artifacts uploaded to Amazon S3 when you packaged your application. To do this, follow these steps:

  1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
  2. Choose the Amazon S3 bucket that you used to package your application.
  3. Choose the Permissions tab.
  4. Choose the Bucket Policy button.
  5. Paste the following policy statement into the Bucket policy editor. Make sure to substitute your bucket name in the Resource property value.
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 properties of the Metadata section in the AWS SAM template, see AWS SAM template Metadata section properties (p. 198).

Step 2: Package the application

Execute the following AWS SAM CLI command:

```
sam package \
  --template-file template.yaml \
```

Step 3: Publish the application

Execute the following AWS SAM CLI command:

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

The output of the `sam publish` command includes a link to the AWS Serverless Application Repository directly to your application. You can also go to the AWS Serverless Application Repository landing page directly and search for your application.

Your application is set to private by default, so it isn’t visible to other AWS accounts. In order to share your application with others, you must either make it public or grant permission to a specific list of AWS accounts. For information on sharing your application by using the AWS CLI, see Using Resource-based Policies for the AWS Serverless Application Repository. For information on sharing your application using the console, see Sharing an Application Through the Console.

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, you publish the application using the same process as above. You provide the same application name that you originally published it with, but with a new SemanticVersion value. You also provide the application name and SemanticVersion number in the Metadata section of the AWS SAM template file.

For example, if you published an application with the name `SampleApp` and SemanticVersion `1.0.0`, to update that application, the AWS SAM template must have application name `SampleApp`, and the SemanticVersion can be `1.0.1` (or anything different from `1.0.0`).

Additional topics

- AWS SAM template Metadata section properties (p. 198)
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>
</table>
| Name             | String | TRUE     | The name of the application. Minimum length=1. Maximum length=140. Pattern: 
|                  |        |          | "[a-zA-Z0-9\-]+";                                                          |
| Description      | String | TRUE     | The description of the application. Minimum length=1. Maximum length=256.   |
| Author           | String | TRUE     | The name of the author publishing the application. Minimum length=1. Maximum length=127. Pattern: 
<p>|                  |        |          | &quot;^<a href="(%5Ba-z0-9%5D%7C-(?!-))*%5Ba-z0-9%5D">a-z0-9</a>?$&quot;;                                 |
| SpdxLicenseId    | String | FALSE    | A valid license identifier. To view the list of valid license identifiers, see SPDX License List on the Software Package Data Exchange (SPDX) website. |
| LicenseUrl       | String | FALSE    | The reference to a local license file, or an Amazon S3 link to a license file, that matches the spdxLicenseId value of your application. An AWS SAM template file that hasn't been packaged using the sam package command can have a reference to a local file for this property. However, for an application to be published using the sam publish command, this property must be a reference to an Amazon S3 bucket. Maximum size: 5 MB. You must provide a value for this property in order to make your application public. Note that you cannot 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. |</p>
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 Semantic Versioning 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. 201)
- Process Amazon S3 events (p. 203)

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.

```
sam init \
--location gh:aws-samples/cookiecutter-aws-sam-dynamodb-python \
--no-input
```

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:

```
sam local generate-event dynamodb update | sam local invoke --event - ReadDynamoDBEvent
```
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.

```bash
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):

- `template.yaml` – Defines three AWS resources that the Amazon S3 application needs: a Lambda function, an Amazon S3 bucket, and a DynamoDB table. The template also defines the mappings and permissions between these resources.
- `src/` directory – Contains the Amazon S3 application code.
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.

```
aws s3 mb s3://bucketname
```

2. Create the deployment package by running the following package CLI command at the command prompt.

```
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.

```
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. 23).

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.

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. 23)
• AWS SAM CLI command reference (p. 207)
• AWS SAM policy templates (p. 226)
AWS SAM CLI command reference

This section is the reference for the AWS SAM CLI commands.

Topics
- sam build (p. 207)
- sam deploy (p. 209)
- sam init (p. 211)
- sam local generate-event (p. 213)
- sam local invoke (p. 215)
- sam local start-api (p. 216)
- sam local start-lambda (p. 218)
- sam logs (p. 220)
- sam package (p. 221)
- sam publish (p. 222)
- sam validate (p. 223)

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.

The `sam build` command processes your AWS SAM template file, application code, and any applicable language-specific files and dependencies, and copies build artifacts in the format and location expected by 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.

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 identifiers for an AWS Lambda runtime, or makefile:

- **AWS Lambda runtime identifier.** Build the resource against a Lambda runtime. For the list of supported runtime identifiers, see AWS Lambda runtimes.
- **makefile.** You must have a makefile that includes a build target named `build-resource-logical-id`. In this case, `sam build` executes the commands of the build target.

You can use the Metadata resource attribute with a BuildMethod entry to build layers and custom runtimes. For information about building layers, see Building layers (p. 173). For information about building custom runtimes, see Building custom runtimes (p. 174).

To see an end-to-end example that uses this command, including locally testing and deploying to the AWS Cloud, see Tutorial: Deploying a Hello World application (p. 12). The `sam build` command is part of Step 2: Build your application (p. 14).

Usage:

```
sam build [OPTIONS] [RESOURCE_LOGICAL_ID]
```
Examples:

To use this command, 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 folder containing
SAM template. Built artifacts will be written to .aws-sam/build folder
$ sam build

To build inside a AWS Lambda like Docker container
$ sam build --use-container

To build & 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

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 folder where the built artifacts are stored. This directory and all of its content is removed with this option.</td>
</tr>
<tr>
<td>-s, --base-dir DIRECTORY</td>
<td>Resolves relative paths to the function's source code with respect to this folder. Use this option if the AWS SAM template and your source code aren't in the same folder. By default, relative paths are resolved with respect to the template's location.</td>
</tr>
<tr>
<td>-u, --use-container</td>
<td>If your functions depend on packages that have natively compiled dependencies, use this flag to build your function inside an AWS Lambda-like Docker container.</td>
</tr>
<tr>
<td>-m, --manifest PATH</td>
<td>The path to a custom dependency manifest file (for example, package.json) to use instead of the default.</td>
</tr>
<tr>
<td>-t, --template-file, --template PATH</td>
<td>The path and file name of AWS SAM template file [default: template.[yaml</td>
</tr>
<tr>
<td>--parameter-overrides</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>--skip-pull-image</td>
<td>Specifies whether the command should skip pulling down the latest Docker image for the Lambda runtime.</td>
</tr>
<tr>
<td>--docker-network TEXT</td>
<td>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.</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>
</tbody>
</table>
### Option

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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. 224).</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. 224).</td>
</tr>
<tr>
<td><code>--debug</code></td>
<td>Turns on debug logging to print debug message generated by the AWS SAM CLI.</td>
</tr>
<tr>
<td><code>--help</code></td>
<td>Shows this message and exits.</td>
</tr>
</tbody>
</table>

### 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. For examples of these objects and the corresponding prompts, see the `Example` 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. 224).

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:

```
sam deploy [OPTIONS] [ARGS]...
```

### Options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-g, --guided</code></td>
<td>Specify this flag to have AWS SAM use prompts to guide you through the deployment.</td>
</tr>
<tr>
<td><code>-t, --template-file, --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>--stack-name TEXT</code></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>
</tbody>
</table>
## Option Description

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--s3-bucket TEXT</code></td>
<td>The name 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><code>--s3-prefix TEXT</code></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><code>--capabilities LIST</code></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 <code>CAPABILITY_IAM</code> and <code>CAPABILITY_NAMED_IAM</code>. If you have IAM resources, you can specify either capability. If you have IAM resources with custom names, you must specify <code>CAPABILITY_NAMED_IAM</code>. If you don't specify this parameter, this operation returns an <code>InsufficientCapabilities</code> error.</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>--profile TEXT</code></td>
<td>The specific profile from your credential file that gets AWS credentials.</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>--force-upload</code></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><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><code>--fail-on-empty-changeset</code></td>
<td>Specify whether to return a non-zero exit code if there are no changes to be made to the stack. The default behavior is to return a non-zero exit code.</td>
</tr>
<tr>
<td><code>--no-fail-on-empty-changeset</code></td>
<td></td>
</tr>
<tr>
<td><code>--confirm-changeset</code></td>
<td>Prompt to confirm whether the AWS SAM CLI deploys the computed changeset.</td>
</tr>
<tr>
<td><code>--no-confirm-changeset</code></td>
<td></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>Option</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>--tags LIST</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>--parameter-overrides</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 (AW CLI). For example, ParameterKey=KeyPairName ParameterValue=MyKey ParameterKey=InstanceType ParameterValue=t1.micro.</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. 224).</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. 224).</td>
</tr>
<tr>
<td>--debug</td>
<td>Turns on debug logging.</td>
</tr>
<tr>
<td>--help</td>
<td>Shows this message and exits.</td>
</tr>
</tbody>
</table>

**Example**

Here is an example object declared in the Parameters section, and the corresponding prompt that appears when using `sam deploy --guided`.

**AWS SAM template:**

```yaml
Parameters:
  MyPar:
    Type: String
    Default: MyParVal
```

**Corresponding sam deploy --guided prompt:**

```
Parameter MyPar [MyParVal]:
```

**sam init**

Initializes a serverless application with an AWS SAM template. The template provides a folder structure for your Lambda functions, and is connected to an event source such as APIs, S3 buckets, or DynamoDB tables. This application includes everything you need to get started and to eventually extend it into a production-scale application.

**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 will be 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

```
# 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 custom 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, path).</td>
</tr>
<tr>
<td></td>
<td>This parameter is required if --no-interactive is specified and --runtime, --name, and --app-template are not provided.</td>
</tr>
<tr>
<td>-r, --runtime [python2.7</td>
<td>nodejs6.10</td>
</tr>
<tr>
<td></td>
<td>This parameter is required if --no-interactive is specified and --location is not provided.</td>
</tr>
</tbody>
</table>
### Option | Description
--- | ---
dotnetcore1.0 | 
python3.6 | go1.x | Dependency manager of your Lambda runtime
-d, --dependency-manager | gradle | mod | maven | bundler | npm | cli-package | pip | The location where the initialized application is output.
-o, --output-dir PATH | The name of your project to be generated as a folder.
-n, --name TEXT | This parameter is required if --no-interactive is specified and --location is not provided.
--app-template TEXT | Identifier of the managed application template you want to use. If 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 only available in SAM CLI version 0.30.0 or later. Specifying this parameter with an earlier version will result in an error.
--no-input | Disables Cookiecutter prompting and accept default 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. 224).
--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. 224).
--debug | Turns on debug logging.
-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

```
sam local generate-event s3 [put/delete] --bucket <bucket> --key <key> | sam local invoke -
e - <function logical id>
```

Options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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. 224).</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. 224).</td>
</tr>
<tr>
<td>--help</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
sam local invoke

Invokes a local Lambda function once and quits after invocation completes.

This is useful for developing serverless functions that handle asynchronous events (such as Amazon S3 or Amazon Kinesis events). It can also be useful if you want to compose a script of test cases. The event body can be passed in using the `--event` parameter. The runtime output (logs etc) is output to `stderr`, and the Lambda function result is output to `stdout`.

Usage:

```
sam local invoke [OPTIONS] [FUNCTION_IDENTIFIER]
```

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 <code>stdin</code> 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 variables files, see Environment Variable File (p. 177).</td>
</tr>
<tr>
<td><code>--parameter-overrides</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>-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 will 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>-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...</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>this isn't specified, the Lambda containers only connect to the default bridge Docker network.</td>
<td></td>
</tr>
<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>Specifies 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. 224).</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. 224).</td>
</tr>
<tr>
<td>--debug</td>
<td>Turns on debug logging.</td>
</tr>
<tr>
<td>--help</td>
<td>Shows this message and exits.</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. 12). The `sam local start-api` command is part of Step 4: Testing your application locally (optional) (p. 17).

**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>-p, --port INTEGER</td>
<td>The local port number to listen on (default: '3000').</td>
</tr>
<tr>
<td>-s, --static-dir TEXT</td>
<td>Any static asset (for example, CSS/JavaScript/HTML) files located in this directory are presented at / .</td>
</tr>
<tr>
<td>-t, --template PATH</td>
<td>The AWS SAM template file [default: template.[yaml</td>
</tr>
<tr>
<td>-n, --env-vars PATH</td>
<td>The JSON file that contains values for the Lambda function's environment variables.</td>
</tr>
<tr>
<td>--parameter-overrides</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>-d, --debug-port TEXT</td>
<td>When specified, starts the Lambda function container in debug mode and exposes this port on the local host.</td>
</tr>
<tr>
<td>--debugger-path TEXT</td>
<td>The host path to a debugger that will be mounted into the Lambda container.</td>
</tr>
<tr>
<td>--debug-args TEXT</td>
<td>Additional arguments to be passed to the debugger.</td>
</tr>
<tr>
<td>-v, --docker-volume-</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>basedir TEXT</td>
<td></td>
</tr>
<tr>
<td>--docker-network TEXT</td>
<td>The name or ID of an existing Docker network that the Lambda Docker containers should connect to, along with the default bridge network. If this isn't specified, the Lambda containers only connect to the default bridge Docker network.</td>
</tr>
<tr>
<td>-l, --log-file TEXT</td>
<td>The log file to send runtime logs to.</td>
</tr>
<tr>
<td>--layer-cache-basedir</td>
<td>Specifies the location basedir where the Layers your template uses are downloaded to.</td>
</tr>
<tr>
<td>DIRECTORY</td>
<td></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>Specifies whether 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. 224).</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</td>
</tr>
</tbody>
</table>
Option | Description
---|---
--- | information about configuration files, see AWS SAM CLI configuration file (p. 224).
--debug | Turns on debug logging.
--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:**

```bash
sam local start-lambda [OPTIONS]
```

**Examples:**

```bash
# 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:
# ```python
#     self.lambda_client = boto3.client('lambda',
#                                         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>--host TEXT</td>
<td>The local hostname or IP address to bind to (default: '127.0.0.1').</td>
</tr>
<tr>
<td>-p, --port INTEGER</td>
<td>The local port number to listen on (default: '3001').</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>-t, --template PATH</td>
<td>The AWS SAM template file [default: template.[yaml</td>
</tr>
<tr>
<td>-n, --env-vars PATH</td>
<td>The JSON file that contains values for the Lambda function's environment variables.</td>
</tr>
<tr>
<td>--parameter-overrides</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'</td>
</tr>
<tr>
<td>-d, --debug-port TEXT</td>
<td>When specified, starts the Lambda function container in debug mode, and exposes this port on the local host.</td>
</tr>
<tr>
<td>--debug-path TEXT</td>
<td>The host path to a debugger to be mounted into the Lambda container.</td>
</tr>
<tr>
<td>--debug-args TEXT</td>
<td>Additional arguments to be passed to the debugger.</td>
</tr>
<tr>
<td>-v, --docker-volume-basedir</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>--docker-network TEXT</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>-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. 224).</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. 224).</td>
</tr>
<tr>
<td>--debug</td>
<td>Turns on debug logging.</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:

```bash
sam logs [OPTIONS]
```

Examples:

```bash
sam logs -n HelloWorldFunction --stack-name mystack
```

# Or, you can fetch logs using the function’s name.
```bash
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
```bash
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.
```bash
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.
```bash
sam logs -n HelloWorldFunction --stack-name mystack --filter "error"
```

Options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-n, --name TEXT</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>--stack-name TEXT</td>
<td>The name of the AWS CloudFormation stack that the function is a part of.</td>
</tr>
<tr>
<td>--filter TEXT</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>-s, --start-time TEXT</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>-e, --end-time TEXT</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>-t, --tail</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>--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>
</tbody>
</table>
Option | Description
--- | ---
--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. 224).
--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. 224).
--debug | Turns on debug logging to print debug messages that are generated by the AWS SAM CLI.
--help | Shows this message and exits.

**sam package**

Packages an AWS SAM application. It creates a ZIP file of your code and dependencies, and uploads it to 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. 209) now implicitly performs the functionality of sam package. You can use the sam deploy (p. 209) command directly to package and deploy your application.

**Usage:**

```
sam package [OPTIONS] [ARGS]...
```

**Options:**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>--s3-bucket TEXT</td>
<td>The name of the Amazon S3 bucket where this command uploads your AWS CloudFormation template. Required.</td>
</tr>
<tr>
<td>--s3-prefix TEXT</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.</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>--output-template-file PATH</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>--use-json</td>
<td>Output JSON for the AWS CloudFormation template. YAML is used by default.</td>
</tr>
</tbody>
</table>
### Option Description

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>A map of metadata to attach to all artifacts that are referenced in your template. Optional.</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. 224).</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. 224).</td>
</tr>
<tr>
<td><code>--debug</code></td>
<td>Turns on debug logging.</td>
</tr>
<tr>
<td><code>--help</code></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. 195).

### sam publish

Publish an AWS SAM application to the AWS Serverless Application Repository. This command takes a packaged AWS SAM template and publishes the application to the specified region.

This command expects the AWS SAM template to include a Metadata containing application metadata required for publishing. Furthermore, these properties must include references to Amazon S3 buckets for LicenseUrl and ReadmeUrl values, and not references to local files. For more details about the Metadata section of the AWS SAM template, see Publishing serverless applications using the AWS SAM CLI (p. 195).

This command creates the application as private by default, so you must share the application before other AWS accounts are allowed to view and deploy the application. For more information on sharing applications see Using Resource-Based Policies for the AWS Serverless Application Repository.

**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><code>-t, --template PATH</code></td>
<td>AWS SAM template file [default: template.[yaml</td>
</tr>
<tr>
<td><code>--semantic-version TEXT</code></td>
<td>Optional. The semantic version of the application provided by this parameter overrides SemanticVersion in the Metadata section of the template file. <a href="https://semver.org/">https://semver.org/</a></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. 224).</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. 224).</td>
</tr>
<tr>
<td><code>--debug</code></td>
<td>Turns on debug logging.</td>
</tr>
<tr>
<td><code>--help</code></td>
<td>Shows this message and exits.</td>
</tr>
</tbody>
</table>

## `sam validate`

Verifies whether an AWS SAM template file is valid.

### Usage:

```
sam validate [OPTIONS]
```

### Options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-t, --template-file PATH</code></td>
<td>The AWS SAM template file [default: template.[yaml</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. 224).</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. 224).</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>--debug</td>
<td>Turns on debug logging.</td>
</tr>
<tr>
<td>--help</td>
<td>Shows this message and exits.</td>
</tr>
</tbody>
</table>

### 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](https://toml.io/). The default file name is `samconfig.toml`, and the file's default location is your project's root directory, which is the directory that 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 details about this functionality, see [Writing configurations with sam deploy --guided](#) later in this topic.

### Example

Here is an example configuration file that contains seven parameters for the `default` environment and the `deploy` command:

```toml
version=0.1
[default.deploy.parameters]
stack_name = "my-app-stack"
s3_bucket = "my-source-bucket"
s3_prefix = "my-s3-prefix"
region = "us-west-2"
confirm_changeset = true
capabilities = ["CAPABILITY_IAM"]
tags = project="my-application" stage="production"
```

### Configuration file rules

The configuration file is in the [TOML file format](https://toml.io/). 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.
- 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]`. 
AWS Serverless Application Model Developer Guide

Writing configurations with `sam deploy --guided`

- For subcommands, the format of the table header is `[$environment.$command.$subcommand].parameters`. For example, for the `sam local invoke` command, the configuration table header is `[default.local_invoke.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/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. 207), or run `sam command --help`.
- The configuration value can take the following forms:
  1. For parameters that take an argument, the entry value is the argument surrounded by double quotes. For example, `region = "us-west-2"`.
  2. For arguments that are key-value pairs, the pairs are space-delimited and value of each pair is surrounded by encoded double quotes. For example, `tags = "project="my-application" stage="production""`.
  3. For toggle parameters, the value can be `true` or `false` (no quotes). For example, `confirm_changeset = true`.

**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`
- `region`
- `config_changeset`
- `capabilities`
- `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**

```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. 232) 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.

```yaml
MyFunction:
  Type: 'AWS::Serverless::Function'
  Properties:
    CodeUri: ${codeuri}
    Handler: hello.handler
    Runtime: python2.7
    Policies:
      - SQSPollerPolicy:
```

---

226
Example 2: Policy template with no placeholder values

The following example contains the CloudWatchPutMetricPolicy (p. 233) 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.

```yaml
MyFunction:
  Type: 'AWS::Serverless::Function'
  Properties:
    CodeUri: ${codeuri}
    Handler: hello.handler
    Runtime: python2.7
    Policies:
      - CloudWatchPutMetricPolicy: {}
```

Policy template table

The following is a table of the available policy templates.

<table>
<thead>
<tr>
<th>Policy Template</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQSPollerPolicy (p. 232)</td>
<td>Gives permission to poll an Amazon Simple Queue Service (Amazon SQS) queue.</td>
</tr>
<tr>
<td>LambdaInvokePolicy (p. 232)</td>
<td>Gives permission to invoke an AWS Lambda function, alias, or version.</td>
</tr>
<tr>
<td>CloudWatchDescribeAlarmHistoryPolicy (p. 232)</td>
<td>Gives permission to describe CloudWatch alarm history.</td>
</tr>
<tr>
<td>CloudWatchPutMetricPolicy (p. 233)</td>
<td>Gives permission to send metrics to CloudWatch.</td>
</tr>
<tr>
<td>EC2DescribePolicy (p. 233)</td>
<td>Gives permission to describe Amazon Elastic Compute Cloud (Amazon EC2) instances.</td>
</tr>
<tr>
<td>DynamoDBCrudPolicy (p. 233)</td>
<td>Gives create, read, update, and delete permissions to an Amazon DynamoDB table.</td>
</tr>
<tr>
<td>DynamoDBReadPolicy (p. 234)</td>
<td>Gives read-only permission to a DynamoDB table.</td>
</tr>
<tr>
<td>DynamoDBWritePolicy (p. 235)</td>
<td>Gives write-only permission to a DynamoDB table.</td>
</tr>
<tr>
<td>DynamoDBReconfigurePolicy (p. 235)</td>
<td>Gives permission to reconfigure a DynamoDB table.</td>
</tr>
<tr>
<td>SESSendBouncePolicy (p. 236)</td>
<td>Gives SendBounce permission to an Amazon Simple Email Service (Amazon SES) identity.</td>
</tr>
<tr>
<td>Policy Template</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>ElasticsearchHttpPostPolicy</td>
<td>Gives POST permission to Amazon Elasticsearch Service.</td>
</tr>
<tr>
<td>S3ReadPolicy (p. 237)</td>
<td>Gives read-only permission to objects in an Amazon Simple Storage Service (Amazon S3) bucket.</td>
</tr>
<tr>
<td>S3WritePolicy (p. 237)</td>
<td>Gives write permission to objects in an Amazon S3 bucket.</td>
</tr>
<tr>
<td>S3CrudPolicy (p. 237)</td>
<td>Gives create, read, update, and delete permission to 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>CloudFormationDescribePolicy</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. 242)</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. 244)</td>
<td>Gives permission to send email and verify identity.</td>
</tr>
<tr>
<td>SNSCrudPolicy (p. 244)</td>
<td>Gives permission to create, publish, and subscribe to Amazon SNS topics.</td>
</tr>
<tr>
<td>KinesisCrudPolicy</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>Policy Template</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------------------------------------------------------------</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 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>AWSSecretsManagerRotateSecretPolicy</td>
<td>Gives permission to rotate a secret in AWS Secrets Manager.</td>
</tr>
<tr>
<td>AWSSecretsManagerGetSecretValuePolicy</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>DynamoDBRestoreFromBackupPolicy</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>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>Policy Template</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</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 permissions to execute Athena queries.</td>
</tr>
<tr>
<td>TextractPolicy (p. 261)</td>
<td>Gives full access to Amazon Textract.</td>
</tr>
<tr>
<td>TextractDetectAnalysePolicy</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>
<tr>
<td>EventBridgePutEventsPolicy</td>
<td>Gives permissions to send events to EventBridge.</td>
</tr>
<tr>
<td>ElasticMapReducePolicy</td>
<td>Gives permission to list details and modify capacities for instance fleets within a cluster.</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. 233).

```
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.
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "cloudwatch:DescribeAlarmHistory"
    ],
    "Resource": "*"
  }
]

**CloudWatchPutMetricPolicy**

Gives permission to send metrics to CloudWatch.

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "cloudwatch:PutMetricData"
    ],
    "Resource": "*"
  }
]

**EC2DescribePolicy**

Gives permission to describe Amazon Elastic Compute Cloud (Amazon EC2) instances.

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ec2:DescribeRegions",
      "ec2:DescribeInstances"
    ],
    "Resource": "*"
  }
]

**DynamoDBCrudPolicy**

Gives create, read, update, and delete permissions to an Amazon DynamoDB table.

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "dynamodb:GetItem",
      "dynamodb:DeleteItem",
      "dynamodb:PutItem",
      "dynamodb:Scan",
      "dynamodb:Query",
      "dynamodb:UpdateItem",
      "dynamodb:GetAttributes",
      "dynamodb:BatchGetItem"
    ],
    "Resource": "*"
  }
]
"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"
}
}
]
}
]
]

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/
${tableName}",
{
"tableName": {
"Ref": "TableName"
}
}
}
],
{
"Fn::Sub": [
"arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
${tableName}/index/*",
{
"tableName": {
"Ref": "TableName"
}
}
]
}
DynamoDBWritePolicy

Gives write-only permission to a DynamoDB table.

```
"Statement": [
  { "Effect": "Allow",
    "Action": [ "dynamodb:PutItem",
                 "dynamodb:UpdateItem",
                 "dynamodb:BatchWriteItem"
               ],
    "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" }
      }
      ]
    ]
  }
]
```

DynamoDBReconfigurePolicy

Gives permission to reconfigure a DynamoDB table.

```
"Statement": [
  { "Effect": "Allow",
    "Action": [ "dynamodb:ReconfigureTable"
               ],
    "Resource": [ "${TableName}" ]
  }
]
```
Policy template list

"Action": [ "dynamodb:UpdateTable"
],
"Resource": {
"Fn::Sub": [
"arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
${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}"
]
}
}
]

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}/*"
]
}
}]

236
"domainName": { 
  "Ref": "DomainName"
}
]

S3ReadPolicy

Gives read-only permission to 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 objects in an Amazon S3 bucket.

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "s3:PutObject"
    ],
    "Resource": [
      "Fn::Sub": [
        "arn:${AWS::Partition}:s3:::${bucketName}/*",
        {
          "bucketName": {
            "Ref": "BucketName"
          }
        }
      ]
    ]
  }
]
S3CrudPolicy

Gives create, read, update, and delete permission to 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"
            }
          }
        ],
      }
    ]
  }
]
"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",
```
**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}"
      ]
    }
  }
]
```

**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}"
      ]
    }
  }
]
```
Policy template list

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": [  
      "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/${streamName}",  
        {  
          "tableName": {  
            "Ref": "TableName"  
          }  
        }  
      ]  
    }  
  }]
"Statement": [  
  {  
    "Effect": "Allow",
    "Action": [  
      "dynamodb:ListStreams"
    ],
    "Resource": {  
      "Fn::Sub": [  
        "arn:${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}"
      ]
    }
  }
]
**SESCrudPolicy**

Gives permission to send email and verify identity.

```json
"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}"
    ]
  }
}
]
```

**SNSCrudPolicy**

Gives permission to create, publish, and subscribe to Amazon SNS topics.

```json
"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}"
    ]
  }
}
]
```
KinesisCrudPolicy

Gives permission to create, publish, and delete an Amazon Kinesis stream.

```
"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.

```
"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.

```
"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"
      }
    }
  ]
}
,
{
  "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 objects in an Amazon S3 bucket.

```json
"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:GetObjectTagging",
      "s3:GetObjectVersionTagging",
      "s3:PutObjectTagging",
      "s3:PutObjectVersionTagging"
    ],
    "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}"
        ]
      }
    ]
  }
]
```
**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": {  
            "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}",
        
      "pipelinename": {
        "Ref": "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}"
    ],
    "collectionId": {
      "Ref": "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.

```json
"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.

```json
"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}"
      ]
   }
}
]
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", 
  ]
}]
```
"ce:GetReservationUtilization",
  "ce:GetTags"
],
"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: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",

```
"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",]
"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: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}:
  "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:ListNamedQuery",
      "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.

```
"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.

```
"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.

```
"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}"
            ]
        },
        "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}"
            ]
        },
        "Effect": "Allow"
    }
]
```

SageMakerCreateEndpointConfigPolicy
Gives permission to create an endpoint configuration in SageMaker.

```
"Statement": [
    {
        "Action": [
```

265
EcsRunTaskPolicy

Gives permission to start a new task for a task definition.

```json
"Statement": [
{
  "Action": [
    "ecs:RunTask"
  ],
  "Resource": {
    "Fn::Sub": [
      "arn:${AWS::Partition}:ecs:${AWS::Region}:${AWS::AccountId}:task-definition/${taskDefinition}",
      { 
        "taskDefinition": { 
          "Ref": "TaskDefinition"
        }
      }
    ]
  },
  "Effect": "Allow"
}
]
```

EFSWriteAccessPolicy

Gives permission to mount an Amazon EFS file system with write access.

```json
"Statement": [
{
  "Effect": "Allow",
  "Action": [
    "elasticfilesystem:ClientMount",
    "elasticfilesystem:ClientWrite"
  ],
  "Resource": {
    "Fn::Sub": [
      "arn:${AWS::Partition}:elasticfilesystem:${AWS::Region}:${AWS::AccountId}:file-system/${FileSystem}",
      { 
        "FileSystem": {
          "Ref": "FileSystem"
        }
      }
    ]
  }
}
]"
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. 268).

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.

```bash
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:
echo "export SAM_CLI_TELEMETRY=0" >> ~/.profile

2. Run:

source ~/.profile

To disable telemetry in macOS

1. Run:

   echo "export SAM_CLI_TELEMETRY=0" >> ~/.profile

2. Run:

   source ~/.profile

To disable telemetry in Windows

1. Run:

   setx SAM_CLI_TELEMETRY 0

2. Run:

   refreshenv

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

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. 7).

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:** September 24, 2020

<table>
<thead>
<tr>
<th>Update History Change</th>
<th>Update History Description</th>
<th>Update History Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support for multiple configuration files and environments</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</td>
<td>Added support for X-Ray as an event source for serverless state machines. For more information, see the <code>Tracing</code> property of the <code>AWS::Serverless::StateMachine</code> resource type. Also added support for references when controlling access to APIs. For more information, see the <code>ResourcePolicyStatement</code> data type.</td>
<td>September 17, 2020</td>
</tr>
<tr>
<td>Support for Amazon MSK</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 <code>EventSource</code> and <code>MSK</code> data types of the <code>AWS::Serverless::Function</code> resource type.</td>
<td>August 13, 2020</td>
</tr>
<tr>
<td>Support for Amazon EFS</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 <code>FileSystemConfigs</code> property of the <code>AWS::Serverless::Function</code> resource type.</td>
<td>June 16, 2020</td>
</tr>
<tr>
<td>Orchestrating serverless applications</td>
<td>Added support for orchestrating applications by creating Step</td>
<td>May 27, 2020</td>
</tr>
<tr>
<td>Update Description</td>
<td>Details</td>
<td></td>
</tr>
<tr>
<td>--------------------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>Functions state machines using AWS SAM</td>
<td>For more information, see Orchestrating AWS Resources with AWS Step Functions and the AWS::Serverless::StateMachine resource type.</td>
<td></td>
</tr>
<tr>
<td>Building custom runtimes (p. 270)</td>
<td>Added the ability to build custom runtimes. For more information, see Building Custom Runtimes. May 21, 2020</td>
<td></td>
</tr>
<tr>
<td>Building layers (p. 270)</td>
<td>Added the ability to build individual LayerVersion resources. For more information, see Building Layers. May 19, 2020</td>
<td></td>
</tr>
<tr>
<td>Generated AWS CloudFormation Resources (p. 270)</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. April 8, 2020</td>
<td></td>
</tr>
<tr>
<td>Setting up AWS credentials (p. 270)</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. January 17, 2020</td>
<td></td>
</tr>
<tr>
<td>AWS SAM Specification and AWS SAM CLI updates (p. 270)</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. November 25, 2019</td>
<td></td>
</tr>
<tr>
<td>New options for controlling access to API Gateway APIs and policy template updates (p. 270)</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. August 29, 2019</td>
<td></td>
</tr>
<tr>
<td>Getting Started updates (p. 270)</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 (p. 270)</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 sam publish to the AWS SAM CLI (p. 270)</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 (p. 270)</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. 270)</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. 270)</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. 270)</td>
<td>This is the first release of the AWS Serverless Application Model Developer Guide.</td>
<td>October 17, 2018</td>
</tr>
</tbody>
</table>