aws

AWS Serverless Application Model

AWS Serverless Application Model: Developer Guide

Table of Contents

What is the AWS Serverless Application Model (AWS SAM)?

The AWS Serverless Application Model (AWS SAM) is a toolkit that improves the developer experience of building and running serverless applications on AWS. AWS SAM offers a variety of benefits:

Define your application infrastructure code quickly, using less code

Author AWS SAM templates to define your serverless application infrastructure code. Deploy your templates directly to AWS CloudFormation to provision your resources.

Manage your serverless applications through their entire development lifecycle

Use the AWS SAM CLI to manage your serverless application through the authoring, building, deploying, testing, and monitoring phases of your development lifecycle. For more information, see Using the AWS SAM CLI.

Quickly provision permissions between resources with AWS SAM connectors

Use AWS SAM connectors in your AWS SAM templates to define permissions between your AWS resources. AWS SAM transforms your code into the IAM permissions required to facilitate your intent. For more information, see Managing resource permissions with AWS SAM connectors.

Continuously sync local changes to the cloud as you develop

Use the AWS SAM CLI **sam sync** command to automatically sync local changes to the cloud, speeding up your development and cloud testing workflows. For more information, see Using sam sync.

Manage your Terraform serverless applications

Use the AWS SAM CLI to perform local debugging and testing of your Lambda functions and layers. For more information, see AWS SAM CLI Terraform support.

Topics

- Basic Concepts
- What is the AWS SAM template specification?
- What is the AWS SAM template?
- What is the AWS SAM CLI?

- [Learn more](#)
- [Next steps](#)
- [Serverless concepts](#)

Basic Concepts

AWS SAM consists of three primary parts:

1. **AWS SAM template specification** – An open-source framework that you can use to define your serverless application infrastructure on AWS. This framework is accessed through AWS SAM templates.

2. **AWS SAM template** – what you use to access AWS SAM template specifications, the open-source framework that you can use to define your serverless application infrastructure on AWS. An AWS SAM template is extension of AWS CloudFormation templates with some additional components that make them easier to work with.

3. **AWS SAM command line interface (AWS SAM CLI)** – A command line tool that you can use with AWS SAM templates and supported third-party integrations to build and run your serverless applications.

> ⓘ **New to serverless?**
>
> We recommend you review [Serverless concepts](#).

What is the AWS SAM template specification?

The AWS SAM template specification is an open-source framework that you can use to define and manage your serverless application infrastructure code. The AWS SAM template specification is:

- **Built on AWS CloudFormation** – Use the AWS CloudFormation syntax directly within your AWS SAM template, taking advantage of its extensive support of resource and property configurations. If you are already familiar with AWS CloudFormation, you don't have to learn a new service to manage your application infrastructure code.

- **An extension of AWS CloudFormation** – AWS SAM offers its own unique syntax that focuses specifically on speeding up serverless development. You can use both the AWS CloudFormation and AWS SAM syntax within the same template.

- **An abstract, short-hand syntax** – Using the AWS SAM syntax, you can define your infrastructure quickly, in fewer lines of code, and with a lower chance of errors. Its syntax is especially curated to abstract away the complexity in defining your serverless application infrastructure.

- **Transformational** – AWS SAM does the complex work of transforming your template into the code necessary to provision your infrastructure through AWS CloudFormation.

What is the AWS SAM template?

An AWS SAM template is what you use to access AWS SAM template specifications, the open-source framework that you can use to define your serverless application infrastructure on AWS, with some additional components that make them easier to work with. In this sense, AWS SAM templates are an extension of AWS CloudFormation templates.

Here's an example of a basic serverless application. This application processes requests to get all items from a database through an HTTP request. It consists of the following parts:

1. A function that contains the logic to process the request.
2. An HTTP API to serve as communication between the client (requestor) and the application.
3. A database to store items.
4. Permissions for the application to run securely.
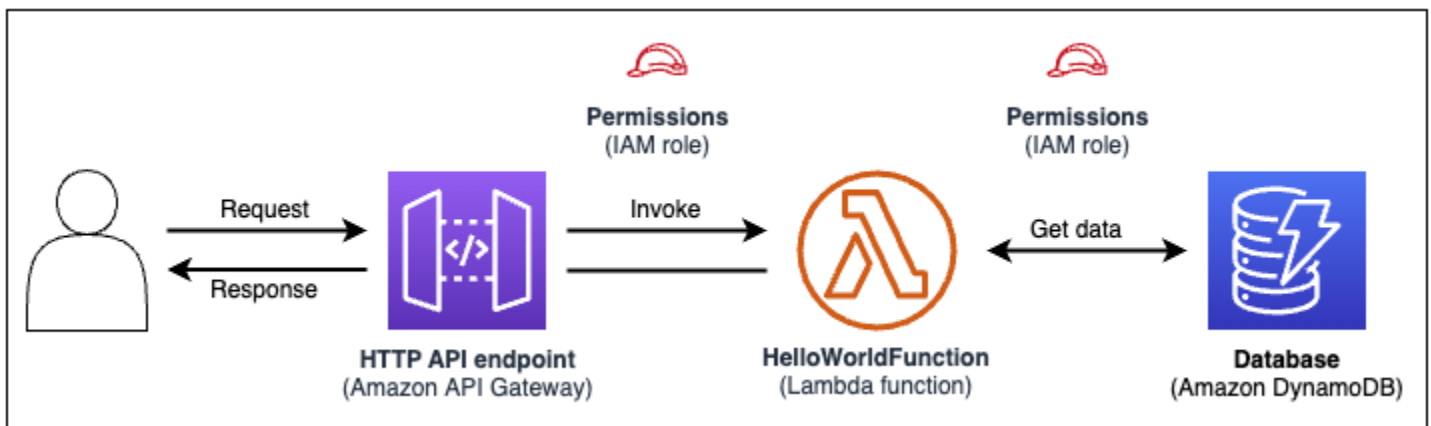
This application's infrastructure code can be defined in the following AWS SAM template:

```
AWSTemplateFormatVersion: 2010-09-09
Transform: AWS::Serverless-2016-10-31
Resources:
  getAllItemsFunction:
```

```
      Type: AWS::Serverless::Function
      Properties:
        Handler: src/get-all-items.getAllItemsHandler
        Runtime: nodejs12.x
        Events:
          Api:
            Type: HttpApi
            Properties:
              Path: /
              Method: GET
      Connectors:
        MyConn:
          Properties:
            Destination:
              Id: SampleTable
            Permissions:
              - Read
  SampleTable:
    Type: AWS::Serverless::SimpleTable
```

In 23 lines of code, the following infrastructure is defined:

- A function using the AWS Lambda service.

- An HTTP API using the Amazon API Gateway service.

- A database using the Amazon DynamoDB service.

- The AWS Identity and Access Management (IAM) permissions necessary for these services to interact with one another.

To provision this infrastructure, the template is deployed to AWS CloudFormation. During deployment, AWS SAM transforms the 23 lines of code into the AWS CloudFormation syntax required to generate these resources in AWS. The transformed AWS CloudFormation template contains over 200 lines of code!

Transformed AWS CloudFormation template

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Resources": {
    "getAllItemsFunction": {
      "Type": "AWS::Lambda::Function",
```

```
      "Metadata": {
        "SamResourceId": "getAllItemsFunction"
      },
      "Properties": {
        "Code": {
          "S3Bucket": "aws-sam-cli-managed-default-samclisourcebucket-1a4x26zbcdkqr",
          "S3Key": "what-is-app/a6f856abf1b2c4f7488c09b367540b5b"
        },
        "Handler": "src/get-all-items.getAllItemsHandler",
        "Role": {
          "Fn::GetAtt": [
            "getAllItemsFunctionRole",
            "Arn"
          ]
        },
        "Runtime": "nodejs12.x",
        "Tags": [
          {
            "Key": "lambda:createdBy",
            "Value": "SAM"
          }
        ]
      }
    },
    "getAllItemsFunctionRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Action": [
                "sts:AssumeRole"
              ],
              "Effect": "Allow",
              "Principal": {
                "Service": [
                  "lambda.amazonaws.com"
                ]
              }
            }
          ]
        },
        "ManagedPolicyArns": [
```

```json
            "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
          ],
          "Tags": [
            {
              "Key": "lambda:createdBy",
              "Value": "SAM"
            }
          ]
        }
      },
      "getAllItemsFunctionApiPermission": {
        "Type": "AWS::Lambda::Permission",
        "Properties": {
          "Action": "lambda:InvokeFunction",
          "FunctionName": {
            "Ref": "getAllItemsFunction"
          },
          "Principal": "apigateway.amazonaws.com",
          "SourceArn": {
            "Fn::Sub": [
              "arn:${AWS::Partition}:execute-api:${AWS::Region}:${AWS::AccountId}:
${__ApiId__}/${__Stage__}/GET/",
              {
                "__ApiId__": {
                  "Ref": "ServerlessHttpApi"
                },
                "__Stage__": "*"
              }
            ]
          }
        }
      },
      "ServerlessHttpApi": {
        "Type": "AWS::ApiGatewayV2::Api",
        "Properties": {
          "Body": {
            "info": {
              "version": "1.0",
              "title": {
                "Ref": "AWS::StackName"
              }
            },
            "paths": {
              "/": {
```

```
                "get": {
                  "x-amazon-apigateway-integration": {
                    "httpMethod": "POST",
                    "type": "aws_proxy",
                    "uri": {
                      "Fn::Sub": "arn:${AWS::Partition}:apigateway:
${AWS::Region}:lambda:path/2015-03-31/functions/${getAllItemsFunction.Arn}/invocations"
                    },
                    "payloadFormatVersion": "2.0"
                  },
                  "responses": {}
                }
              }
            },
            "openapi": "3.0.1",
            "tags": [
              {
                "name": "httpapi:createdBy",
                "x-amazon-apigateway-tag-value": "SAM"
              }
            ]
          }
        }
      },
      "ServerlessHttpApiApiGatewayDefaultStage": {
        "Type": "AWS::ApiGatewayV2::Stage",
        "Properties": {
          "ApiId": {
            "Ref": "ServerlessHttpApi"
          },
          "StageName": "$default",
          "Tags": {
            "httpapi:createdBy": "SAM"
          },
          "AutoDeploy": true
        }
      },
      "SampleTable": {
        "Type": "AWS::DynamoDB::Table",
        "Metadata": {
          "SamResourceId": "SampleTable"
        },
        "Properties": {
          "AttributeDefinitions": [
```

```
        {
          "AttributeName": "id",
          "AttributeType": "S"
        }
      ],
      "KeySchema": [
        {
          "AttributeName": "id",
          "KeyType": "HASH"
        }
      ],
      "BillingMode": "PAY_PER_REQUEST"
    }
  },
  "getAllItemsFunctionMyConnPolicy": {
    "Type": "AWS::IAM::ManagedPolicy",
    "Metadata": {
      "aws:sam:connectors": {
        "getAllItemsFunctionMyConn": {
          "Source": {
            "Type": "AWS::Serverless::Function"
          },
          "Destination": {
            "Type": "AWS::Serverless::SimpleTable"
          }
        }
      }
    },
    "Properties": {
      "PolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
          {
            "Effect": "Allow",
            "Action": [
              "dynamodb:GetItem",
              "dynamodb:Query",
              "dynamodb:Scan",
              "dynamodb:BatchGetItem",
              "dynamodb:ConditionCheckItem",
              "dynamodb:PartiQLSelect"
            ],
            "Resource": [
              {
```

```
                "Fn::GetAtt": [
                  "SampleTable",
                  "Arn"
                ]
              },
              {
                "Fn::Sub": [
                  "${DestinationArn}/index/*",
                  {
                    "DestinationArn": {
                      "Fn::GetAtt": [
                        "SampleTable",
                        "Arn"
                      ]
                    }
                  }
                ]
              }
            ]
          }
        ]
      },
      "Roles": [
        {
          "Ref": "getAllItemsFunctionRole"
        }
      ]
    }
  }
 }
}
```

By using AWS SAM, you define 23 lines of infrastructure code. AWS SAM transforms your code into the 200+ lines of AWS CloudFormation code necessary to provision your application.

What is the AWS SAM CLI?

The AWS SAM CLI is a command line tool that you can use with AWS SAM templates and supported third-party integrations to build and run your serverless applications. Use the AWS SAM CLI to:

- Quickly initialize a new application project.
- Build your application for deployment.

- Perform local debugging and testing.

- Deploy your application.

- Configure CI/CD deployment pipelines.

- Monitor and troubleshoot your application in the cloud.

- Sync local changes to the cloud as you develop.

- And more!

The AWS SAM CLI is best utilized when used with AWS SAM and AWS CloudFormation templates. It also works with third-party products such as Terraform.

Initialize a new project

Select from starter templates or choose a custom template location to begin a new project.

Here, we use the **sam init** command to initialize a new application project. We select the **Hello World Example** project to start with. The AWS SAM CLI downloads a starter template and creates our project folder directory structure.

```
→   what-is sam init

You can preselect a particular runtime or package type when using the `sam init` experience.
Call `sam init --help` to learn more.

Which template source would you like to use?
        1 - AWS Quick Start Templates
        2 - Custom Template Location
Choice: 1

Choose an AWS Quick Start application template
        1 - Hello World Example
        2 - Multi-step workflow
        3 - Serverless API
        4 - Scheduled task
        5 - Standalone function
        6 - Data processing
        7 - Infrastructure event management
        8 - Serverless Connector Hello World Example
        9 - Multi-step workflow with Connectors
        10 - Lambda EFS example
        11 - Machine Learning
Template: 1

Use the most popular runtime and package type? (Python and zip) [y/N]: ▌
```

Build your application for deployment

Package your function dependencies and organize your project code and folder structure to prepare for deployment.

Here, we use the **sam build** command to prepare our application for deployment. The AWS SAM CLI creates a `.aws-sam`directory and organizes our application dependencies and files there for deployment.

```
→  sam-app sam build
Building codeuri: /Users/evzz/Demo/what-is/sam-app/hello_world runtime: python3.9 metadata: {} archit
ecture: x86_64 functions: HelloWorldFunction
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource

Build Succeeded

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

Commands you can use next
=========================
[*] Validate SAM template: sam validate
[*] Invoke Function: sam local invoke
[*] Test Function in the Cloud: sam sync --stack-name {{stack-name}} --watch
[*] Deploy: sam deploy --guided
→  sam-app cd .aws-sam
→  .aws-sam ls
build      build.toml
→  .aws-sam ▊
```

Perform local debugging and testing

On your local machine, simulate events, test APIs, invoke functions, and more to debug and test your application.

Here, we use the **sam local invoke** command to invoke our `HelloWorldFunction` locally. To accomplish this, the AWS SAM CLI creates a local container, builds our function, invokes it, and outputs the results.

```
→  sam-app sam local invoke HelloWorldFunction
Invoking app.lambda_handler (python3.9)
Local image was not found.
Removing rapid images for repo public.ecr.aws/sam/emulation-python3.9
Building image.........................................................................
...............................................................................................
.....................................................
Using local image: public.ecr.aws/lambda/python:3.9-rapid-x86_64.

Mounting /Users/evzz/Demo/what-is/sam-app/.aws-sam/build/HelloWorldFunction as /var/task:ro,delegated
 inside runtime container
START RequestId: 6f8347ce-6b04-4246-a0de-6dc37f0eef51 Version: $LATEST
END RequestId: 6f8347ce-6b04-4246-a0de-6dc37f0eef51
REPORT RequestId: 6f8347ce-6b04-4246-a0de-6dc37f0eef51  Init Duration: 1.23 ms  Duration: 639.26 ms B
illed Duration: 640 ms  Memory Size: 128 MB     Max Memory Used: 128 MB
{"statusCode": 200, "body": "{\"message\": \"hello world\"}"}
```

Deploy your application

Configure your application's deployment settings and deploy to the AWS Cloud to provision your resources.

Here, we use the **sam deploy --guided** command to deploy our application through an interactive flow. The AWS SAM CLI guides us through configuring our application's deployment settings, transforms our template into AWS CloudFormation, and deploys to AWS CloudFormation to create our resources.

```
→  sam-app sam deploy --guided

Configuring SAM deploy
=====================

        Looking for config file [samconfig.toml] :  Not found

        Setting default arguments for 'sam deploy'
        ==========================================
        Stack Name [sam-app]:
        AWS Region [us-west-2]:
        #Shows you resources changes to be deployed and require a 'Y' to initiate deploy
        Confirm changes before deploy [y/N]:
        #SAM needs permission to be able to create roles to connect to the resources in your template
        Allow SAM CLI IAM role creation [Y/n]:
        #Preserves the state of previously provisioned resources when an operation fails
        Disable rollback [y/N]:
        HelloWorldFunction may not have authorization defined, Is this okay? [y/N]: y
        Save arguments to configuration file [Y/n]:
        SAM configuration file [samconfig.toml]:
        SAM configuration environment [default]:

        Looking for resources needed for deployment:
         Managed S3 bucket: aws-sam-cli-managed-default-samclisourcebucket-1a4x26zbcdkqr
         A different default S3 bucket can be set in samconfig.toml
```

Configure CI/CD deployment pipelines

Create secure *continuous integration and delivery (CI/CD)* pipelines, using a supported CI/CD system.

Here, we use the **sam pipeline init --bootstrap** command to configure a CI/CD deployment pipeline for our application. The AWS SAM CLI guides us through our options and generates the AWS resources and configuration file to use with our CI/CD system.

```
[3] Reference application build resources
Enter the pipeline execution role ARN if you have previously created one, or we will create one for y
ou []:
Enter the CloudFormation execution role ARN if you have previously created one, or we will create one
 for you []:
Please enter the artifact bucket ARN for your Lambda function. If you do not have a bucket, we will c
reate one for you []:
Does your application contain any IMAGE type Lambda functions? [y/N]: n

[4] Summary
Below is the summary of the answers:
        1 - Account: 513423067560
        2 - Stage configuration name: dev
        3 - Region: us-west-2
        4 - Pipeline user: [to be created]
        5 - Pipeline execution role: [to be created]
        6 - CloudFormation execution role: [to be created]
        7 - Artifacts bucket: [to be created]
        8 - ECR image repository: [skipped]
Press enter to confirm the values above, or select an item to edit the value:

This will create the following required resources for the 'dev' configuration:
        - Pipeline IAM user
        - Pipeline execution role
        - CloudFormation execution role
        - Artifact bucket
Should we proceed with the creation? [y/N]:
```

Monitor and troubleshoot your application in the cloud

View important information about your deployed resources, gather logs, and utilize built-in monitoring tools such as AWS X-Ray.

Here, we use the **sam list** command to view our deployed resources. We get our API endpoint and invoke it, which triggers our function. Then, we use **sam logs** to view our function's logs.

```
→  sam-app sam logs --stack-name sam-app
2023/03/13/[$LATEST]0a433e844dd445bd82d0d78cd55e0cdc 2023-03-13T21:06:42.075000 INIT_START Runtime Ve
rsion: python:3.9.v16   Runtime Version ARN: arn:aws:lambda:us-west-2::runtime:07a48df201798d627f2b95
0f03bb227aab4a655a1d019c3296406f95937e2525
2023/03/13/[$LATEST]0a433e844dd445bd82d0d78cd55e0cdc 2023-03-13T21:06:42.180000 START RequestId: 778e
4226-0a80-435f-929b-5b19292ed9a7 Version: $LATEST
2023/03/13/[$LATEST]0a433e844dd445bd82d0d78cd55e0cdc 2023-03-13T21:06:42.181000 END RequestId: 778e42
26-0a80-435f-929b-5b19292ed9a7
2023/03/13/[$LATEST]0a433e844dd445bd82d0d78cd55e0cdc 2023-03-13T21:06:42.182000 REPORT RequestId: 778
e4226-0a80-435f-929b-5b19292ed9a7        Duration: 1.69 ms      Billed Duration: 2 ms   Memory Size:
128 MB  Max Memory Used: 36 MB  Init Duration: 104.13 ms
```

Sync local changes to the cloud as you develop

As you develop on your local machine, automatically sync changes to the cloud. Quickly see your changes and perform testing and validation in the cloud.

Here, we use the **sam sync --watch** command to have the AWS SAM CLI watch for local changes. We modify our `HelloWorldFunction` code and the AWS SAM CLI automatically detects the change and deploys our updates to the cloud.

```
--------------------------------------------------------------------------------
Key                  HelloWorldFunctionIamRole
Description          Implicit IAM Role created for Hello World function
Value                arn:aws:iam::513423067560:role/sam-app-HelloWorldFunctionRole-15GLOUR9LMT1W

Key                  HelloWorldApi
Description          API Gateway endpoint URL for Prod stage for Hello World function
Value                https://ets1gv8lxi.execute-api.us-west-2.amazonaws.com/Prod/hello/

Key                  HelloWorldFunction
Description          Hello World Lambda Function ARN
Value                arn:aws:lambda:us-west-2:513423067560:function:sam-app-HelloWorldFunction-
yQDNe17r9maD
--------------------------------------------------------------------------------

Stack update succeeded. Sync infra completed.

Infra sync completed.
CodeTrigger not created as CodeUri or DefinitionUri is missing for ServerlessRestApi.
Syncing Lambda Function HelloWorldFunction...
Manifest is not changed for (HelloWorldFunction), running incremental build
Building codeuri: /Users/evzz/Demo/what-is/sam-app/hello_world runtime: python3.9 metadata: {} archit
ecture: x86_64 functions: HelloWorldFunction
Running PythonPipBuilder:CopySource
Finished syncing Lambda Function HelloWorldFunction.
⬚
```

Test supported resources in the cloud

Invoke and pass events to supported resources in the cloud.

Here, we use the **sam remote invoke** command to test a deployed Lambda function in the cloud. We invoke our Lambda function and receive its logs and response. With our Lambda function configured to stream responses, the AWS SAM CLI streams its response back in real time.

Learn more

To continue learning about AWS SAM, see the following resources:

- **The Complete AWS SAM Workshop** – A workshop designed to teach you many of the major features that AWS SAM provides.

- **Sessions with SAM** – Video series created by our AWS Serverless Developer Advocate team on using AWS SAM.

- **Serverless Land** – Site that brings together the latest information, blogs, videos, code, and learning resources for AWS serverless.

Next steps

If this is your first time using AWS SAM, see Getting started with AWS SAM.

Serverless concepts

Learn about basic serverless concepts before using the AWS Serverless Application Model (AWS SAM).

Serverless concepts

Event-driven architecture

A serverless application consists of individual AWS services, such as AWS Lambda for compute and Amazon DynamoDB for database management, that each perform a specialized role. These services are then loosely integrated with each other through an event-driven architecture. To learn more about event-driven architecture, see What is an Event-Driven Architecture?.

Infrastructure as Code (IaC)

Infrastructure as Code (IaC) is a way of treating infrastructure in the same way that developers treat code, applying the same rigor of application code development to infrastructure provisioning. You define your infrastructure in a template file, deploy it to AWS, and AWS creates the resources for you. With IaC, you define in code what you want AWS to provision. For more information, see Infrastructure as Code in the *Introduction to DevOps on AWS* AWS Whitepaper.

Serverless technologies

With AWS serverless technologies, you can build and run applications without having to manage your own servers. All server management is done by AWS, providing many benefits such as automatic scaling and built-in high availability, letting you take your idea to production quickly. Using serverless technologies, you can focus on the core of your product without having to worry about managing and operating servers. To learn more about serverless, see the following:

- Serverless on AWS

- Serverless Developer Guide – Provides a conceptual overview of serverless development in the AWS Cloud.

For a basic introduction to the core AWS serverless services, see Serverless 101: Understanding the serverless services at *Serverless Land*.

Next steps

For an introduction to AWS SAM, see What is the AWS Serverless Application Model (AWS SAM)?

Getting started with AWS SAM

Get started with the AWS Serverless Application Model (AWS SAM) by installing the AWS SAM Command Line Interface (AWS SAM CLI).

Topics

- [AWS SAM prerequisites](#)
- [Installing the AWS SAM CLI](#)
- [Tutorial: Deploying a Hello World application](#)

AWS SAM prerequisites

Complete the following prerequisites before installing and using the AWS Serverless Application Model Command Line Interface (AWS SAM CLI).

To use the AWS SAM CLI, you need the following:

- An AWS account, AWS Identity and Access Management (IAM) credentials, and an IAM access key pair.
- The AWS Command Line Interface (AWS CLI) to configure AWS credentials.

Topics

- [Step 1: Sign up for an AWS account](#)
- [Step 2: Create an IAM user account](#)
- [Step 3: Create an access key ID and secret access key](#)
- [Step 4: Install the AWS CLI](#)
- [Step 5: Use the AWS CLI to configure AWS credentials](#)
- [Next steps](#)

Step 1: Sign up for an AWS account

If you do not have an AWS account, complete the following steps to create one.

To sign up for an AWS account

1. Open https://portal.aws.amazon.com/billing/signup.

2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

   When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to an administrative user, and use only the root user to perform tasks that require root user access.

Step 2: Create an IAM user account

To create an administrator user, choose one of the following options.

Choose one way to manage your administrator	To	By	You can also
In IAM Identity Center (Recommended)	Use short-term credentials to access AWS. This aligns with the security best practices. For information about best practices, see Security best practices in IAM in the *IAM User Guide*.	Following the instructions in Getting started in the *AWS IAM Identity Center User Guide*.	Configure programmatic access by Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide*.

Choose one way to manage your administrator	To	By	You can also
In IAM (Not recommended)	Use long-term credentials to access AWS.	Following the instructions in Creating your first IAM admin user and user group in the *IAM User Guide*.	Configure programmatic access by Managing access keys for IAM users in the *IAM User Guide*.

Step 3: Create an access key ID and secret access key

For CLI access, you need an access key ID and a secret access key. Use temporary credentials instead of long-term access keys when possible. Temporary credentials include an access key ID, a secret access key, and a security token that indicates when the credentials expire. For more information, see Using temporary credentials with AWS resources in the *IAM User Guide*.

Users need programmatic access if they want to interact with AWS outside of the AWS Management Console. The way to grant programmatic access depends on the type of user that's accessing AWS.

To grant users programmatic access, choose one of the following options.

Which user needs programmatic access?	To	By
Workforce identity (Users managed in IAM Identity Center)	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS*

Which user needs programmatic access?	To	By
		Command Line Interface User Guide. • For AWS SDKs, tools, and AWS APIs, see IAM Identity Center authentication in the *AWS SDKs and Tools Reference Guide*.
IAM	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions in Using temporary credentials with AWS resources in the *IAM User Guide*.
IAM	(Not recommended) Use long-term credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Authenticating using IAM user credentials in the *AWS Command Line Interface User Guide*. • For AWS SDKs and tools, see Authenticate using long-term credentials in the *AWS SDKs and Tools Reference Guide*. • For AWS APIs, see Managing access keys for IAM users in the *IAM User Guide*.

Step 4: Install the AWS CLI

The AWS CLI is an open source tool that enables you to interact with AWS services using commands in your command-line shell. The AWS SAM CLI requires the AWS CLI for activities such as configuring credentials. To learn more about the AWS CLI, see What is the AWS Command Line Interface? in the *AWS Command Line Interface User Guide*.

To install the AWS CLI, see Installing or updating the latest version of the AWS CLI in the *AWS Command Line Interface User Guide*.

Step 5: Use the AWS CLI to configure AWS credentials

To configure credentials with the AWS CLI

1. Run the `aws configure` command from the command line.
2. Configure the following. Select each link to learn more:

    a.   Access key ID
    b.   Secret access key
    c.   AWS Region
    d.   Output format

    The following example shows sample values.

    ```
    $ aws configure
    AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
    AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
    Default region name [None]: us-west-2
    Default output format [None]: json
    ```

The AWS CLI stores this information in a *profile* (a collection of settings) named `default` in the `credentials` and `config` files. These files are located in the `.aws` file in your home directory. By default, the information in this profile is used when you run an AWS CLI command that doesn't explicitly specify a profile to use. For more information on the `credentials` file, see Configuration and credential file settings in the *AWS Command Line Interface User Guide*.

For more information on configuring credentials, such as using an existing configuration and credentials file, see Quick setup in the *AWS Command Line Interface User Guide*.

Next steps

You are now ready to install the AWS SAM CLI and start using AWS SAM. To install the AWS SAM CLI, see Installing the AWS SAM CLI.

Installing the AWS SAM CLI

Install the latest release of the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) on supported operating systems.

For information on managing a currently installed version of the AWS SAM CLI, including how to upgrade, uninstall, or manage nightly builds, see Managing AWS SAM CLI versions.

> ⓘ **Is this your first time installing the AWS SAM CLI?**
>
> Complete all prerequisites in the previous section before moving forward. This includes:
>
> 1. Signing up for an AWS account.
> 2. Creating an administrative IAM user.
> 3. Creating an access key ID and secret access key.
> 4. Installing the AWS CLI.
> 5. Configuring AWS credentials.

Topics

- Installing the AWS SAM CLI
- Troubleshooting
- Next steps

Installing the AWS SAM CLI

> ⓘ **Note**
>
> Starting September 2023, AWS will no longer maintain the AWS managed Homebrew installer for the AWS SAM CLI (`aws/tap/aws-sam-cli`). If you use Homebrew to install and manage the AWS SAM CLI, see the following options:

- To continue using Homebrew, you can use the community managed installer. For more information, see Managing the AWS SAM CLI with Homebrew.

- We recommend using one of the first party installation methods that are documented on this page. Before using one of these methods, see Switch from Homebrew.

To install the AWS SAM CLI, follow the instructions for your operating system.

Linux

arm64 - command line installer

1. Download the AWS SAM CLI .zip file to a directory of your choice.

2. **(Optional)** You can verify the integrity of the installer before installation. For instructions, see Verify the integrity of the AWS SAM CLI installer.

3. Unzip the installation files into a directory of your choice. The following is an example, using the `sam-installation` subdirectory.

> ℹ️ **Note**
>
> If your operating system doesn't have the built-in **unzip** command, use an equivalent.

```
$ unzip aws-sam-cli-linux-arm64.zip -d sam-installation
```

4. Install the AWS SAM CLI by running the `install` executable. This executable is located in the directory used in the previous step. The following is an example, using the `sam-installation` subdirectory:

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

5. Verify the installation.

```
$ sam --version
```

To confirm a successful installation, you should see an output like the following but that replaces the bracketed text with the latest SAM CLI version:

```
SAM CLI, <latest version>
```

x86_64 - command line installer

1. Download the AWS SAM CLI .zip file to a directory of your choice.

2. **(Optional)** You can verify the integrity of the installer before installation. For instructions, see Verify the integrity of the AWS SAM CLI installer.

3. Unzip the installation files into a directory of your choice. The following is an example, using the `sam-installation` subdirectory.

   > ⓘ **Note**
   >
   > If your operating system doesn't have the built-in **unzip** command, use an equivalent.

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

4. Install the AWS SAM CLI by running the `install` executable. This executable is located in the directory used in the previous step. The following is an example, using the `sam-installation` subdirectory:

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

5. Verify the installation.

   ```
   $ sam --version
   ```

   To confirm a successful installation, you should see an output that replaces the following bracketed text with the latest available version:

   ```
   SAM CLI, <latest version>
   ```

macOS

Install the AWS SAM CLI using its package installer.

> **ⓘ Note**
>
> The package installer is available in x86_64 and arm64 options. Choose the package that matches your Mac hardware:
>
> - **x86_64** – Use for Macs running Intel processors.
> - **arm64** – Use for Macs running Apple silicon.
>
> For more information, see [Mac computers with Apple silicon](#).

Using the package installer

The package installer has two installation methods that you can choose from:

1. GUI
2. Command line

You can install for all users or just your current user. To install for all users, superuser authorization is required.

Installation steps

Install the AWS SAM CLI using any of the following options.

GUI - All users

To download the package installer

> **ⓘ Note**
>
> If you previously installed the AWS SAM CLI through Homebrew or pip, you need to uninstall it first. For instructions, see [Uninstalling the AWS SAM CLI](#).

- To begin installation, download the macOS pkg to a directory of your choice:

- x86_64 – [aws-sam-cli-macos-x86_64.pkg](#)

- arm64 – [aws-sam-cli-macos-arm64.pkg](#)

To verify the package installer (optional)

- You can verify the integrity of the installer before installation. For instructions, see [Verify the integrity of the AWS SAM CLI installer](#).

To install the AWS SAM CLI

1. Run your downloaded file and follow the on-screen instructions to continue through the **Introduction**, **Read Me**, and **License** steps.

2. For **Destination Select**, select **Install for all users of this computer**.

3. For **Installation Type**, choose where the AWS SAM CLI will be installed and press **Install**. The recommended default location is `/usr/local/aws-sam-cli`.

   > ⓘ **Note**
   >
   > To invoke the AWS SAM CLI with the **sam** command, the installer automatically creates a symlink between `/usr/local/bin/sam` and either `/usr/local/aws-sam-cli/sam` or the installation folder you chose.

4. The AWS SAM CLI will install and **The installation was successful** message will display. Press **Close**.

To verify a successful installation

- Verify that the AWS SAM CLI has properly installed and that your symlink is configured by running:

```
$ which sam
/usr/local/bin/sam
$ sam --version
SAM CLI, <latest version>
```

GUI - Current user

To download the package installer

> **ⓘ Note**
>
> If you previously installed the AWS SAM CLI through Homebrew or pip, you need to uninstall it first. For instructions, see Uninstalling the AWS SAM CLI.

- To begin installation, download the macOS `pkg` to a directory of your choice:

  - x86_64 – aws-sam-cli-macos-x86_64.pkg
  - arm64 – aws-sam-cli-macos-arm64.pkg

To verify the package installer (optional)

- You can verify the integrity of the installer before installation. For instructions, see Verify the integrity of the AWS SAM CLI installer.

To install the AWS SAM CLI

1. Run your downloaded file and follow the on-screen instructions to continue through the **Introduction**, **Read Me**, and **License** steps.

2. For **Destination Select**, select **Install for me only**. If you don't see this option, go to the next step.

3. For **Installation Type**, do the following:

   1. Choose where the AWS SAM CLI will be installed. The default location is `/usr/local/aws-sam-cli`. Select a location that you have write permissions for. To change the installation location, select **local** and choose your location. Press **Continue** when done.

   2. If you didn't get the option to choose **Install for me only** in the previous step, select **Change Install Location** > **Install for me only** and press **Continue**.

   3. Press **Install**.

4. The AWS SAM CLI will install and **The installation was successful** message will display. Press **Close**.

To create a symlink

- To invoke the AWS SAM CLI with the **sam** command, you must manually create a symlink between the AWS SAM CLI program and your $PATH. Create your symlink by modifying and running the following command:

  ```
  $ sudo ln -s /path-to/aws-sam-cli/sam /path-to-symlink-directory/sam
  ```

  - *sudo* – If your user has write permissions to $PATH, **sudo** is not required. Otherwise, **sudo** is required.
  - *path-to* – Path to where you installed the AWS SAM CLI program. For example, /Users/myUser/Desktop.
  - *path-to-symlink-directory* – Your $PATH environment variable. The default location is /usr/local/bin.

To verify a successful installation

- Verify that the AWS SAM CLI has properly installed and that your symlink is configured by running:

  ```
  $ which sam
  /usr/local/bin/sam
  $ sam --version
  SAM CLI, <latest version>
  ```

Command line - All users

To download the package installer

> ⓘ **Note**
>
> If you previously installed the AWS SAM CLI through Homebrew or pip, you need to uninstall it first. For instructions, see [Uninstalling the AWS SAM CLI](#).

- To begin installation, download the macOS pkg to a directory of your choice:

- x86_64 – [aws-sam-cli-macos-x86_64.pkg](#)

- arm64 – [aws-sam-cli-macos-arm64.pkg](#)

To verify the package installer (optional)

- You can verify the integrity of the installer before installation. For instructions, see [Verify the integrity of the AWS SAM CLI installer](#).

To install the AWS SAM CLI

- Modify and run the installation script:

```
$ sudo installer -pkg path-to-pkg-installer/name-of-pkg-installer -target /
installer: Package name is AWS SAM CLI
installer: Upgrading at base path /
installer: The upgrade was successful.
```

> ⓘ **Note**
>
> To invoke the AWS SAM CLI with the **sam** command, the installer automatically creates a symlink between `/usr/local/bin/sam` and `/usr/local/aws-sam-cli/sam`.

To verify a successful installation

- Verify that the AWS SAM CLI has properly installed and that your symlink is configured by running:

```
$ which sam
/usr/local/bin/sam
$ sam --version
SAM CLI, <latest version>
```

Command line - Current user

To download the package installer

> **ⓘ Note**
>
> If you previously installed the AWS SAM CLI through Homebrew or pip, you need to uninstall it first. For instructions, see Uninstalling the AWS SAM CLI.

- To begin installation, download the macOS `pkg` to a directory of your choice:

  - x86_64 – aws-sam-cli-macos-x86_64.pkg
  - arm64 – aws-sam-cli-macos-arm64.pkg

To verify the package installer (optional)

- You can verify the integrity of the installer before installation. For instructions, see Verify the integrity of the AWS SAM CLI installer.

To install the AWS SAM CLI

1. Determine an installation directory that you have write permissions to. Then, create an `xml` file using the template and modify it to reflect your installation directory. The directory must already exist.

   For example, if you replace *path-to-my-directory* with /Users/myUser/Desktop, the `aws-sam-cli` program folder will be installed there.

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/
   PropertyList-1.0.dtd">
   <plist version="1.0">
     <array>
       <dict>
         <key>choiceAttribute</key>
         <string>customLocation</string>
         <key>attributeSetting</key>
         <string>path-to-my-directory</string>
         <key>choiceIdentifier</key>
   ```

```
            <string>default</string>
        </dict>
    </array>
</plist>
```

2. Save the `xml` file and verify that its valid by running the following:

```
$ installer -pkg path-to-pkg-installer \
-target CurrentUserHomeDirectory \
-showChoicesAfterApplyingChangesXML path-to-your-xml-file
```

The output should display the preferences that will be applied to the AWS SAM CLI program.

3. Run the following to install the AWS SAM CLI:

```
$ installer -pkg path-to-pkg-installer \
-target CurrentUserHomeDirectory \
-applyChoiceChangesXML path-to-your-xml-file

# Example output
installer: Package name is AWS SAM CLI
installer: choices changes file 'path-to-your-xml-file' applied
installer: Upgrading at base path base-path-of-xml-file
installer: The upgrade was successful.
```

To create a symlink

*   To invoke the AWS SAM CLI with the **sam** command, you must manually create a symlink between the AWS SAM CLI program and your $PATH. Create your symlink by modifying and running the following command:

```
$ sudo ln -s /path-to/aws-sam-cli/sam /path-to-symlink-directory/sam
```

    *   *sudo* – If your user has write permissions to $PATH, **sudo** is not required. Otherwise, **sudo** is required.

    *   *path-to* – Path to where you installed the AWS SAM CLI program. For example, `/Users/myUser/Desktop`.

- *path-to-symlink-directory* – Your $PATH environment variable. The default location is /usr/local/bin.

To verify a successful installation

- Verify that the AWS SAM CLI has properly installed and that your symlink is configured by running:

```
$ which sam
/usr/local/bin/sam
$ sam --version
SAM CLI, <latest version>
```

Windows

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. Download the AWS SAM CLI [64-bit](#).

> ⓘ **Note**
>
> If you use a 32-bit version of Windows, see [Installing the AWS SAM CLI on 32-bit Windows](#).

2. **(Optional)** You can verify the integrity of the installer before installation. For instructions, see [Verify the integrity of the AWS SAM CLI installer](#).

3. 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.

   ```
   sam --version
   ```

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

   ```
   SAM CLI, <latest version>
   ```

4. Enable long paths (Windows 10 and newer only).

> ⚠️ **Important**
>
> The AWS SAM CLI might interact with filepaths that exceed the Windows max path limitation. This may cause errors when running `sam init` due to Windows 10 **MAX_PATH** limitations. To resolve this issue, the new long paths behavior must be configured.

To enable long paths, see [Enable Long Paths in Windows 10, Version 1607, and Later](#) in the *Microsoft Windows App Development Documentation*.

5. Install Git.

To download sample applications using the `sam init` command, you must also install Git. For instructions, see [Installing Git](#).

Troubleshooting

If you come across issues while installing the AWS SAM CLI, see [Installation errors](#).

Next steps

To learn more about the AWS SAM CLI and to begin building your own serverless applications, see the following:

- [Tutorial: Deploying a Hello World application](#) – Step-by-step instructions to download, build, and deploy a basic serverless application.
- [The Complete AWS SAM Workshop](#) – A workshop designed to teach you many of the major features that AWS SAM provides.
- [AWS SAM example applications and patterns](#) – Sample applications and patterns from community authors that you can further experiment with.

Tutorial: Deploying a Hello World application

In this tutorial, you use the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) to complete the following:

- Initialize, build, and deploy a sample **Hello World** application.

- Make local changes and sync to AWS CloudFormation.

- Test your application in the AWS Cloud.

- Optionally, perform local testing on your development host.

- Delete the sample application from the AWS Cloud.

The sample **Hello World** application implements a basic API backend. It consists of the following resources:

- **Amazon API Gateway** – API endpoint that you will use to invoke your function.

- **AWS Lambda** – Function that processes the HTTP API GET request and returns a `hello world` message.

- **AWS Identity and Access Management (IAM) role** – Provisions permissions for the services to securely interact.

The following diagram shows the components of this application:
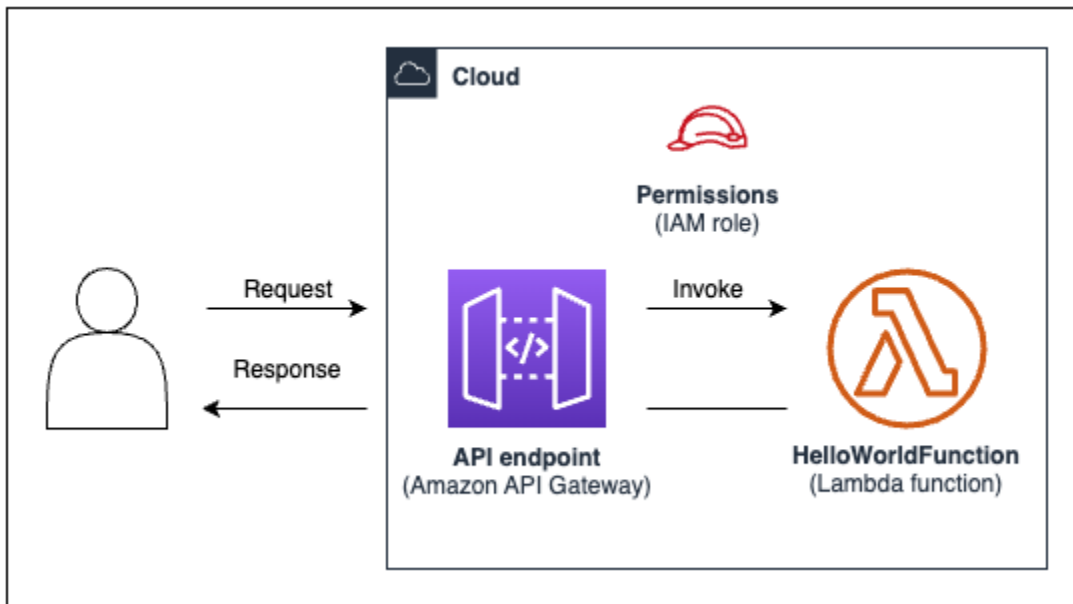
Topics

- [Prerequisites](#)
- [Step 1: Initialize the sample Hello World application](#)
- [Step 2: Build your application](#)

- Step 3: Deploy your application to the AWS Cloud

- Step 4: Run your application

- Step 5: Interact with your function in the AWS Cloud

- Step 6: Modify and sync your application to the AWS Cloud

- Step 7: (Optional) Test your application locally

- Step 8: Delete your application from the AWS Cloud

- Troubleshooting

- Learn more

Prerequisites

Verify that you have completed the following:

- AWS SAM prerequisites

- Installing the AWS SAM CLI

Step 1: Initialize the sample Hello World application

In this step, you will use the AWS SAM CLI to create a sample **Hello World** application project on your local machine.

To initialize the sample Hello World application

1. In your command line, run the following from a starting directory of your choice:

    ```
    $ sam init
    ```

2. The AWS SAM CLI will guide you through initializing a new application. Configure the following:

    1. Select **AWS Quick Start Templates** to choose a starting template.

    2. Choose the **Hello World Example** template and download it.

    3. Use the Python runtime and `zip` package type.

    4. For this tutorial, opt out of AWS X-Ray tracing. To learn more, see What is AWS X-Ray? in the
        AWS X-Ray Developer Guide.

5. For this tutorial, opt out of monitoring with Amazon CloudWatch Application Insights. To learn more, see Amazon CloudWatch Application Insights in the *Amazon CloudWatch User Guide*.

6. Name your application as **sam-app**.

To use the AWS SAM CLI interactive flow:

- Brackets ([ ]) indicate default values. Leave your answer blank to select the default value.

- Enter **y** for **yes**, and **n** for **no**.

The following is an example of the `sam init` interactive flow:

```
$ sam init
...
Which template source would you like to use?
    1 - AWS Quick Start Templates
    2 - Custom Template Location
Choice: 1

Choose an AWS Quick Start application template
    1 - Hello World Example
    2 - Multi-step workflow
    3 - Serverless API
    4 - Scheduled task
    5 - Standalone function
    6 - Data processing
    7 - Hello World Example With Powertools
    8 - Infrastructure event management
    9 - Serverless Connector Hello World Example
    10 - Multi-step workflow with Connectors
    11 - Lambda EFS example
    12 - DynamoDB Example
    13 - Machine Learning
Template: 1

Use the most popular runtime and package type? (Python and zip) [y/N]: y

Would you like to enable X-Ray tracing on the function(s) in your application?  [y/
N]: ENTER
```

```
Would you like to enable monitoring using CloudWatch Application Insights?
For more info, please view https://docs.aws.amazon.com/AmazonCloudWatch/latest/
monitoring/cloudwatch-application-insights.html [y/N]: ENTER

Project name [sam-app]: ENTER
```

3. The AWS SAM CLI downloads your starting template and creates the application project directory structure on your local machine. The following is an example of the AWS SAM CLI output:

```
Cloning from https://github.com/aws/aws-sam-cli-app-templates (process may take a
 moment)

    -----------------------
    Generating application:
    -----------------------
    Name: sam-app
    Runtime: python3.9
    Architectures: x86_64
    Dependency Manager: pip
    Application Template: hello-world
    Output Directory: .
    Configuration file: sam-app/samconfig.toml

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


Commands you can use next
=========================
[*] Create pipeline: cd sam-app && sam pipeline init --bootstrap
[*] Validate SAM template: cd sam-app && sam validate
[*] Test Function in the Cloud: cd sam-app && sam sync --stack-name {stack-name} --
watch
```

4. From your command line, move to the newly created sam-app directory. The following is an example of what the AWS SAM CLI has created:

```
$ cd sam-app

$ tree

### README.md
### __init__.py
```

```
### events
#    ### event.json
### hello_world
#    ### __init__.py
#    ### app.py
#    ### requirements.txt
### samconfig.toml
### template.yaml
### tests
    ### __init__.py
    ### integration
    #    ### __init__.py
    #    ### test_api_gateway.py
    ### requirements.txt
    ### unit
        ### __init__.py
        ### test_handler.py

6 directories, 14 files
```

Some important files to highlight:

- `hello_world/app.py` – Contains your Lambda function code.

- `hello_world/requirements.txt` – Contains any Python dependencies that your
  Lambda function requires.

- `samconfig.toml` – Configuration file for your application that stores default parameters
  used by the AWS SAM CLI.

- `template.yaml` – The AWS SAM template that contains your application infrastructure
  code.

You now have a completely authored serverless application on your local machine!

Step 2: Build your application

In this step, you use the AWS SAM CLI to build your application and prepare for deployment.
When you build, the AWS SAM CLI creates a `.aws-sam` directory and organizes your function
dependencies, project code, and project files there.

To build your application

- In your command line, from the `sam-app` project directory, run the following:

```
$ sam build
```

> **ⓘ Note**
>
> If you don't have Python on your local machine, use the **sam build --use-container** command instead. The AWS SAM CLI will create a Docker container that includes your function's runtime and dependencies. This command requires Docker on your local machine. To install Docker, see [Installing Docker](#).

The following is an example of the AWS SAM CLI output:

```
$ sam build
Starting Build use cache
Manifest file is changed (new hash: 3298f1304...d4d421) or dependency folder (.aws-
sam/deps/4d3dfad6-a267-47a6-a6cd-e07d6fae318c) is missing for (HelloWorldFunction),
 downloading dependencies and copying/building source
Building codeuri: /Users/.../Demo/sam-tutorial1/sam-app/hello_world runtime:
 python3.9 metadata: {} architecture: x86_64 functions: HelloWorldFunction
Running PythonPipBuilder:CleanUp
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource
Running PythonPipBuilder:CopySource

Build Succeeded

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

Commands you can use next
=========================
[*] Validate SAM template: sam validate
[*] Invoke Function: sam local invoke
[*] Test Function in the Cloud: sam sync --stack-name {{stack-name}} --watch
[*] Deploy: sam deploy --guided
```

The following is a shortened example of the `.aws-sam` directory created by the AWS SAM CLI:

```
.aws-sam
### build
#   ### HelloWorldFunction
#   #   ### __init__.py
#   #   ### app.py
#   #   ### requirements.txt
#   ### template.yaml
### build.toml
```

Some important files to highlight:

- `build/HelloWorldFunction` – Contains your Lambda function code and dependencies. The AWS SAM CLI creates a directory for each function in your application.
- `build/template.yaml` – Contains a copy of your AWS SAM template that is referenced by AWS CloudFormation at deployment.
- `build.toml` – Configuration file that stores default parameter values referenced by the AWS SAM CLI when building and deploying your application.

You are now ready to deploy your application to the AWS Cloud.

Step 3: Deploy your application to the AWS Cloud

> ⓘ **Note**
>
> This step requires AWS credentials configuration. For more information, see Step 5: Use the AWS CLI to configure AWS credentials in AWS SAM prerequisites.

In this step, you use the AWS SAM CLI to deploy your application to the AWS Cloud. The AWS SAM CLI will do the following:

- Guide you through configuring your application settings for deployment.
- Upload your application files to Amazon Simple Storage Service (Amazon S3).
- Transform your AWS SAM template into an AWS CloudFormation template. It then uploads your template to the AWS CloudFormation service to provision your AWS resources.

To deploy your application

1. In your command line, from the `sam-app` project directory, run the following:

    ```
    $ sam deploy --guided
    ```

2. Follow the AWS SAM CLI interactive flow to configure your application settings. Configure the following:

    1. The **AWS CloudFormation stack name** – A stack is a collection of AWS resources that you can manage as a single unit. To learn more, see Working with stacks in the *AWS CloudFormation User Guide*.

    2. The **AWS Region** to deploy your AWS CloudFormation stack to. For more information, see AWS CloudFormation endpoints in the *AWS CloudFormation User Guide*.

    3. For this tutorial, opt out of **confirming changes before deploy**.

    4. Allow **IAM role creation** – This lets AWS SAM create the IAM role necessary for your API Gateway resource and Lambda function resource to interact.

    5. For this tutorial, opt out of **disabling rollback**.

    6. Allow **HelloWorldFunction without authorization defined** – This message displays because your API Gateway endpoint is configured to be publicly accessible, without authorization. Since this is the intended configuration for your Hello World application, allow the AWS SAM CLI to continue. For more information about configuring authorization, see Controlling access to API Gateway APIs.

    7. **Save arguments to configuration file** – This will update your application's `samconfig.toml` file with your deployment preferences.

    8. Select the default **configuration file name**.

    9. Select the default **configuration environment**.

    The following is an example output of the `sam deploy --guided` interactive flow:

    ```
    $ sam-app sam deploy --guided

    Configuring SAM deploy
    ======================

        Looking for config file [samconfig.toml] :  Found
        Reading default arguments  :  Success
    ```

```
    Setting default arguments for 'sam deploy'
    =========================================
    Stack Name [sam-app]: ENTER
    AWS Region [us-west-2]: ENTER
    #Shows you resources changes to be deployed and require a 'Y' to initiate
 deploy
    Confirm changes before deploy [Y/n]: n
    #SAM needs permission to be able to create roles to connect to the resources in
 your template
    Allow SAM CLI IAM role creation [Y/n]: ENTER
    #Preserves the state of previously provisioned resources when an operation
 fails
    Disable rollback [y/N]: ENTER
    HelloWorldFunction may not have authorization defined, Is this okay? [y/N]: y
    Save arguments to configuration file [Y/n]: ENTER
    SAM configuration file [samconfig.toml]: ENTER
    SAM configuration environment [default]: ENTER
```

3.   The AWS SAM CLI deploys your application by doing the following:

- The AWS SAM CLI creates an Amazon S3 bucket and uploads your `.aws-sam` directory.

- The AWS SAM CLI transforms your AWS SAM template into AWS CloudFormation and uploads it to the AWS CloudFormation service.

- AWS CloudFormation provisions your resources.

During deployment, the AWS SAM CLI displays your progress. The following is an example output:

```
Looking for resources needed for deployment:

    Managed S3 bucket: aws-sam-cli-managed-default-samclisourcebucket-1a4x26zbcdkqr
    A different default S3 bucket can be set in samconfig.toml

    Parameter "stack_name=sam-app" in [default.deploy.parameters] is defined as a
 global parameter [default.global.parameters].
    This parameter will be only saved under [default.global.parameters] in /
Users/.../Demo/sam-tutorial1/sam-app/samconfig.toml.

    Saved arguments to config file
    Running 'sam deploy' for future deployments will use the parameters saved
 above.
```

```
    The above parameters can be changed by modifying samconfig.toml
    Learn more about samconfig.toml syntax at
    https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/
serverless-sam-cli-config.html


File with same data already exists at sam-app/da3c598813f1c2151579b73ad788cac8,
 skipping upload


    Deploying with following values
    ===============================
    Stack name                 : sam-app
    Region                     : us-west-2
    Confirm changeset          : False
    Disable rollback           : False
    Deployment s3 bucket       : aws-sam-cli-managed-default-
samclisourcebucket-1a4x26zbcdkqr
    Capabilities               : ["CAPABILITY_IAM"]
    Parameter overrides        : {}
    Signing Profiles           : {}


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


File with same data already exists at sam-
app/2bebf67c79f6a743cc5312f6dfc1efee.template, skipping upload



Waiting for changeset to be created..

CloudFormation stack changeset
-------------------------------------------------------------------------------------
Operation                          LogicalResourceId
 ResourceType                       Replacement
-------------------------------------------------------------------------------------
* Modify                           HelloWorldFunction
 AWS::Lambda::Function               False
* Modify                           ServerlessRestApi
 AWS::ApiGateway::RestApi            False
- Delete                           AwsSamAutoDependencyLayerNestedSt
 AWS::CloudFormation::Stack          N/A
                                    ack
-------------------------------------------------------------------------------------
```

```
Changeset created successfully. arn:aws:cloudformation:us-
west-2:012345678910:changeSet/samcli-deploy1678917603/22e05525-08f9-4c52-a2c4-
f7f1fd055072


2023-03-15 12:00:16 - Waiting for stack create/update to complete

CloudFormation events from stack operations (refresh every 0.5 seconds)
-------------------------------------------------------------------------------
ResourceStatus                      ResourceType
 LogicalResourceId                    ResourceStatusReason
-------------------------------------------------------------------------------
UPDATE_IN_PROGRESS                  AWS::Lambda::Function
 HelloWorldFunction                   -
UPDATE_COMPLETE                     AWS::Lambda::Function
 HelloWorldFunction                   -
UPDATE_COMPLETE_CLEANUP_IN_PROGRE   AWS::CloudFormation::Stack        sam-app
                          -
SS
DELETE_IN_PROGRESS                  AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt   -
                                                          ack
DELETE_COMPLETE                     AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt   -
                                                          ack
UPDATE_COMPLETE                     AWS::CloudFormation::Stack        sam-app
                          -
-------------------------------------------------------------------------------

CloudFormation outputs from deployed stack
-------------------------------------------------------------------------------
Outputs
-------------------------------------------------------------------------------
Key                 HelloWorldFunctionIamRole
Description         Implicit IAM Role created for Hello World function
Value               arn:aws:iam::012345678910:role/sam-app-
HelloWorldFunctionRole-15GLOUR9LMT1W

Key                 HelloWorldApi
Description         API Gateway endpoint URL for Prod stage for Hello World
 function
Value               https://<restapiid>.execute-api.us-west-2.amazonaws.com/Prod/
hello/
```

```
Key                   HelloWorldFunction
Description           Hello World Lambda Function ARN
Value                arn:aws:lambda:us-west-2:012345678910:function:sam-app-
HelloWorldFunction-yQDNe17r9maD
-------------------------------------------------------------------------------



Successfully created/updated stack - sam-app in us-west-2
```

Your application is now deployed and running in the AWS Cloud!

Step 4: Run your application

In this step, you will send a GET request to your API endpoint and see your Lambda function output.

To get your API endpoint value

1. From the information displayed by the AWS SAM CLI in the previous step, locate the `Outputs` section. In this section, locate your `HelloWorldApi` resource to find your HTTP endpoint value. The following is an example output:

   ```
   -------------------------------------------------------------------------------
   Outputs
   -------------------------------------------------------------------------------
   ...
   Key                   HelloWorldApi
   Description           API Gateway endpoint URL for Prod stage for Hello World
    function
   Value                https://ets1gv8lxi.execute-api.us-west-2.amazonaws.com/Prod/
   hello/
   ...
   -------------------------------------------------------------------------------
   ```

2. Alternatively, you can use the **sam list endpoints --output json** command to get this information. The following is an example output:

   ```
   $ sam list endpoints --output json
   2023-03-15 12:39:19 Loading policies from IAM...
   2023-03-15 12:39:25 Finished loading policies from IAM.
   [
   ```

```
    {
      "LogicalResourceId": "HelloWorldFunction",
      "PhysicalResourceId": "sam-app-HelloWorldFunction-yQDNe17r9maD",
      "CloudEndpoint": "-",
      "Methods": "-"
    },
    {
      "LogicalResourceId": "ServerlessRestApi",
      "PhysicalResourceId": "ets1gv8lxi",
      "CloudEndpoint": [
        "https://ets1gv8lxi.execute-api.us-west-2.amazonaws.com/Prod",
        "https://ets1gv8lxi.execute-api.us-west-2.amazonaws.com/Stage"
      ],
      "Methods": [
        "/hello['get']"
      ]
    }
  ]
```

To invoke your function

- Using your browser or the command line, send a GET request to your API endpoint. The
  following is an example using the curl command:

  ```
  $ curl https://ets1gv8lxi.execute-api.us-west-2.amazonaws.com/Prod/hello/
  {"message": "hello world"}
  ```

Step 5: Interact with your function in the AWS Cloud

In this step, you use the AWS SAM CLI to invoke your Lambda function in the AWS Cloud.

To invoke your Lambda function in the cloud

1. Take note of your function's LogicalResourceId from the previous step. It should be
   HelloWorldFunction.

2. In your command line, from the sam-app project directory, run the following:

   ```
   $ sam remote invoke HelloWorldFunction --stack-name sam-app
   ```

3. The AWS SAM CLI invokes your function in the cloud and returns a response. The following is an example output:

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app

Invoking Lambda Function HelloWorldFunction
START RequestId: d5ef494b-5f45-4086-86fd-d7322fa1a1f9 Version: $LATEST
END RequestId: d5ef494b-5f45-4086-86fd-d7322fa1a1f9
REPORT RequestId: d5ef494b-5f45-4086-86fd-d7322fa1a1f9  Duration: 6.62 ms
 Billed Duration: 7 ms      Memory Size: 128 MB     Max Memory Used: 67 MB  Init
 Duration: 164.06 ms
{"statusCode":200,"body":"{\"message\":\"hello world\"}"}%
```

Step 6: Modify and sync your application to the AWS Cloud

In this step, you use the AWS SAM CLI **sam sync --watch** command to sync local changes to the AWS Cloud.

To use sam sync

1. In your command line, from the `sam-app` project directory, run the following:

```
$ sam sync --watch
```

2. The AWS SAM CLI prompts you to confirm that you are syncing a development stack. Since the **sam sync --watch** command automatically deploys local changes to the AWS Cloud in real time, we recommend it for development environments only.

   The AWS SAM CLI performs an initial deployment before it begins monitoring for local changes. The following is an example output:

```
$ sam sync --watch
The SAM CLI will use the AWS Lambda, Amazon API Gateway, and AWS StepFunctions APIs
 to upload your code without
performing a CloudFormation deployment. This will cause drift in your
 CloudFormation stack.
**The sync command should only be used against a development stack**.

Confirm that you are synchronizing a development stack.
```

```
Enter Y to proceed with the command, or enter N to cancel:
  [Y/n]: y
Queued infra sync. Waiting for in progress code syncs to complete...
Starting infra sync.
Manifest is not changed for (HelloWorldFunction), running incremental build
Building codeuri: /Users/.../Demo/sam-tutorial1/sam-app/hello_world runtime:
 python3.9 metadata: {} architecture: x86_64 functions: HelloWorldFunction
Running PythonPipBuilder:CopySource


Build Succeeded


Successfully packaged artifacts and wrote output template to file /var/
folders/45/5ct135bx3fn2551_ptl5g6_80000gr/T/tmpq3x9vh63.
Execute the following command to deploy the packaged template
sam deploy --template-file /var/folders/45/5ct135bx3fn2551_ptl5g6_80000gr/T/
tmpq3x9vh63 --stack-name <YOUR STACK NAME>



    Deploying with following values
    ===============================
    Stack name                 : sam-app
    Region                     : us-west-2
    Disable rollback           : False
    Deployment s3 bucket       : aws-sam-cli-managed-default-
samclisourcebucket-1a4x26zbcdkqr
    Capabilities               : ["CAPABILITY_NAMED_IAM",
 "CAPABILITY_AUTO_EXPAND"]
    Parameter overrides        : {}
    Signing Profiles           : null


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



2023-03-15 13:10:05 - Waiting for stack create/update to complete


CloudFormation events from stack operations (refresh every 0.5 seconds)
-------------------------------------------------------------------------------
ResourceStatus                    ResourceType
 LogicalResourceId                   ResourceStatusReason
-------------------------------------------------------------------------------
UPDATE_IN_PROGRESS                AWS::CloudFormation::Stack         sam-app
                    Transformation succeeded
```

```
CREATE_IN_PROGRESS                      AWS::CloudFormation::Stack
  AwsSamAutoDependencyLayerNestedSt      -
                                                                       ack
CREATE_IN_PROGRESS                      AWS::CloudFormation::Stack
  AwsSamAutoDependencyLayerNestedSt      Resource creation Initiated
                                                                       ack
CREATE_COMPLETE                         AWS::CloudFormation::Stack
  AwsSamAutoDependencyLayerNestedSt      -
                                                                       ack
UPDATE_IN_PROGRESS                      AWS::Lambda::Function
  HelloWorldFunction                     -
UPDATE_COMPLETE                         AWS::Lambda::Function
  HelloWorldFunction                     -
UPDATE_COMPLETE_CLEANUP_IN_PROGRE       AWS::CloudFormation::Stack       sam-app
                               -
SS
UPDATE_COMPLETE                         AWS::CloudFormation::Stack       sam-app
                               -
---------------------------------------------------------------------------

CloudFormation outputs from deployed stack
---------------------------------------------------------------------------
Outputs
---------------------------------------------------------------------------
Key                 HelloWorldFunctionIamRole
Description         Implicit IAM Role created for Hello World function
Value               arn:aws:iam::012345678910:role/sam-app-
HelloWorldFunctionRole-15GLOUR9LMT1W

Key                 HelloWorldApi
Description         API Gateway endpoint URL for Prod stage for Hello World
  function
Value               https://ets1gv8lxi.execute-api.us-west-2.amazonaws.com/Prod/
hello/

Key                 HelloWorldFunction
Description         Hello World Lambda Function ARN
Value               arn:aws:lambda:us-west-2:012345678910:function:sam-app-
HelloWorldFunction-yQDNe17r9maD
---------------------------------------------------------------------------


Stack update succeeded. Sync infra completed.
```

```
Infra sync completed.
CodeTrigger not created as CodeUri or DefinitionUri is missing for
  ServerlessRestApi.
```

Next, you will modify your Lambda function code. The AWS SAM CLI will automatically detect this change and sync your application to the AWS Cloud.

To modify and sync your application

1. In your IDE of choice, open the `sam-app/hello_world/app.py` file.

2. Change the `message` and save your file. The following is an example:

```python
import json
...
def lambda_handler(event, context):

    ...
    return {
        "statusCode": 200,
        "body": json.dumps({
            "message": "hello everyone!",

            ...
        }),
    }
```

3. The AWS SAM CLI detects your change and syncs your application to the AWS Cloud. The following is an example output:

```
Syncing Lambda Function HelloWorldFunction...
Manifest is not changed for (HelloWorldFunction), running incremental build
Building codeuri: /Users/.../Demo/sam-tutorial1/sam-app/hello_world runtime:
  python3.9 metadata: {} architecture: x86_64 functions: HelloWorldFunction
Running PythonPipBuilder:CopySource
Finished syncing Lambda Function HelloWorldFunction.
```

4. To verify your change, send a GET request to your API endpoint again.

```
$ curl https://ets1gv8lxi.execute-api.us-west-2.amazonaws.com/Prod/hello/
{"message": "hello everyone!"}
```

Step 7: (Optional) Test your application locally

> **ⓘ Note**
>
> This step is optional since it requires Docker on your local machine.

> **⚠ Important**
>
> To use the AWS SAM CLI for local testing, you must have Docker installed and configured. For more information, see [Installing Docker](#).

In this step, you use the AWS SAM CLI **sam local** command to test your application locally. To accomplish this, the AWS SAM CLI creates a local environment using Docker. This local environment emulates the cloud-based execution environment of your Lambda function.

You will do the following:

1. Create a local environment for your Lambda function and invoke it.

2. Host your HTTP API endpoint locally and use it to invoke your Lambda function.

To invoke your Lambda function locally

1. In your command line, from the `sam-app` project directory, run the following:

   ```
   $ sam local invoke
   ```

2. The AWS SAM CLI creates a local Docker container and invokes your function. The following is an example output:

   ```
   $ sam local invoke
   Invoking app.lambda_handler (python3.9)
   Local image was not found.
   Removing rapid images for repo public.ecr.aws/sam/emulation-python3.9
   Building image...................
   Using local image: public.ecr.aws/lambda/python:3.9-rapid-x86_64.
   ```

```
Mounting /Users/.../Demo/sam-tutorial1/sam-app/.aws-sam/build/HelloWorldFunction
 as /var/task:ro,delegated inside runtime container
START RequestId: b046db01-2a00-415d-af97-35f3a02e9eb6 Version: $LATEST
END RequestId: b046db01-2a00-415d-af97-35f3a02e9eb6
REPORT RequestId: b046db01-2a00-415d-af97-35f3a02e9eb6    Init Duration: 1.01 ms
    Duration: 633.45 ms    Billed Duration: 634 ms    Memory Size: 128 MB    Max
 Memory Used: 128 MB
{"statusCode": 200, "body": "{\"message\": \"hello world\"}"}
```

To host your API locally

1. In your command line, from the `sam-app` project directory, run the following:

```
$ sam local start-api
```

2. The AWS SAM CLI creates a local Docker container for your Lambda function and creates a local HTTP server to simulate your API endpoint. The following is an example output:

```
$ sam local start-api
Initializing the lambda functions containers.
Local image is up-to-date
Using local image: public.ecr.aws/lambda/python:3.9-rapid-x86_64.

Mounting /Users/.../Demo/sam-tutorial1/sam-app/.aws-sam/build/HelloWorldFunction
 as /var/task:ro,delegated inside runtime container
Containers Initialization is done.
Mounting HelloWorldFunction at http://127.0.0.1:3000/hello [GET]
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. If you used sam build before running local
 commands, you will need to re-run sam build for the changes to be picked up. You
 only need to restart SAM CLI if you update your AWS SAM template
2023-03-15 14:25:21 WARNING: This is a development server. Do not use it in a
 production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:3000
2023-03-15 14:25:21 Press CTRL+C to quit
```

3. Using your browser or the command line, send a GET request to your local API endpoint. The following is an example using the curl command:

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

```
{"message": "hello world"}
```

Step 8: Delete your application from the AWS Cloud

In this step, you use the AWS SAM CLI **sam delete** command to delete your application from the AWS Cloud.

To delete your application from the AWS Cloud

1. In your command line, from the `sam-app` project directory, run the following:

```
$ sam delete
```

2. The AWS SAM CLI will ask you to confirm. Then, it will delete your application's Amazon S3 bucket and AWS CloudFormation stack. The following is an example output:

```
$ sam delete
    Are you sure you want to delete the stack sam-app in the region us-west-2 ? [y/
N]: y
    Are you sure you want to delete the folder sam-app in S3 which contains the
 artifacts? [y/N]: y
    - Deleting S3 object with key c6ce8fa8b5a97dd022ecd006536eb5a4
    - Deleting S3 object with key 5d513a459d062d644f3b7dd0c8b56a2a.template
    - Deleting S3 object with key sam-app/2bebf67c79f6a743cc5312f6dfc1efee.template
    - Deleting S3 object with key sam-app/6b208d0e42ad15d1cee77d967834784b.template
    - Deleting S3 object with key sam-app/da3c598813f1c2151579b73ad788cac8
    - Deleting S3 object with key sam-app/f798cdd93cee188a71d120f14a035b11
    - Deleting Cloudformation stack sam-app

Deleted successfully
```

Troubleshooting

To troubleshoot the AWS SAM CLI, see [AWS SAM CLI troubleshooting](#).

Learn more

To continue learning about AWS SAM, see the following resources:

- **The Complete AWS SAM Workshop** – A workshop designed to teach you many of the major features that AWS SAM provides.

- **Sessions with SAM** – Video series created by our AWS Serverless Developer Advocate team on using AWS SAM.

- **Serverless Land** – Site that brings together the latest information, blogs, videos, code, and learning resources for AWS serverless.

Using the AWS Serverless Application Model (AWS SAM)

Use the AWS Serverless Application Model (AWS SAM) to build and run your serverless applications.

For an introduction to AWS SAM, see What is AWS SAM?

For information on using AWS SAM templates, see AWS Serverless Application Model (AWS SAM) specification.

Topics

- Using the AWS SAM CLI

Using the AWS SAM CLI

Use the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) with AWS SAM templates and supported third-party integrations to build and run your serverless applications.

For an introduction to AWS SAM, see What is AWS SAM?

Topics

- How AWS SAM CLI commands are documented
- Configuring the AWS SAM CLI
- Using sam build
- Using sam deploy
- Using sam init
- Using sam local
- Using sam remote invoke
- Using sam remote test-event
- Using sam sync

How AWS SAM CLI commands are documented

AWS SAM CLI commands are documented using the following format:

- **Prompt** – The Linux prompt is documented by default and is displayed as ($). For commands that are Windows specific, (>) is used as the prompt. Do not include the prompt when you type commands.

- **Directory** – When commands must be executed from a specific directory, the directory name is shown before the prompt symbol.

- **User input** – Command text that you enter at the command line is formatted as **`user input`**.

- **Replaceable text** – Variable text, such as file names and parameters are formatted as *`replaceable text`*. In multiple-line commands or commands where specific keyboard input is required, keyboard input can also be shown as replaceable text. For example, *ENTER*.

- **Output** – Output returned as a response to the command is formatted as `computer output`.

The following `sam deploy` command and output is an example:

```
$ sam deploy --guided --template template.yaml

Configuring SAM deploy
======================

    Looking for config file [samconfig.toml] :  Found
    Reading default arguments  :  Success

    Setting default arguments for 'sam deploy'
    ========================================
    Stack Name [sam-app]: ENTER
    AWS Region [us-west-2]: ENTER
    #Shows you resources changes to be deployed and require a 'Y' to initiate deploy
    Confirm changes before deploy [y/N]: ENTER
    #SAM needs permission to be able to create roles to connect to the resources in
 your template
    Allow SAM CLI IAM role creation [Y/n]: ENTER
    #Preserves the state of previously provisioned resources when an operation fails
    Disable rollback [y/N]: ENTER
    HelloWorldFunction may not have authorization defined, Is this okay? [y/N]: y
    Save arguments to configuration file [Y/n]: ENTER
    SAM configuration file [samconfig.toml]: ENTER
    SAM configuration environment [default]: ENTER
```

1. `sam deploy --guided --template template.yaml` is the command you enter at the command line.

2. `sam deploy --guided --template` should be provided as is.

3. *template.yaml* can be replaced with your specific file name.

4. The output starts at `Configuring SAM deploy`.

5. In the output, *ENTER* and *y* indicate replaceable values that you provide.

Configuring the AWS SAM CLI

Configure credentials, basic settings, and project settings for the AWS Serverless Application Model Command Line Interface (AWS SAM CLI).

Topics

- [Configure credentials and basic settings](#)
- [Configure project settings](#)

Configure credentials and basic settings

Use the AWS Command Line Interface (AWS CLI) to configure basic settings such as AWS credentials, default region name, and default output format. Once configured, you can use these settings with the AWS SAM CLI. To learn more, see the following from the *AWS Command Line Interface User Guide*:

- [Configuration basics](#)
- [Configuration and credential file settings](#)
- [Named profiles for the AWS CLI](#)
- [Using an IAM Identity Center enabled named profile](#)

For quick setup instructions, see [Step 5: Use the AWS CLI to configure AWS credentials](#).

Configure project settings

You can specify project-specific settings, such as AWS SAM CLI command parameter values, in a configuration file to use with the AWS SAM CLI. For more information about this configuration file, see [AWS SAM CLI configuration file](#).

Using configuration files

Configuration files are structured by environment, command, and parameter value. For more information, see Configuration file basics.

To configure a new environment

1. Specify your new environment in your configuration file.

    The following is an example of specifying a new `prod` environment:

    TOML

    ```
    [prod.global.parameters]
    ```

    YAML

    ```
    prod:
      global:
        parameters:
    ```

2. Specify parameter values as key-value pairs in the parameters section of the configuration file.

    The following is an example of specifying your application's stack name for the `prod` environment.

    TOML

    ```
    [prod.global.parameters]
    stack_name = "prod-app"
    ```

    YAML

    ```
    prod:
      global:
        parameters:
          stack_name: prod-app
    ```

3. Use the `--config-env` option to specify the environment to use.

    The following is an example:

```
$ sam deploy --config-env "prod"
```

To configure parameter values

1. Specify the AWS SAM CLI command that you'd like to configure parameter values for. To configure parameter values for all AWS SAM CLI commands, use the `global` identifier.

   The following is an example of specifying parameter values for the `default` environment's `sam deploy` command:

   TOML

   ```
   [default.deploy.parameters]
   confirm_changeset = true
   ```

   YAML

   ```
   default:
     deploy:
       parameters:
         confirm_changeset: true
   ```

   The following is an example of specifying parameter values for all AWS SAM CLI commands in the `default` environment:

   TOML

   ```
   [default.global.parameters]
   stack_name = "sam-app"
   ```

   YAML

   ```
   default:
     global:
       parameters:
         stack_name: sam-app
   ```

2. You can also specify parameter values and modify your configuration file through the AWS
    SAM CLI interactive flow.

    The following is an example of the `sam deploy --guided` interactive flow:

    ```
    $ sam deploy --guided

    Configuring SAM deploy
    ======================

        Looking for config file [samconfig.toml] :  Found
        Reading default arguments  :  Success

        Setting default arguments for 'sam deploy'
        ==========================================
        Stack Name [sam-app]: ENTER
        AWS Region [us-west-2]: ENTER
        #Shows you resources changes to be deployed and require a 'Y' to initiate
    deploy
        Confirm changes before deploy [Y/n]: n
        #SAM needs permission to be able to create roles to connect to the resources in
    your template
        Allow SAM CLI IAM role creation [Y/n]: ENTER
        #Preserves the state of previously provisioned resources when an operation
    fails
        Disable rollback [y/N]: ENTER
        HelloWorldFunction may not have authorization defined, Is this okay? [y/N]: y
        Save arguments to configuration file [Y/n]: ENTER
        SAM configuration file [samconfig.toml]: ENTER
        SAM configuration environment [default]: ENTER
    ```

For more information, see [Creating and modifying configuration files](#).

Examples

Basic TOML example

The following is an example of a `samconfig.toml` configuration file:

```
...
version = 0.1
```

```
[default]
[default.global]
[default.global.parameters]
stack_name = "sam-app"

[default.build.parameters]
cached = true
parallel = true

[default.deploy.parameters]
capabilities = "CAPABILITY_IAM"
confirm_changeset = true
resolve_s3 = true

[default.sync.parameters]
watch = true

[default.local_start_api.parameters]
warm_containers = "EAGER"

[prod]
[prod.sync]
[prod.sync.parameters]
watch = false
```

Basic YAML example

The following is an example of a `samconfig.yaml` configuration file:

```
version 0.1
default:
  global:
    parameters:
      stack_name: sam-app
  build:
    parameters:
      cached: true
      parallel: true
  deploy:
    parameters:
      capabilities: CAPABILITY_IAM
      confirm_changeset: true
      resolve_s3: true
  sync:
```

```
      parameters:
          watch: true
    local_start_api:
        parameters:
          warm_containers: EAGER
 prod:
   sync:
      parameters:
          watch: false
```

Using sam build

Use the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam build` command to prepare your serverless application for subsequent steps in your development workflow, such as local testing or deploying to the AWS Cloud. This command creates a `.aws-sam` directory that structures your application in a format and location that `sam local` and `sam deploy` require.

- For an introduction to the AWS SAM CLI, see [What is the AWS SAM CLI?](#).
- For a list of `sam build` command options, see [sam build](#).
- For an example of using `sam build` during a typical development workflow, see [Step 2: Build your application](#).

> ⓘ **Note**
>
> Using `sam build` requires that you start with the basic components of a serverless application on your development machine. This includes an AWS SAM template, AWS Lambda function code, and any language-specific files and dependencies. To learn more, see [Using sam init](#).

Topics

- [Building applications with sam build](#)
- [Local testing and deployment](#)
- [Best practices](#)
- [Options for sam build](#)
- [Troubleshooting](#)

- [Examples](#)

- [Learn more](#)

Building applications with sam build

Before using `sam build`, consider and configure the following:

1. **Lambda functions and layers** – The `sam build` command can build Lambda functions and layers. To learn more about Lambda layers, see [Building layers](#).

2. **Lambda runtime** – The *runtime* provides a language-specific environment that runs your function in an execution environment when invoked. You can configure native and custom runtimes.

   a. **Native runtime** – Author your Lambda functions in a supported Lambda runtime and build you functions to use a native Lambda runtime in the AWS Cloud.

   b. **Custom runtime** – Author your Lambda functions using any programming language and build your runtime using a custom process defined in a makefile or third-party builder such as esbuild. To learn more, see [Building custom runtimes](#).

3. **Lambda package type** – Lambda functions can be packaged in the following Lambda deployment package types:

   a. **.zip file archive** – Contains your application code and its dependencies.

   b. **Container image** – Contains the base operating system, the runtime, Lambda extensions, your application code and its dependencies.

These application settings can be configured when initializing an application using `sam init`.

- To learn more about using `sam init`, see [Using sam init](#).

- To learn more about configuring these settings in your application, see [Building applications](#).

To build an application

1. cd to the root of your project. This is the same location as your AWS SAM template.

   ```
   $ cd sam-app
   ```

2. Run the following:

```
sam-app $ sam build <arguments> <options>
```

> **ⓘ Note**
>
> A commonly used option is `--use-container`. To learn more, see Building a Lambda function inside of a provided container.

The following is an example of the AWS SAM CLI output:

```
sam-app $ sam build
Starting Build use cache
Manifest file is changed (new hash: 3298f1304...d4d421) or dependency folder (.aws-
sam/deps/4d3dfad6-a267-47a6-a6cd-e07d6fae318c) is missing for (HelloWorldFunction),
 downloading dependencies and copying/building source
Building codeuri: /Users/.../sam-app/hello_world runtime: python3.12 metadata: {}
 architecture: x86_64 functions: HelloWorldFunction
Running PythonPipBuilder:CleanUp
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource
Running PythonPipBuilder:CopySource


Build Succeeded


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


Commands you can use next
=========================
[*] Validate SAM template: sam validate
[*] Invoke Function: sam local invoke
[*] Test Function in the Cloud: sam sync --stack-name {{stack-name}} --watch
[*] Deploy: sam deploy --guided
```

3. The AWS SAM CLI creates a `.aws-sam` build directory. The following is an example:

```
.aws-sam
### build
#   ### HelloWorldFunction
#   #   ### __init__.py
```

```
#    #    ### app.py
#    #    ### requirements.txt
#    ### template.yaml
### build.toml
```

Depending on how your application is configured, the AWS SAM CLI does the following:

1. Downloads, installs, and organizes dependencies in the `.aws-sam/build` directory.

2. Prepares your Lambda code. This can include compiling your code, creating executable binaries, and building container images.

3. Copies build artifacts to the `.aws-sam` directory. The format will vary based on your application package type.

   a. For .zip package types, the artifacts are not zipped yet so that they can be used for local testing. The AWS SAM CLI zips your application when using `sam deploy`.

   b. For container image package types, a container image is created locally and referenced in the `.aws-sam/build.toml` file.

4. Copies the AWS SAM template over to the `.aws-sam` directory and modifies it with new file paths when necessary.

The following are the major components that make up your build artifacts in the `.aws-sam` directory:

- **The build directory** – Contains your Lambda functions and layers structured independently of each other. This results in a unique structure for each function or layer in the `.aws-sam/build` directory.

- **The AWS SAM template** – Modified with updated values based on changes during the build process.

- **The build.toml file** – A configuration file that contains build settings used by the AWS SAM CLI.

Local testing and deployment

When performing local testing with `sam local` or deployment with `sam deploy`, the AWS SAM CLI does the following:

1. It first checks to see if an `.aws-sam` directory exists and if an AWS SAM template is located within that directory. If these conditions are met, the AWS SAM CLI considers this as the root directory of your application.

2. If these conditions are not met, the AWS SAM CLI considers the original location of your AWS SAM template as the root directory of your application.

When developing, if changes are made to your original application files, run `sam build` to update the `.aws-sam` directory before testing locally.

Best practices

- Don't edit any code under the `.aws-sam/build` directory. Instead, update your original source code in your project folder and run `sam build` to update the `.aws-sam/build` directory.

- When you modify your original files, run `sam build` to update the `.aws-sam/build` directory.

- You may want the AWS SAM CLI to reference your project's original root directory instead of the `.aws-sam` directory, such as when developing and testing with `sam local`. Delete the `.aws-sam` directory or the AWS SAM template in the `.aws-sam` directory to have the AWS SAM CLI recognize your original project directory as the root project directory. When ready, run `sam build` again to create the `.aws-sam` directory.

- When you run `sam build`, the `.aws-sam/build` directory gets overwritten each time. The `.aws-sam` directory does not. If you want to store files, such as logs, store them in `.aws-sam` to prevent them from being overwritten.

Options for sam build

Building a single resource

Provide the resource's logical ID to build only that resource. The following is an example:

```
$ sam build HelloWorldFunction
```

To build a resource of a nested application or stack, provide the application or stack logical ID along with the resource logical ID using the format *<stack-logical-id>/<resource-logical-id>*:

```
$ sam build MyNestedStack/MyFunction
```

Building a Lambda function inside of a provided container

The `--use-container` option downloads a container image and uses it to build your Lambda functions. The local container is then referenced in your `.aws-sam/build.toml` file.

This option requires Docker to be installed. For instructions, see [Installing Docker](#).

The following is an example of this command:

```
$ sam build --use-container
```

You can specify the container image to use with the `--build-image` option. The following is an example:

```
$ sam build --use-container --build-image amazon/aws-sam-cli-build-image-nodejs20.x
```

To specify the container image to use for a single function, provide the function logical ID. The following is an example:

```
$ sam build --use-container --build-image Function1=amazon/aws-sam-cli-build-image-python3.12
```

Pass environment variables to the build container

Use the `--container-env-var` to pass environment variables to the build container. The following is an example:

```
$ sam build --use-container --container-env-var Function1.GITHUB_TOKEN=<token1> --container-env-var GLOBAL_ENV_VAR=<global-token>
```

To pass environment variables from a file, use the `--container-env-var-file` option. The following is an example:

```
$ sam build --use-container --container-env-var-file <env.json>
```

Example of the `env.json` file:

```
{
  "MyFunction1": {
    "GITHUB_TOKEN": "TOKEN1"
  },
```

```
    "MyFunction2": {
      "GITHUB_TOKEN": "TOKEN2"
    }
}
```

Speed up the building of applications that contain multiple functions

When you run `sam build` on an application with multiple functions, the AWS SAM CLI builds each function one at a time. To speed up the build process, use the `--parallel` option. This builds all of your functions and layers at the same time.

The following is an example of this command:

```
$ sam build —-parallel
```

Speed up build times by building your project in the source folder

For supported runtimes and build methods, you can use the `--build-in-source` option to build your project directly in the source folder. By default, the AWS SAM CLI builds in a temporary directory, which involves copying over source code and project files. With `--build-in-source`, the AWS SAM CLI builds directly in your source folder, which speeds up the build process by removing the need to copy files to a temporary directory.

For a list of supported runtimes and build methods, see [--build-in-source](#).

Troubleshooting

To troubleshoot the AWS SAM CLI, see [AWS SAM CLI troubleshooting](#).

Examples

Building an application that uses a native runtime and .zip package type

For this example, see [Tutorial: Deploying a Hello World application](#).

Building an application that uses a native runtime and image package type

First, we run `sam init` to initialize a new application. During the interactive flow, we select the `Image` package type. The following is an example:

```
$ sam init
...
Which template source would you like to use?
```

```
        1 - AWS Quick Start Templates
        2 - Custom Template Location
Choice: 1


Choose an AWS Quick Start application template
        1 - Hello World Example
        2 - Multi-step workflow
        3 - Serverless API
        4 - Scheduled task
        5 - Standalone function
        6 - Data processing
        7 - Hello World Example With Powertools
        8 - Infrastructure event management
        9 - Serverless Connector Hello World Example
        10 - Multi-step workflow with Connectors
        11 - Lambda EFS example
        12 - DynamoDB Example
        13 - Machine Learning
Template: 1


Use the most popular runtime and package type? (Python and zip) [y/N]: ENTER


Which runtime would you like to use?
        ...
        10 - java8
        11 - nodejs20.x
        12 - nodejs18.x
        13 - nodejs16.x
        ...
Runtime: 12


What package type would you like to use?
        1 - Zip
        2 - Image
Package type: 2


Based on your selections, the only dependency manager available is npm.
We will proceed copying the template using npm.


Would you like to enable X-Ray tracing on the function(s) in your application?  [y/
N]: ENTER


Would you like to enable monitoring using CloudWatch Application Insights?
```

```
For more info, please view https://docs.aws.amazon.com/AmazonCloudWatch/latest/
monitoring/cloudwatch-application-insights.html [y/N]: ENTER


Project name [sam-app]: ENTER


Cloning from https://github.com/aws/aws-sam-cli-app-templates (process may take a
 moment)


    ----------------------
    Generating application:
    ----------------------
    Name: sam-app
    Base Image: amazon/nodejs18.x-base
    Architectures: x86_64
    Dependency Manager: npm
    Output Directory: .
    Configuration file: sam-app/samconfig.toml


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


...
```

The AWS SAM CLI initializes an application and creates the following project directory:

```
sam-app
### README.md
### events
#   ### event.json
### hello-world
#   ### Dockerfile
#   ### app.mjs
#   ### package.json
#   ### tests
#       ### unit
#           ### test-handler.mjs
### samconfig.toml
### template.yaml
```

Next, we run `sam build` to build our application:

```
sam-app $ sam build
```

```
Building codeuri: /Users/.../build-demo/sam-app runtime: None metadata: {'DockerTag':
 'nodejs18.x-v1', 'DockerContext': '/Users/.../build-demo/sam-app/hello-world',
 'Dockerfile': 'Dockerfile'} architecture: arm64 functions: HelloWorldFunction
Building image for HelloWorldFunction function
Setting DockerBuildArgs: {} for HelloWorldFunction function
Step 1/4 : FROM public.ecr.aws/lambda/nodejs:18
 ---> f5b68038c080
Step 2/4 : COPY app.mjs package*.json ./
 ---> Using cache
 ---> 834e565aae80
Step 3/4 : RUN npm install
 ---> Using cache
 ---> 31c2209dd7b5
Step 4/4 : CMD ["app.lambdaHandler"]
 ---> Using cache
 ---> 2ce2a438e89d
Successfully built 2ce2a438e89d
Successfully tagged helloworldfunction:nodejs18.x-v1


Build Succeeded

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


Commands you can use next
=========================
[*] Validate SAM template: sam validate
[*] Invoke Function: sam local invoke
[*] Test Function in the Cloud: sam sync --stack-name {{stack-name}} --watch
[*] Deploy: sam deploy --guided
```

Building an application that includes a compiled programming language

In this example, we build an application that contains a Lambda function using the Go runtime.

First, we initialize a new application using `sam init` and configure our application to use Go:

```
$ sam init

...

Which template source would you like to use?
        1 - AWS Quick Start Templates
        2 - Custom Template Location
```

```
Choice: 1

Choose an AWS Quick Start application template
        1 - Hello World Example
        2 - Multi-step workflow
        3 - Serverless API
        ...
Template: 1

Use the most popular runtime and package type? (Python and zip) [y/N]: ENTER

Which runtime would you like to use?
        ...
        4 - dotnetcore3.1
        5 - go1.x
        6 - go (provided.al2)
        ...
Runtime: 5

What package type would you like to use?
        1 - Zip
        2 - Image
Package type: 1

Based on your selections, the only dependency manager available is mod.
We will proceed copying the template using mod.

Would you like to enable X-Ray tracing on the function(s) in your application?  [y/
N]: ENTER

Would you like to enable monitoring using CloudWatch Application Insights?
For more info, please view https://docs.aws.amazon.com/AmazonCloudWatch/latest/
monitoring/cloudwatch-application-insights.html [y/N]: ENTER

Project name [sam-app]: ENTER

Cloning from https://github.com/aws/aws-sam-cli-app-templates (process may take a
 moment)

    -----------------------
    Generating application:
    -----------------------
    Name: sam-app
    Runtime: go1.x
```

```
    Architectures: x86_64
    Dependency Manager: mod
    Application Template: hello-world
    Output Directory: .
    Configuration file: sam-app/samconfig.toml


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


...
```

The AWS SAM CLI initializes the application. The following is an example of the application directory structure:

```
sam-app
### Makefile
### README.md
### events
#   ### event.json
### hello-world
#   ### go.mod
#   ### go.sum
#   ### main.go
#   ### main_test.go
### samconfig.toml
### template.yaml
```

We reference the README.md file for this application's requirements.

```
...
## Requirements
* AWS CLI already configured with Administrator permission
* [Docker installed](https://www.docker.com/community-edition)
* [Golang](https://golang.org)
* SAM CLI - [Install the SAM CLI](https://docs.aws.amazon.com/serverless-application-
model/latest/developerguide/serverless-sam-cli-install.html)
...
```

Next, we run `sam local invoke` to test our function. This command errors since Go is not installed on our local machine:

```
sam-app $ sam local invoke
Invoking hello-world (go1.x)
```

```
Local image was not found.
Removing rapid images for repo public.ecr.aws/sam/emulation-go1.x
Building
 image.......................................................................................................
Using local image: public.ecr.aws/lambda/go:1-rapid-x86_64.


Mounting /Users/.../Playground/build/sam-app/hello-world as /var/task:ro,delegated
 inside runtime container
START RequestId: c6c5eddf-042b-4e1e-ba66-745f7c86dd31 Version: $LATEST
fork/exec /var/task/hello-world: no such file or directory: PathError
null
END RequestId: c6c5eddf-042b-4e1e-ba66-745f7c86dd31
REPORT RequestId: c6c5eddf-042b-4e1e-ba66-745f7c86dd31  Init Duration: 0.88 ms
 Duration: 175.75 ms Billed Duration: 176 ms Memory Size: 128 MB     Max Memory Used:
 128 MB
{"errorMessage":"fork/exec /var/task/hello-world: no such file or
 directory","errorType":"PathError"}%
```

Next, we run `sam build` to build our application. We encounter an error since Go is not installed on our local machine:

```
sam-app $ sam build
Starting Build use cache
Cache is invalid, running build and copying resources for following functions
 (HelloWorldFunction)
Building codeuri: /Users/.../Playground/build/sam-app/hello-world runtime: go1.x
 metadata: {} architecture: x86_64 functions: HelloWorldFunction

Build Failed
Error: GoModulesBuilder:Resolver - Path resolution for runtime: go1.x of binary: go was
 not successful
```

While we could configure our local machine to properly build our function, we instead use the `--use-container` option with `sam build`. The AWS SAM CLI downloads a container image, builds our function using the native GoModulesBuilder, and copies the resulting binary to our `.aws-sam/build/HelloWorldFunction` directory.

```
sam-app $ sam build --use-container
Starting Build use cache
Starting Build inside a container
Cache is invalid, running build and copying resources for following functions
 (HelloWorldFunction)
```

```
Building codeuri: /Users/.../build/sam-app/hello-world runtime: go1.x metadata: {}
 architecture: x86_64 functions: HelloWorldFunction

Fetching public.ecr.aws/sam/build-go1.x:latest-x86_64 Docker container
 image...........................................................................................
Mounting /Users/.../build/sam-app/hello-world as /tmp/samcli/source:ro,delegated inside
 runtime container
Running GoModulesBuilder:Build

Build Succeeded

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

Commands you can use next
=========================
[*] Validate SAM template: sam validate
[*] Invoke Function: sam local invoke
[*] Test Function in the Cloud: sam sync --stack-name {{stack-name}} --watch
[*] Deploy: sam deploy --guided
```

The following is an example of the .aws-sam directory:

```
.aws-sam
### build
#   ### HelloWorldFunction
#   #   ### hello-world
#   ### template.yaml
### build.toml
### cache
#   ### c860d011-4147-4010-addb-2eaa289f4d95
#       ### hello-world
### deps
```

Next, we run sam local invoke. Our function is successfully invoked:

```
sam-app $ sam local invoke
Invoking hello-world (go1.x)
Local image is up-to-date
Using local image: public.ecr.aws/lambda/go:1-rapid-x86_64.

Mounting /Users/.../Playground/build/sam-app/.aws-sam/build/HelloWorldFunction as /var/
task:ro,delegated inside runtime container
```

```
START RequestId: cfc8ffa8-29f2-49d4-b461-45e8c7c80479 Version: $LATEST
END RequestId: cfc8ffa8-29f2-49d4-b461-45e8c7c80479
REPORT RequestId: cfc8ffa8-29f2-49d4-b461-45e8c7c80479  Init Duration: 1.20 ms
 Duration: 1782.46 ms         Billed Duration: 1783 ms         Memory Size: 128 MB
 Max Memory Used: 128 MB
{"statusCode":200,"headers":null,"multiValueHeaders":null,"body":"Hello,
 72.21.198.67\n"}%
```

Learn more

To learn more about using the `sam build` command, see the following:

- **Learning AWS SAM: sam build** – Serverless Land "Learning AWS SAM" series on YouTube.
- **Learning AWS SAM | sam build | E3** – Serverless Land "Learning AWS SAM" series on YouTube.
- **AWS SAM build: how it provides artifacts for deployment (Sessions With SAM S2E8)** – Sessions with AWS SAM series on YouTube.
- **AWS SAM custom builds: How to use Makefiles to customize builds in SAM (S2E9)** – Sessions with AWS SAM series on YouTube.

Using sam deploy

Use the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam deploy` command to deploy your serverless application to the AWS Cloud.

- For an introduction to the AWS SAM CLI, see What is the AWS SAM CLI?.
- For a list of `sam deploy` command options, see sam deploy.
- For an example of using `sam deploy` during a typical development workflow, see Step 3: Deploy your application to the AWS Cloud.

Topics

- Prerequisites
- Deploying applications using sam deploy
- Best practices
- Options for sam deploy
- Troubleshooting
- Examples

- Learn more

Prerequisites

To use `sam deploy`, install the AWS SAM CLI by completing the following:

- AWS SAM prerequisites.

- Installing the AWS SAM CLI.

Before using `sam deploy`, we recommend a basic understanding of the following:

- Configuring the AWS SAM CLI.

- Using sam init.

- Using sam build.

Deploying applications using sam deploy

When you deploy a serverless application for the first time, use the `--guided` option. The AWS SAM CLI will guide you through an interactive flow to configure your application's deployment settings.

To deploy an application using the interactive flow

1. Go to the root directory of your project. This is the same location as your AWS SAM template.

   ```
   $ cd sam-app
   ```

2. Run the following command:

   ```
   $ sam deploy --guided
   ```

3. During the interactive flow, the AWS SAM CLI prompts you with options to configure your application's deployment settings.

   Brackets ([ ]) indicate default values. Leave your answer blank to select the default value. Default values are obtained from the following configuration files:

   - `~/.aws/config` – Your general AWS account settings.

- ~/.aws/credentials – Your AWS account credentials.

- *<project>*/samconfig.toml – Your project's configuration file.

Provide values by answering the AWS SAM CLI prompts. For example, you can enter **y** for **yes**, **n** for **no**, or string values.

The AWS SAM CLI writes your responses to your project's samconfig.toml file. For subsequent deployments, you can use sam deploy to deploy using these configured values. To reconfigure these values, use sam deploy --guided again or directly modify your configuration files.

The following is an example output:

```
sam-app $ sam deploy --guided

Configuring SAM deploy
======================

        Looking for config file [samconfig.toml] :  Found
        Reading default arguments  :  Success

        Setting default arguments for 'sam deploy'
        ========================================
        Stack Name [sam-app]: ENTER
        AWS Region [us-west-2]: ENTER
        #Shows you resources changes to be deployed and require a 'Y' to initiate
  deploy
        Confirm changes before deploy [Y/n]: ENTER
        #SAM needs permission to be able to create roles to connect to the
  resources in your template
        Allow SAM CLI IAM role creation [Y/n]: ENTER
        #Preserves the state of previously provisioned resources when an operation
  fails
        Disable rollback [y/N]: ENTER
        HelloWorldFunction may not have authorization defined, Is this okay? [y/
N]: y
        Save arguments to configuration file [Y/n]: ENTER
        SAM configuration file [samconfig.toml]: ENTER
        SAM configuration environment [default]: ENTER
```

4. Next, the AWS SAM CLI deploys your application to the AWS Cloud. During deployment, progress is displayed in your command prompt. The following are the major stages in deployment:

    - For applications with AWS Lambda functions packaged as a .zip file archive, the AWS SAM CLI zips and uploads the package to an Amazon Simple Storage Service (Amazon S3) bucket. If necessary, the AWS SAM CLI will create a new bucket.

    - For applications with Lambda functions package as a container image, the AWS SAM CLI uploads the image to Amazon Elastic Container Registry (Amazon ECR). If necessary, the AWS SAM CLI will create a new repository.

    - The AWS SAM CLI creates an AWS CloudFormation change set and deploys your application to AWS CloudFormation as a stack.

    - The AWS SAM CLI modifies your deployed AWS SAM template with the new `CodeUri` value for your Lambda functions.

    The following is an example of the AWS SAM CLI deployment output:

    ```
    Looking for resources needed for deployment:

    Managed S3 bucket: aws-sam-cli-managed-default-
    samclisourcebucket-1a4x26zbcdkqr
    A different default S3 bucket can be set in samconfig.toml and auto
    resolution of buckets turned off by setting resolve_s3=False

    Parameter "stack_name=sam-app" in [default.deploy.parameters] is defined as
    a global parameter [default.global.parameters].
    This parameter will be only saved under [default.global.parameters] in /
    Users/.../sam-app/samconfig.toml.

    Saved arguments to config file
    Running 'sam deploy' for future deployments will use the parameters saved
    above.
    The above parameters can be changed by modifying samconfig.toml
    Learn more about samconfig.toml syntax at
    https://docs.aws.amazon.com/serverless-application-model/latest/
    developerguide/serverless-sam-cli-config.html

    Uploading to sam-app-zip/da3c598813f1c2151579b73ad788cac8  262144 / 619839
    (42.29%)Uploading to sam-app-zip/da3c598813f1c2151579b73ad788cac8  524288 / 619839
    ```

```
    (84.58%)Uploading to sam-app-zip/da3c598813f1c2151579b73ad788cac8  619839 /
 619839  (100.00%)


        Deploying with following values
        ===============================
        Stack name                   : sam-app
        Region                       : us-west-2
        Confirm changeset            : True
        Disable rollback             : False
        Deployment s3 bucket         : aws-sam-cli-managed-default-
samclisourcebucket-1a4x26zbcdkqr
        Capabilities                 : ["CAPABILITY_IAM"]
        Parameter overrides          : {}
        Signing Profiles             : {}


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


        Uploading to sam-app-zip/be84c20f868068e4dc4a2c11966edf2d.template  1212 /
 1212  (100.00%)



Waiting for changeset to be created..

CloudFormation stack changeset
-------------------------------------------------------------------------------
Operation              LogicalResourceId      ResourceType
 Replacement
-------------------------------------------------------------------------------
+ Add                  HelloWorldFunctionHell  AWS::Lambda::Permissio  N/A

                       oWorldPermissionProd    n

+ Add                  HelloWorldFunctionRole  AWS::IAM::Role          N/A

+ Add                  HelloWorldFunction      AWS::Lambda::Function    N/A

+ Add                  ServerlessRestApiDeplo  AWS::ApiGateway::Deplo  N/A

                       yment47fc2d5f9d         yment

+ Add                  ServerlessRestApiProdS  AWS::ApiGateway::Stage  N/A
```

```
                                  tage

+ Add                      ServerlessRestApi        AWS::ApiGateway::RestA    N/A

                                                    pi


-------------------------------------------------------------------------------


Changeset created successfully. arn:aws:cloudformation:us-
west-2:012345678910:changeSet/samcli-deploy1680559234/d9f58a77-98bc-41cd-
b9f4-433a5a450d7a


Previewing CloudFormation changeset before deployment
========================================================
Deploy this changeset? [y/N]: y

2023-04-03 12:00:50 - Waiting for stack create/update to complete

CloudFormation events from stack operations (refresh every 5.0 seconds)
-------------------------------------------------------------------------------
ResourceStatus            ResourceType            LogicalResourceId
 ResourceStatusReason
-------------------------------------------------------------------------------
CREATE_IN_PROGRESS        AWS::IAM::Role          HelloWorldFunctionRole    -

CREATE_IN_PROGRESS        AWS::IAM::Role          HelloWorldFunctionRole    Resource
 creation

 Initiated
CREATE_COMPLETE           AWS::IAM::Role          HelloWorldFunctionRole    -

CREATE_IN_PROGRESS        AWS::Lambda::Function    HelloWorldFunction        -

CREATE_IN_PROGRESS        AWS::Lambda::Function    HelloWorldFunction        Resource
 creation

 Initiated
CREATE_COMPLETE           AWS::Lambda::Function    HelloWorldFunction        -

CREATE_IN_PROGRESS        AWS::ApiGateway::RestA    ServerlessRestApi         -
```

		pi	
CREATE_IN_PROGRESS creation Initiated	AWS::ApiGateway::RestA pi	ServerlessRestApi	Resource
CREATE_COMPLETE	AWS::ApiGateway::RestA pi	ServerlessRestApi	-
CREATE_IN_PROGRESS	AWS::Lambda::Permissio n	HelloWorldFunctionHell oWorldPermissionProd	-
CREATE_IN_PROGRESS	AWS::ApiGateway::Deplo yment	ServerlessRestApiDeplo yment47fc2d5f9d	-
CREATE_IN_PROGRESS creation Initiated	AWS::Lambda::Permissio n	HelloWorldFunctionHell oWorldPermissionProd	Resource
CREATE_IN_PROGRESS creation Initiated	AWS::ApiGateway::Deplo yment	ServerlessRestApiDeplo yment47fc2d5f9d	Resource
CREATE_COMPLETE	AWS::ApiGateway::Deplo yment	ServerlessRestApiDeplo yment47fc2d5f9d	-
CREATE_IN_PROGRESS	AWS::ApiGateway::Stage	ServerlessRestApiProdS tage	-
CREATE_IN_PROGRESS creation Initiated	AWS::ApiGateway::Stage	ServerlessRestApiProdS tage	Resource
CREATE_COMPLETE	AWS::ApiGateway::Stage	ServerlessRestApiProdS tage	-
CREATE_COMPLETE	AWS::Lambda::Permissio	HelloWorldFunctionHell	-

```
                               n                            oWorldPermissionProd

CREATE_COMPLETE                AWS::CloudFormation::S    sam-app-zip                      -

                               tack


-------------------------------------------------------------------------------------

CloudFormation outputs from deployed stack
-------------------------------------------------------------------------------------
Outputs


-------------------------------------------------------------------------------------
Key                 HelloWorldFunctionIamRole

Description          Implicit IAM Role created for Hello World function

Value               arn:aws:iam::012345678910:role/sam-app-zip-

HelloWorldFunctionRole-11ZOGSCG28H0M


Key                 HelloWorldApi

Description         API Gateway endpoint URL for Prod stage for Hello World
 function
Value               https://njzfhdmls0.execute-api.us-west-2.amazonaws.com/Prod/
hello/

Key                 HelloWorldFunction

Description         Hello World Lambda Function ARN

Value               arn:aws:lambda:us-west-2:012345678910:function:sam-app-

HelloWorldFunction-XPqNX4TBu7qn


-------------------------------------------------------------------------------------


Successfully created/updated stack - sam-app-zip in us-west-2
```

5. To view your deployed application, do the following:

1. Open the AWS CloudFormation console directly with the URL https://
   console.aws.amazon.com/cloudformation.

2. Select **Stacks**.

3. Identify your stack by application name and select it.

Verify changes before deployment

You can configure the AWS SAM CLI to display your AWS CloudFormation change set and ask for confirmation before deploying.

To confirm changes before deployment

1. During `sam deploy --guided`, enter **Y** to confirm changes before deployment.

   ```
   #Shows you resources changes to be deployed and require a 'Y' to initiate deploy
   Confirm changes before deploy [Y/n]: Y
   ```

   Alternatively, you can modify your `samconfig.toml` file with the following:

   ```
   [default.deploy]
   [default.deploy.parameters]
   confirm_changeset = true
   ```

2. During deployment, the AWS SAM CLI will ask you to confirm changes before deployment. The following is an example:

   ```
   Waiting for changeset to be created..

   CloudFormation stack changeset
   -------------------------------------------------------------------------------
   Operation              LogicalResourceId        ResourceType
     Replacement
   -------------------------------------------------------------------------------
   + Add                  HelloWorldFunctionHell   AWS::Lambda::Permissio    N/A

                          oWorldPermissionProd     n

   + Add                  HelloWorldFunctionRole   AWS::IAM::Role            N/A
   ```

```
 + Add                    HelloWorldFunction        AWS::Lambda::Function    N/A

 + Add                    ServerlessRestApiDeplo    AWS::ApiGateway::Deplo   N/A

                          yment47fc2d5f9d           yment

 + Add                    ServerlessRestApiProdS    AWS::ApiGateway::Stage   N/A

                          tage

 + Add                    ServerlessRestApi         AWS::ApiGateway::RestA   N/A

                                                    pi


-------------------------------------------------------------------------------

Changeset created successfully. arn:aws:cloudformation:us-
west-2:012345678910:changeSet/samcli-deploy1680559234/d9f58a77-98bc-41cd-
b9f4-433a5a450d7a
Previewing CloudFormation changeset before deployment
======================================================
Deploy this changeset? [y/N]: y
```

Specify additional parameters during deployment

You can specify additional parameter values to configure when deploying. You do this by modifying your AWS SAM template and configuring your parameter value during deployment.

To specify additional parameters

1. Modify the `Parameters` section of your AWS SAM template. The following is an example:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Globals:
...
Parameters:
  DomainName:
    Type: String
    Default: example
    Description: Domain name
```

2. Run `sam deploy --guided`. The following is an example output:

```
sam-app $ sam deploy --guided

Configuring SAM deploy
======================

        Looking for config file [samconfig.toml] :  Found
        Reading default arguments  :  Success

        Setting default arguments for 'sam deploy'
        =========================================
        Stack Name [sam-app-zip]: ENTER
        AWS Region [us-west-2]: ENTER
        Parameter DomainName [example]: ENTER
```

Configure code signing for your Lambda functions

You can configure code signing for your Lambda functions at deployment. You do this by modifying your AWS SAM template and configuring code signing during deployment.

To configure code signing

1. Specify `CodeSigningConfigArn` in your AWS SAM template. The following is an example:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: hello_world/
      Handler: app.lambda_handler
      Runtime: python3.7
      CodeSigningConfigArn: arn:aws:lambda:us-east-1:111122223333:code-signing-
config:csc-12e12345db1234567
```

2. Run `sam deploy --guided`. The AWS SAM CLI will prompt you to configure code signing. The following is an example output:

```
#Found code signing configurations in your function definitions
Do you want to sign your code? [Y/n]: ENTER
#Please provide signing profile details for the following functions & layers
#Signing profile details for function 'HelloWorld'
Signing Profile Name:
Signing Profile Owner Account ID (optional):
#Signing profile details for layer 'MyLayer', which is used by functions
 {'HelloWorld'}
Signing Profile Name:
Signing Profile Owner Account ID (optional):
```

Best practices

- When using `sam deploy`, the AWS SAM CLI deploys your application's build artifacts located in the `.aws-sam` directory. When you make changes to your application's original files, run `sam build` to update the `.aws-sam` directory before deploying.

- When deploying an application for the first time, use `sam deploy --guided` to configure deployment settings. For subsequent deployments, you can use `sam deploy` to deploy with your configured settings.

Options for sam deploy

The following are commonly used options for `sam deploy`. For a list of all options, see sam deploy.

Use the guided interactive flow to deploy your application

Use the `--guided` option to configure your application's deployment settings through an interactive flow. The following is an example:

```
$ sam deploy --guided
```

Your application's deployment settings are saved in your project's `samconfig.toml` file. To learn more, see Configure project settings.

Troubleshooting

To troubleshoot the AWS SAM CLI, see AWS SAM CLI troubleshooting.

Examples

Deploy a Hello World application that contains a Lambda function packaged as a .zip file archive

For an example, see [Step 3: Deploy your application to the AWS Cloud](#) in the Hello World application tutorial.

Deploy a Hello World application that contains a Lambda function packaged as a container image

First, we use `sam init` to create our Hello World application. During the interactive flow, we choose the `Python3.9` runtime and `Image` package type.

```
$ sam init
...
Which template source would you like to use?
        1 - AWS Quick Start Templates
        2 - Custom Template Location
Choice: 1

Choose an AWS Quick Start application template
        1 - Hello World Example
        2 - Multi-step workflow
        ...
Template: 1

Use the most popular runtime and package type? (Python and zip) [y/N]:  ENTER

Which runtime would you like to use?
        1 - aot.dotnet7 (provided.al2)
        ...
        15 - nodejs12.x
        16 - python3.9
        17 - python3.8
        ...
Runtime: 16

What package type would you like to use?
        1 - Zip
        2 - Image
Package type: 2
```

```
Based on your selections, the only dependency manager available is pip.
We will proceed copying the template using pip.
...
Project name [sam-app]: ENTER


    ----------------------
    Generating application:
    ----------------------
    Name: sam-app
    Base Image: amazon/python3.9-base
    Architectures: x86_64
    Dependency Manager: pip
    Output Directory: .
    Configuration file: sam-app/samconfig.toml

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

Next, we `cd` to the root directory of our project and run `sam build`. The AWS SAM CLI builds our Lambda function locally using Docker.

```
sam-app $ sam build
Building codeuri: /Users/.../sam-app runtime: None metadata: {'Dockerfile':
 'Dockerfile', 'DockerContext': '/Users/.../sam-app/hello_world', 'DockerTag':
 'python3.9-v1'} architecture: x86_64 functions: HelloWorldFunction
Building image for HelloWorldFunction function
Setting DockerBuildArgs: {} for HelloWorldFunction function
Step 1/5 : FROM public.ecr.aws/lambda/python:3.9
 ---> 0a5e3da309aa
Step 2/5 : COPY requirements.txt ./
 ---> abc4e82e85f9
Step 3/5 : RUN python3.9 -m pip install -r requirements.txt -t .
 ---> [Warning] The requested image's platform (linux/amd64) does not match the
 detected host platform (linux/arm64/v8) and no specific platform was requested
 ---> Running in 43845e7aa22d
Collecting requests
  Downloading requests-2.28.2-py3-none-any.whl (62 kB)
     ################################### 62.8/62.8 KB 829.5 kB/s eta 0:00:00
Collecting idna<4,>=2.5
  Downloading idna-3.4-py3-none-any.whl (61 kB)
     ################################### 61.5/61.5 KB 2.4 MB/s eta 0:00:00
Collecting charset-normalizer<4,>=2
```

```
    Downloading charset_normalizer-3.1.0-cp39-cp39-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl (199 kB)
      ################################### 199.2/199.2 KB 2.1 MB/s eta 0:00:00
Collecting certifi>=2017.4.17
  Downloading certifi-2022.12.7-py3-none-any.whl (155 kB)
      ################################### 155.3/155.3 KB 10.2 MB/s eta 0:00:00
Collecting urllib3<1.27,>=1.21.1
  Downloading urllib3-1.26.15-py2.py3-none-any.whl (140 kB)
      ################################### 140.9/140.9 KB 9.1 MB/s eta 0:00:00
Installing collected packages: urllib3, idna, charset-normalizer, certifi, requests
Successfully installed certifi-2022.12.7 charset-normalizer-3.1.0 idna-3.4
 requests-2.28.2 urllib3-1.26.15
Removing intermediate container 43845e7aa22d
 ---> cab8ace899ce
Step 4/5 : COPY app.py ./
 ---> 4146f3cd69f2
Step 5/5 : CMD ["app.lambda_handler"]
 ---> [Warning] The requested image's platform (linux/amd64) does not match the
 detected host platform (linux/arm64/v8) and no specific platform was requested
 ---> Running in f4131ddffb31
Removing intermediate container f4131ddffb31
 ---> d2f5180b2154
Successfully built d2f5180b2154
Successfully tagged helloworldfunction:python3.9-v1


Build Succeeded

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

Commands you can use next
=========================
[*] Validate SAM template: sam validate
[*] Invoke Function: sam local invoke
[*] Test Function in the Cloud: sam sync --stack-name {{stack-name}} --watch
[*] Deploy: sam deploy --guided
```

Next, we run `sam deploy --guided` to deploy our application. The AWS SAM CLI guides us through configuring our deployment settings. Then, the AWS SAM CLI deploys our application to the AWS Cloud.

```
sam-app $ sam deploy --guided
```

```
Configuring SAM deploy
======================

        Looking for config file [samconfig.toml] :  Found
        Reading default arguments  :  Success

        Setting default arguments for 'sam deploy'
        ========================================
        Stack Name [sam-app]: ENTER
        AWS Region [us-west-2]: ENTER
        #Shows you resources changes to be deployed and require a 'Y' to initiate
 deploy
        Confirm changes before deploy [Y/n]: ENTER
        #SAM needs permission to be able to create roles to connect to the resources in
 your template
        Allow SAM CLI IAM role creation [Y/n]: ENTER
        #Preserves the state of previously provisioned resources when an operation
 fails
        Disable rollback [y/N]: ENTER
        HelloWorldFunction may not have authorization defined, Is this okay? [y/N]: y
        Save arguments to configuration file [Y/n]: ENTER
        SAM configuration file [samconfig.toml]: ENTER
        SAM configuration environment [default]: ENTER


        Looking for resources needed for deployment:


        Managed S3 bucket: aws-sam-cli-managed-default-samclisourcebucket-1a4x26zbcdkqr
        A different default S3 bucket can be set in samconfig.toml and auto resolution
 of buckets turned off by setting resolve_s3=False


        Parameter "stack_name=sam-app" in [default.deploy.parameters] is defined as a
 global parameter [default.global.parameters].
        This parameter will be only saved under [default.global.parameters] in /
Users/.../sam-app/samconfig.toml.


        Saved arguments to config file
        Running 'sam deploy' for future deployments will use the parameters saved
 above.
        The above parameters can be changed by modifying samconfig.toml
        Learn more about samconfig.toml syntax at
        https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/
serverless-sam-cli-config.html
```

```
e95fc5e75742: Pushed
d8df51e7bdd7: Pushed
b1d0d7e0b34a: Pushed
0071317b94d8: Pushed
d98f98baf147: Pushed
2d244e0816c6: Pushed
eb2eeb1ebe42: Pushed
a5ca065a3279: Pushed
fe9e144829c9: Pushed
helloworldfunction-d2f5180b2154-python3.9-v1: digest:
 sha256:cceb71401b47dc3007a7a1e1f2e0baf162999e0e6841d15954745ecc0c447533 size: 2206


        Deploying with following values
        ===============================
        Stack name                   : sam-app
        Region                       : us-west-2
        Confirm changeset            : True
        Disable rollback             : False
        Deployment image repository  :
                                       {
                                             "HelloWorldFunction":
 "012345678910.dkr.ecr.us-west-2.amazonaws.com/samapp7427b055/
helloworldfunction19d43fc4repo"
                                       }
        Deployment s3 bucket         : aws-sam-cli-managed-default-
samclisourcebucket-1a4x26zbcdkqr
        Capabilities                 : ["CAPABILITY_IAM"]
        Parameter overrides          : {}
        Signing Profiles             : {}

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

HelloWorldFunction may not have authorization defined.
        Uploading to sam-app/682ad27c7cf7a17c7f77a1688b0844f2.template  1328 / 1328
 (100.00%)



Waiting for changeset to be created..

CloudFormation stack changeset
---------------------------------------------------------------------------------------
Operation                 LogicalResourceId         ResourceType              Replacement
```

```
------------------------------------------------------------------------------
+ Add                    HelloWorldFunctionHell    AWS::Lambda::Permissio    N/A

                         oWorldPermissionProd      n

+ Add                    HelloWorldFunctionRole    AWS::IAM::Role            N/A

+ Add                    HelloWorldFunction        AWS::Lambda::Function     N/A

+ Add                    ServerlessRestApiDeplo    AWS::ApiGateway::Deplo    N/A

                         yment47fc2d5f9d           yment

+ Add                    ServerlessRestApiProdS    AWS::ApiGateway::Stage    N/A

                         tage

+ Add                    ServerlessRestApi         AWS::ApiGateway::RestA    N/A

                                                   pi

------------------------------------------------------------------------------


Changeset created successfully. arn:aws:cloudformation:us-
west-2:012345678910:changeSet/samcli-deploy1680634124/0ffd4faf-2e2b-487e-
b9e0-9116e8299ac4


Previewing CloudFormation changeset before deployment
=======================================================
Deploy this changeset? [y/N]: y

2023-04-04 08:49:15 - Waiting for stack create/update to complete

CloudFormation events from stack operations (refresh every 5.0 seconds)
------------------------------------------------------------------------------
ResourceStatus          ResourceType              LogicalResourceId
 ResourceStatusReason
------------------------------------------------------------------------------
CREATE_IN_PROGRESS      AWS::CloudFormation::S     sam-app                  User
 Initiated
                        tack
```

Status	Type	Logical ID	Reason
CREATE_IN_PROGRESS	AWS::IAM::Role	HelloWorldFunctionRole	-
CREATE_IN_PROGRESS creation	AWS::IAM::Role	HelloWorldFunctionRole	Resource Initiated
CREATE_COMPLETE	AWS::IAM::Role	HelloWorldFunctionRole	-
CREATE_IN_PROGRESS	AWS::Lambda::Function	HelloWorldFunction	-
CREATE_IN_PROGRESS creation	AWS::Lambda::Function	HelloWorldFunction	Resource Initiated
CREATE_COMPLETE	AWS::Lambda::Function	HelloWorldFunction	-
CREATE_IN_PROGRESS	AWS::ApiGateway::RestApi	ServerlessRestApi	-
CREATE_IN_PROGRESS creation	AWS::ApiGateway::RestApi	ServerlessRestApi	Resource Initiated
CREATE_COMPLETE	AWS::ApiGateway::RestApi	ServerlessRestApi	-
CREATE_IN_PROGRESS	AWS::Lambda::Permission	HelloWorldFunctionHelloWorldPermissionProd	-
CREATE_IN_PROGRESS	AWS::ApiGateway::Deployment	ServerlessRestApiDeployment47fc2d5f9d	-
CREATE_IN_PROGRESS creation	AWS::Lambda::Permission	HelloWorldFunctionHelloWorldPermissionProd	Resource Initiated
CREATE_IN_PROGRESS creation	AWS::ApiGateway::Deployment	ServerlessRestApiDeployment47fc2d5f9d	Resource Initiated

CREATE_COMPLETE	AWS::ApiGateway::Deplo yment	ServerlessRestApiDeplo yment47fc2d5f9d	-
CREATE_IN_PROGRESS	AWS::ApiGateway::Stage	ServerlessRestApiProdS tage	-
CREATE_IN_PROGRESS creation	AWS::ApiGateway::Stage	ServerlessRestApiProdS tage	Resource Initiated
CREATE_COMPLETE	AWS::ApiGateway::Stage	ServerlessRestApiProdS tage	-
CREATE_COMPLETE	AWS::Lambda::Permissio n	HelloWorldFunctionHell oWorldPermissionProd	-
CREATE_COMPLETE	AWS::CloudFormation::S tack	sam-app	-

```
-------------------------------------------------------------------------------------
CloudFormation outputs from deployed stack
-------------------------------------------------------------------------------------
Outputs
-------------------------------------------------------------------------------------
Key                 HelloWorldFunctionIamRole

Description         Implicit IAM Role created for Hello World function

Value               arn:aws:iam::012345678910:role/sam-app-HelloWorldFunctionRole-
JFML1JOKHJ71

Key                 HelloWorldApi

Description         API Gateway endpoint URL for Prod stage for Hello World function

Value               https://endlwiqqod.execute-api.us-west-2.amazonaws.com/Prod/hello/
```

```
Key                    HelloWorldFunction

Description            Hello World Lambda Function ARN

Value                 arn:aws:lambda:us-west-2:012345678910:function:sam-app-
HelloWorldFunction-
kyg6Y2iNRUPg


-------------------------------------------------------------------------------


Successfully created/updated stack - sam-app in us-west-2
```

Learn more

To learn more about using the AWS SAM CLI `sam deploy` command, see the following:

- **The Complete AWS SAM Workshop: Module 3 - Deploy manually** – Learn how to build, package, and deploy a serverless application using the AWS SAM CLI.

Using sam init

The AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam init` command provides options to initialize a new serverless application that consists of:

- An AWS SAM template to define your infrastructure code.
- A folder structure that organizes your application.
- Configuration for your AWS Lambda functions.

For an introduction to the AWS SAM CLI, see What is the AWS SAM CLI?.

For a list of AWS SAM CLI `sam init` command options, see sam init.

Topics

- Initialize a new serverless application
- Options for sam init
- Troubleshooting
- Examples

- [Learn more](#)

Initialize a new serverless application

To initialize a new serverless application using the AWS SAM CLI

1. `cd` to a starting directory.

2. Run the following at the command line:

    ```
    $ sam init
    ```

3. The AWS SAM CLI will guide you through an interactive flow to create a new serverless application.

Choose a starting template

A *template* consists of the following:

1. An AWS SAM template for your infrastructure code.

2. A starting project directory that organizes your project files. For example, this may include:

   a. A structure for your Lambda function code and their dependencies.

   b. An `events` folder that contains test events for local testing.

   c. A `tests` folder to support unit testing.

   d. A `samconfig.toml` file to configure project settings.

   e. A `ReadMe` file and other basic starting project files.

   The following is an example of a starting project directory:

   ```
   sam-app
   ### README.md
   ### __init__.py
   ### events
   #   ### event.json
   ### hello_world
   #   ### __init__.py
   #   ### app.py
   #   ### requirements.txt
   ### samconfig.toml
   ```

```
### template.yaml
### tests
    ### __init__.py
    ### integration
    #   ### __init__.py
    #   ### test_api_gateway.py
    ### requirements.txt
    ### unit
        ### __init__.py
        ### test_handler.py
```

You can select from a list of available *AWS Quick Start Templates* or provide your own *Custom Template Location*.

To choose an AWS Quick Start Template

1. When prompted, select **AWS Quick Start Templates**.

2. Select an AWS Quick Start template to begin with. The following is an example:

```
Which template source would you like to use?
    1 - AWS Quick Start Templates
    2 - Custom Template Location
Choice: 1

Choose an AWS Quick Start application template
    1 - Hello World Example
    2 - Multi-step workflow
    3 - Serverless API
    4 - Scheduled task
    5 - Standalone function
    6 - Data processing
    7 - Hello World Example With Powertools
    8 - Infrastructure event management
    9 - Serverless Connector Hello World Example
    10 - Multi-step workflow with Connectors
    11 - Lambda EFS example
    12 - DynamoDB Example
    13 - Machine Learning
Template: 4
```

To choose your own custom template location

1. When prompted, select the **Custom Template Location**.

```
Which template source would you like to use?
    1 - AWS Quick Start Templates
    2 - Custom Template Location
Choice: 2
```

2. The AWS SAM CLI will prompt you to provide a template location.

```
Template location (git, mercurial, http(s), zip, path):
```

Provide any of the following locations to your template .zip file archive:

- **GitHub repository** – The path to the .zip file in your GitHub repository. The file must be in the root of your repository.
- **Mercurial repository** – The path to the .zip file in your Mercurial repository. The file must be in the root of your repository.
- **.zip path** – An HTTPS or local path to your .zip file.

3. The AWS SAM CLI will initialize your serverless application using your custom template.

Choose a runtime

When you choose an *AWS Quick Start Template*, the AWS SAM CLI prompts you to select a runtime for your Lambda functions. The list of options displayed by the AWS SAM CLI are the runtimes supported natively by Lambda.

- The runtime provides a language-specific environment that runs in an execution environment.
- When deployed to the AWS Cloud, the Lambda service invokes your function in an execution environment.

You can use any other programming language with a custom runtime. To do this, you need to manually create your starting application structure. You can then use `sam init` to quickly initialize your application by configuring a custom template location.

From your selection, the AWS SAM CLI creates the starting directory for your Lambda function code and dependencies.

If Lambda supports multiple dependency managers for your runtime, you will be prompted to choose your preferred dependency manager.

Choose a package type

When you choose an *AWS Quick Start Template* and a *runtime*, the AWS SAM CLI prompts you to select a *package type.* The package type determines how your Lambda functions are deployed to use with the Lambda service. The two supported package types are:

1. **Container image** – Contains the base operating system, the runtime, Lambda extensions, your application code, and its dependencies.
2. **.zip file archive** – Contains your application code and its dependencies.

To learn more about deployment package types, see [Lambda deployment packages](#) in the *AWS Lambda Developer Guide*.

The following is an example directory structure of an application with a Lambda function packaged as a container image. The AWS SAM CLI downloads the image and creates a `Dockerfile` in the function's directory to specify the image.

```
sam-app
### README.md
### __init__.py
### events
#   ### event.json
### hello_world
#   ### Dockerfile
#   ### __init__.py
#   ### app.py
#   ### requirements.txt
### samconfig.toml
### template.yaml
### tests
    ### __init__.py
    ### unit
        ### __init__.py
        ### test_handler.py
```

The following is an example directory structure of an application with a function packaged as a .zip file archive.

```
sam-app
### README.md
### __init__.py
### events
#    ### event.json
### hello_world
#    ### __init__.py
#    ### app.py
#    ### requirements.txt
### samconfig.toml
### template.yaml
### tests
    ### __init__.py
    ### integration
    #    ### __init__.py
    #    ### test_api_gateway.py
    ### requirements.txt
    ### unit
        ### __init__.py
        ### test_handler.py
```

Configure AWS X-Ray tracing

You can choose to activate AWS X-Ray tracing. To learn more, see What is AWS X-Ray? in the *AWS X-Ray Developer Guide*.

If you activate, the AWS SAM CLI configures your AWS SAM template. The following is an example:

```
Globals:
  Function:
    ...
    Tracing: Active
  Api:
    TracingEnabled: True
```

Configure monitoring with Amazon CloudWatch Application Insights

You can choose to activate monitoring using Amazon CloudWatch Application Insights. To learn more, see Amazon CloudWatch Application Insights in the *Amazon CloudWatch User Guide*.

If you activate, the AWS SAM CLI configures your AWS SAM template. The following is an example:

```
Resources:
```

```
  ApplicationResourceGroup:
    Type: AWS::ResourceGroups::Group
    Properties:
      Name:
        Fn::Join:
        - ''
        - - ApplicationInsights-SAM-
          - Ref: AWS::StackName
      ResourceQuery:
        Type: CLOUDFORMATION_STACK_1_0
  ApplicationInsightsMonitoring:
    Type: AWS::ApplicationInsights::Application
    Properties:
      ResourceGroupName:
        Fn::Join:
        - ''
        - - ApplicationInsights-SAM-
          - Ref: AWS::StackName
      AutoConfigurationEnabled: 'true'
    DependsOn: ApplicationResourceGroup
```

Name your application

Provide a name for your application. The AWS SAM CLI creates a top-level folder for your
application using this name.

Options for sam init

The following are some of the main options you can use with the `sam init` command. For a list of
all options, see sam init.

Initialize an application using a custom template location

Use the `--location` option and provide a supported custom template location. The following is
an example:

```
$ sam init --location https://github.com/aws-samples/sessions-with-aws-sam/raw/master/
starter-templates/web-app.zip
```

Initialize an application without the interactive flow

Use the `--no-interactive` option and provide your configuration choices at the command line
to skip the interactive flow. The following is an example:

```
$ sam init --no-interactive --runtime go1.x --name go-demo --dependency-manager mod --
app-template hello-world
```

Troubleshooting

To troubleshoot the AWS SAM CLI, see [AWS SAM CLI troubleshooting](#).

Examples

Initialize a new serverless application using the Hello World AWS Starter Template

For this example, see [Step 1: Initialize the sample Hello World application](#) in *Tutorial: Deploying a Hello World application*.

Initialize a new serverless application with a custom template location

The following are examples of providing a GitHub location to your custom template:

```
$ sam init --location gh:aws-samples/cookiecutter-aws-sam-python
$ sam init --location git+sh://git@github.com/aws-samples/cookiecutter-aws-sam-
python.git
$ sam init --location hg+ssh://hg@bitbucket.org/repo/template-name
```

The following is an example of a local file path:

```
$ sam init --location /path/to/template.zip
```

The following is an example of a path reachable by HTTPS:

```
$ sam init --location https://github.com/aws-samples/sessions-with-aws-sam/raw/master/
starter-templates/web-app.zip
```

Learn more

To learn more about using the `sam init` command, see the following:

- [Learning AWS SAM: sam init](#) – Serverless Land "Learning AWS SAM" series on YouTube.
- [Structuring serverless applications for use with the AWS SAM CLI (Sessions with SAM S2E7)](#) – Sessions with AWS SAM series on YouTube.

Using sam local

Use the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam local` command to test your serverless applications locally.

For an introduction to the AWS SAM CLI, see [What is the AWS SAM CLI?](#).

Prerequisites

To use `sam local`, install the AWS SAM CLI by completing the following:

- [AWS SAM prerequisites](#).
- [Installing the AWS SAM CLI](#).

Before using `sam local`, we recommend a basic understanding of the following:

- [Configuring the AWS SAM CLI](#).
- [Using sam init](#).
- [Using sam build](#).
- [Using sam deploy](#).

Using the sam local command

Use the `sam local` command with any of its subcommands to perform different types of local testing for your application.

```
$ sam local <subcommand>
```

To learn more about each subcommand, see the following:

- **[sam local generate-event](#)** – Generate AWS service events for local testing.
- **[sam local invoke](#)** – Initiate a one-time invocation of an AWS Lambda function locally.
- **[sam local start-api](#)** – Run your Lambda functions using a local HTTP server.
- **[sam local start-lambda](#)** – Run your Lambda functions using a local HTTP server for use with the AWS CLI or SDKs.

Using sam local generate-event

Use the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam local generate-event` subcommand to generate event payload samples for supported AWS services. You can then modify and pass these events to local resources for testing.

- For an introduction to the AWS SAM CLI, see [What is the AWS SAM CLI?](#).
- For a list of `sam local generate-event` command options, see [sam local generate-event](#).

An *event* is a JSON object that gets generated when an AWS service performs an action or task. These events contain specific information, such as the data that was processed or the timestamp of the event. Most AWS services generate events and each service's events are formatted uniquely for its service.

Events generated by one service are passed to other services as an *event source*. For example, an item placed in an Amazon Simple Storage Service (Amazon S3) bucket can generate an event. This event can then be used as the event source for an AWS Lambda function to process the data further.

Events that you generate with `sam local generate-event` are formatted in the same structure as the actual events created by the AWS service. You can modify the contents of these events and use them to test resources in your application.

Prerequisites

To use `sam local generate-event`, install the AWS SAM CLI by completing the following:

- [AWS SAM prerequisites](#).
- [Installing the AWS SAM CLI](#).

Before using `sam local generate-event`, we recommend a basic understanding of the following:

- [Configuring the AWS SAM CLI](#).
- [Using sam init](#).
- [Using sam build](#).
- [Using sam deploy](#).

Generate sample events

Use the AWS SAM CLI `sam local generate-event` subcommand to generate events for supported AWS services.

To see a list of supported AWS services

1. Run the following:

```
$ sam local generate-event
```

2. The list of supported AWS services will display. The following is an example:

```
$ sam local generate-event
...
Commands:
  alb
  alexa-skills-kit
  alexa-smart-home
  apigateway
  appsync
  batch
  cloudformation
  ...
```

To generate a local event

1. Run `sam local generate-event` and provide the supported service name. This will display a list of event types that you can generate. The following is an example:

```
$ sam local generate-event s3

Usage: sam local generate-event s3 [OPTIONS] COMMAND [ARGS]...

Options:
  -h, --help  Show this message and exit.

Commands:
  batch-invocation  Generates an Amazon S3 Batch Operations Invocation Event
  delete            Generates an Amazon S3 Delete Event
  put               Generates an Amazon S3 Put Event
```

2. To generate the sample event, run `sam local generate-event`, providing the service and
    event type.

    ```
    $ sam local generate-event <service> <event>
    ```

The following is an example:

```
$ sam local generate-event s3 put
{
  "Records": [
    {
      "eventVersion": "2.0",
      "eventSource": "aws:s3",
      "awsRegion": "us-east-1",
      "eventTime": "1970-01-01T00:00:00.000Z",
      "eventName": "ObjectCreated:Put",
      "userIdentity": {
        "principalId": "EXAMPLE"
      },
      "requestParameters": {
        "sourceIPAddress": "127.0.0.1"
      },
      "responseElements": {
        "x-amz-request-id": "EXAMPLE123456789",
        "x-amz-id-2": "EXAMPLE123/5678abcdefghijklambdaisawesome/
mnopqrstuvwxyzABCDEFGH"
      },
      "s3": {
        "s3SchemaVersion": "1.0",
        "configurationId": "testConfigRule",
        "bucket": {
          "name": "example-bucket",
          "ownerIdentity": {
            "principalId": "EXAMPLE"
          },
          "arn": "arn:aws:s3:::example-bucket"
        },
        "object": {
          "key": "test/key",
          "size": 1024,
          "eTag": "0123456789abcdef0123456789abcdef",
          "sequencer": "0A1B2C3D4E5F678901"
```

```
            }
        }
      }
    ]
  }
```

These sample events contain placeholder values. You can modify these values to reference actual resources in your application or values to help with local testing.

To modify a sample event

1. You can modify sample events at the command prompt. To see your options, run the following:

```
$ sam local generate-event <service> <event> --help
```

The following is an example:

```
$ sam local generate-event s3 put --help

Usage: sam local generate-event s3 put [OPTIONS]

Options:
  --region TEXT        Specify the region name you'd like, otherwise the
                       default = us-east-1
  --partition TEXT     Specify the partition name you'd like, otherwise the
                       default = aws
  --bucket TEXT        Specify the bucket name you'd like, otherwise the
                       default = example-bucket
  --key TEXT           Specify the key name you'd like, otherwise the default =
                       test/key
  --debug              Turn on debug logging to print debug message generated
                       by AWS SAM CLI and display timestamps.
  --config-file TEXT   Configuration file containing default parameter values.
                       [default: samconfig.toml]
  --config-env TEXT    Environment name specifying default parameter values in
                       the configuration file.  [default: default]
  -h, --help           Show this message and exit.
```

2. Use any of these options at the command prompt to modify your sample event payload. The following is an example:

```
$ sam local generate-event s3 put--bucket MyBucket

{
  "Records": [
    {
      "eventVersion": "2.0",
      "eventSource": "aws:s3",
      "awsRegion": "us-east-1",
      "eventTime": "1970-01-01T00:00:00.000Z",
      "eventName": "ObjectCreated:Put",
      "userIdentity": {
        "principalId": "EXAMPLE"
      },
      "requestParameters": {
        "sourceIPAddress": "127.0.0.1"
      },
      "responseElements": {
        "x-amz-request-id": "EXAMPLE123456789",
        "x-amz-id-2": "EXAMPLE123/5678abcdefghijklambdaisawesome/
mnopqrstuvwxyzABCDEFGH"
      },
      "s3": {
        "s3SchemaVersion": "1.0",
        "configurationId": "testConfigRule",
        "bucket": {
          "name": "MyBucket",
          "ownerIdentity": {
            "principalId": "EXAMPLE"
          },
          "arn": "arn:aws:s3:::MyBucket"
        },
        "object": {
          "key": "test/key",
          "size": 1024,
          "eTag": "0123456789abcdef0123456789abcdef",
          "sequencer": "0A1B2C3D4E5F678901"
        }
      }
    }
  ]
}
```

Use generated events for local testing

Save your generated events locally and use other `sam local` subcommands to test.

To save your generated events locally

- Run the following:

  ```
  $ sam local generate-event <service> <event> <event-option> > <filename.json>
  ```

  The following is an example of an event being saved as an `s3.json` file in the `events` folder of our project.

  ```
  sam-app$ sam local generate-event s3 put --bucket MyBucket > events/s3.json
  ```

To use a generated event for local testing

- Pass the event with other `sam local` subcommands by using the `--event` option.

  The following is an example of using the `s3.json` event to invoke our Lambda function locally:

  ```
  sam-app$ sam local invoke --event events/s3.json S3JsonLoggerFunction

  Invoking src/handlers/s3-json-logger.s3JsonLoggerHandler (nodejs18.x)
  Local image is up-to-date
  Using local image: public.ecr.aws/lambda/nodejs:18-rapid-x86_64.

  Mounting /Users/.../sam-app/.aws-sam/build/S3JsonLoggerFunction as /var/
  task:ro,delegated, inside runtime container
  START RequestId: f4f45b6d-2ec6-4235-bc7b-495ec2ae0128 Version: $LATEST
  END RequestId: f4f45b6d-2ec6-4235-bc7b-495ec2ae0128
  REPORT RequestId: f4f45b6d-2ec6-4235-bc7b-495ec2ae0128  Init Duration: 1.23 ms
   Duration: 9371.93 ms      Billed Duration: 9372 ms        Memory Size: 128 MB
   Max Memory Used: 128 MB
  ```

Learn more

For a list of all `sam local generate-event` options, see sam local generate-event.

For a demo of using `sam local`, see [AWS SAM for local development. Testing AWS Cloud resources from local development environments](#) in the *Serverless Land Sessions with SAM series on YouTube*.

Using sam local invoke

Use the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam local invoke` subcommand to initiate a one-time invocation of an AWS Lambda function locally.

- For an introduction to the AWS SAM CLI, see [What is the AWS SAM CLI?](#).
- For a list of `sam local invoke` command options, see [sam local invoke](#).
- For an example of using `sam local invoke` during a typical development workflow, see [Step 7: (Optional) Test your application locally](#).

Prerequisites

To use `sam local invoke`, install the AWS SAM CLI by completing the following:

- [AWS SAM prerequisites](#).
- [Installing the AWS SAM CLI](#).

Before using `sam local invoke`, we recommend a basic understanding of the following:

- [Configuring the AWS SAM CLI](#).
- [Using sam init](#).
- [Using sam build](#).
- [Using sam deploy](#).

Invoke a Lambda function locally

When you run `sam local invoke`, the AWS SAM CLI assumes that your current working directory is your project's root directory. The AWS SAM CLI will first look for a `template.[yaml|yml]` file within a `.aws-sam` subfolder. If not found, the AWS SAM CLI will look for a `template.[yaml|yml]` file within your current working directory.

To invoke a Lambda function locally

1. From the root directory of your project, run the following:

```
$ sam local invoke <options>
```

2. If your application contains more than one function, provide the function's logical ID. The following is an example:

```
$ sam local invoke HelloWorldFunction
```

3. The AWS SAM CLI builds your function in a local container using Docker. It then invokes your function and outputs your function's response.

   The following is an example:

```
$ sam local invoke
Invoking app.lambda_handler (python3.9)
Local image is out of date and will be updated to the latest runtime. To skip this,
 pass in the parameter --skip-pull-image
Building
 image........................................................................................
Using local image: public.ecr.aws/lambda/python:3.9-rapid-x86_64.

Mounting /Users/.../sam-app/.aws-sam/build/HelloWorldFunction as /var/
task:ro,delegated, inside runtime container
START RequestId: 64bf7e54-5509-4762-a97c-3d740498d3df Version: $LATEST
END RequestId: 64bf7e54-5509-4762-a97c-3d740498d3df
REPORT RequestId: 64bf7e54-5509-4762-a97c-3d740498d3df  Init Duration: 1.09 ms
  Duration: 608.42 ms       Billed Duration: 609 ms Memory Size: 128 MB     Max
 Memory Used: 128 MB
{"statusCode": 200, "body": "{\"message\": \"hello world\"}"}%
```

Managing logs

When using `sam local invoke`, the Lambda function runtime output (for example, logs) is output to `stderr`, and the Lambda function result is output to `stdout`.

The following is an example of a basic Lambda function:

```
def handler(event, context):
    print("some log") # this goes to stderr
    return "hello world" # this goes to stdout
```

You can save these standard outputs. The following is an example:

```
$ sam local invoke 1> stdout.log
...

$ cat stdout.log
"hello world"

$ sam local invoke 2> stderr.log
...

$ cat stderr.log
Invoking app.lambda_handler (python3.9)
Local image is up-to-date
Using local image: public.ecr.aws/lambda/python:3.9-rapid-x86_64.
Mounting /Users/.../sam-app/.aws-sam/build/HelloWorldFunction as /var/
task:ro,delegated, inside runtime container
START RequestId: 0b46e646-3bdf-4b58-8beb-242d00912c46 Version: $LATEST
some log
END RequestId: 0b46e646-3bdf-4b58-8beb-242d00912c46
REPORT RequestId: 0b46e646-3bdf-4b58-8beb-242d00912c46  Init Duration: 0.91 ms
 Duration: 589.19 ms Billed Duration: 590 ms Memory Size: 128 MB Max Memory Used: 128
 MB
```

You can use these standard outputs to further automate your local development processes.

Options

Pass custom events to invoke the Lambda function

To pass an event to the Lambda function, use the `--event` option. The following is an example:

```
$ sam local invoke --event events/s3.json S3JsonLoggerFunction
```

You can create events with the `sam local generate-event` subcommand. To learn more, see [Using sam local generate-event](#).

Pass environment variables when invoking your Lambda function

If your Lambda function uses environment variables, you can pass them during local testing with the `--env-vars` option. This is a great way to test a Lambda function locally with services in your application that are already deployed the cloud. The following is an example:

```
$ sam local invoke --env-vars locals.json
```

Specify a template or function

To specify a template for the AWS SAM CLI to reference, use the `--template` option. The AWS SAM CLI will load just that AWS SAM template and the resources it points to.

To invoke a function of a nested application or stack, provide the application or stack logical ID along with the function logical ID. The following is an example:

```
$ sam local invoke StackLogicalId/FunctionLogicalId
```

Test a Lambda function from your Terraform project

Use the `--hook-name` option to locally test Lambda functions from your Terraform projects. To learn more, see [Using the AWS SAM CLI with Terraform for local debugging and testing](#).

The following is an example:

```
$ sam local invoke --hook-name terraform --beta-features
```

Best practices

If your application has a `.aws-sam` directory from running `sam build`, be sure to run `sam build` every time you update your function code. Then, run `sam local invoke` to locally test your updated function code.

Local testing is a great solution for quick development and testing before deploying to the cloud. However, local testing doesn't validate everything, such as permissions between your resources in the cloud. As much as possible, test your applications in the cloud. We recommend [using `sam sync`](#) to speed up your cloud testing workflows.

Examples

Generate an Amazon API Gateway sample event and use it to invoke a Lambda function locally

First, we generate an API Gateway HTTP API event payload and save it to our `events` folder.

```
$ sam local generate-event apigateway http-api-proxy > events/apigateway_event.json
```

Next, we modify our Lambda function to return a parameter value from the event.

```python
def lambda_handler(event, context):
    print("HelloWorldFunction invoked")
    return {
        "statusCode": 200,
        "body": json.dumps({
            "message": event['queryStringParameters']['parameter2'],
        }),
    }
```

Next, we locally invoke our Lambda function and provide our custom event.

```
$ sam local invoke --event events/apigateway_event.json

Invoking app.lambda_handler (python3.9)
Local image is up-to-date
Using local image: public.ecr.aws/lambda/python:3.9-rapid-x86_64.

Mounting /Users/...sam-app/.aws-sam/build/HelloWorldFunction as /var/task:ro,delegated,
 inside runtime container
START RequestId: 59535d0d-3d9e-493d-8c98-6264e8e961b8 Version: $LATEST
some log
END RequestId: 59535d0d-3d9e-493d-8c98-6264e8e961b8
REPORT RequestId: 59535d0d-3d9e-493d-8c98-6264e8e961b8  Init Duration: 1.63 ms
 Duration: 564.07 ms      Billed Duration: 565 ms Memory Size: 128 MB    Max Memory
 Used: 128 MB
{"statusCode": 200, "body": "{\"message\": \"value\"}"}%
```

Pass environment variables when invoking a Lambda function locally

This application has a Lambda function that uses an environment variable for an Amazon DynamoDB table name. The following is an example of the function defined in the AWS SAM template:

```yaml
AWSTemplateFormatVersion: 2010-09-09
Transform: AWS::Serverless-2016-10-31
...
Resources:
  getAllItemsFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: src/get-all-items.getAllItemsHandler
      Description: get all items
```

```
      Policies:
        - DynamoDBReadPolicy:
            TableName: !Ref SampleTable
      Environment:
        Variables:
          SAMPLE_TABLE: !Ref SampleTable
...
```

We want to locally test our Lambda function while having it interact with our DynamoDB table in the cloud. To do this, we create our environment variables file and save it in our project's root directory as `locals.json`. The value provided here for SAMPLE_TABLE references our DynamoDB table in the cloud.

```
{
    "getAllItemsFunction": {
        "SAMPLE_TABLE": "dev-demo-SampleTable-1U991234LD5UM98"
    }
}
```

Next, we run `sam local invoke` and pass in our environment variables with the `--env-vars` option.

```
$ sam local invoke getAllItemsFunction --env-vars locals.json

Mounting /Users/...sam-app/.aws-sam/build/HelloWorldFunction as /var/task:ro,delegated,
 inside runtime container
START RequestId: 59535d0d-3d9e-493d-8c98-6264e8e961b8 Version: $LATEST
some log
END RequestId: 59535d0d-3d9e-493d-8c98-6264e8e961b8
REPORT RequestId: 59535d0d-3d9e-493d-8c98-6264e8e961b8  Init Duration: 1.63 ms
 Duration: 564.07 ms       Billed Duration: 565 ms Memory Size: 128 MB     Max Memory
 Used: 128 MB
{"statusCode":200,"body":"{}"}
```

Learn more

For a list of all `sam local invoke` options, see sam local invoke.

For a demo of using `sam local`, see AWS SAM for local development. Testing AWS Cloud resources from local development environments in the *Serverless Land Sessions with SAM series on YouTube*.

Using sam local start-api

Use the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam local start-api` subcommand to run your AWS Lambda functions locally and test through a local HTTP server host. This type of test is helpful for Lambda functions that are invoked by an Amazon API Gateway endpoint.

- For an introduction to the AWS SAM CLI, see [What is the AWS SAM CLI?](#).

- For a list of `sam local start-api` command options, see [sam local start-api](#).

- For an example of using `sam local start-api` during a typical development workflow, see [Step 7: (Optional) Test your application locally](#).

Prerequisites

To use `sam local start-api`, install the AWS SAM CLI by completing the following:

- [AWS SAM prerequisites](#).

- [Installing the AWS SAM CLI](#).

Before using `sam local start-api`, we recommend a basic understanding of the following:

- [Configuring the AWS SAM CLI](#).

- [Using sam init](#).

- [Using sam build](#).

- [Using sam deploy](#).

Using sam local start-api

When you run `sam local start-api`, the AWS SAM CLI assumes that your current working directory is your project's root directory. The AWS SAM CLI will first look for a `template.[yaml|yml]` file within a `.aws-sam` subfolder. If not found, the AWS SAM CLI will look for a `template.[yaml|yml]` file within your current working directory.

To start a local HTTP server

1. From the root directory of your project, run the following:

```
$ sam local start-api <options>
```

2. The AWS SAM CLI builds your Lambda functions in a local Docker container. It then outputs the local address of your HTTP server endpoint. The following is an example:

```
$ sam local start-api

Initializing the lambda functions containers.
Local image is up-to-date
Using local image: public.ecr.aws/lambda/python:3.9-rapid-x86_64.

Mounting /Users/.../sam-app/.aws-sam/build/HelloWorldFunction as /var/
task:ro,delegated, inside runtime container
Containers Initialization is done.
Mounting HelloWorldFunction at http://127.0.0.1:3000/hello [GET]
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. If you used sam build before running local
 commands, you will need to re-run sam build for the changes to be picked up. You
 only need to restart SAM CLI if you update your AWS SAM template
2023-04-12 14:41:05 WARNING: This is a development server. Do not use it in a
 production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:3000
```

3. You can invoke your Lambda function through the browser or command prompt. The following is an example:

```
sam-app$ curl http://127.0.0.1:3000/hello
{"message": "Hello world!"}%
```

4. When you make changes to your Lambda function code, consider the following to refresh your local HTTP server:

    *   If your application doesn't have a `.aws-sam` directory and your function uses an interpreted language, the AWS SAM CLI will automatically update your function by creating a new container and hosting it.

    *   If your application does have a `.aws-sam` directory, you need to run `sam build` to update your function. Then run `sam local start-api` again to host the function.

- If your function uses a compiled language or if your project requires complex packaging support, run your own build solution to update your function. Then run `sam local start-api` again to host the function.

Lambda functions that use Lambda authorizers

> ⓘ **Note**
>
> This feature is new in AWS SAM CLI version 1.80.0. To upgrade, see Upgrading the AWS SAM CLI.

For Lambda functions that use Lambda authorizers, the AWS SAM CLI will automatically invoke your Lambda authorizer before invoking your Lambda function endpoint.

The following is an example of starting a local HTTP server for a function that uses a Lambda authorizer:

```
$ sam local start-api
2023-04-17 15:02:13 Attaching import module proxy for analyzing dynamic imports

AWS SAM CLI does not guarantee 100% fidelity between authorizers locally
and authorizers deployed on AWS. Any application critical behavior should
be validated thoroughly before deploying to production.

Testing application behaviour against authorizers deployed on AWS can be done using the
 sam sync command.

Mounting HelloWorldFunction at http://127.0.0.1:3000/authorized-request [GET]
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. If you used sam build before running local commands, you will
 need to re-run sam build for the changes to be picked up. You only need to restart SAM
 CLI if you update your AWS SAM template
2023-04-17 15:02:13 WARNING: This is a development server. Do not use it in a
 production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:3000
2023-04-17 15:02:13 Press CTRL+C to quit
```

When you invoke your Lambda function endpoint through the local HTTP server, the AWS SAM CLI first invokes your Lambda authorizer. If authorization is successful, the AWS SAM CLI will invoke your Lambda function endpoint. The following is an example:

```
$ curl http://127.0.0.1:3000/authorized-request --header "header:my_token"
{"message": "from authorizer"}%

Invoking app.authorizer_handler (python3.8)
Local image is up-to-date
Using local image: public.ecr.aws/lambda/python:3.8-rapid-x86_64.

Mounting /Users/.../sam-app/... as /var/task:ro,delegated, inside runtime container
START RequestId: 38d3b472-a2c8-4ea6-9a77-9b386989bef0 Version: $LATEST
END RequestId: 38d3b472-a2c8-4ea6-9a77-9b386989bef0
REPORT RequestId: 38d3b472-a2c8-4ea6-9a77-9b386989bef0    Init Duration: 1.08 ms
 Duration: 628.26 msBilled Duration: 629 ms    Memory Size: 128 MB    Max Memory Used:
 128 MB
Invoking app.request_handler (python3.8)
Using local image: public.ecr.aws/lambda/python:3.8-rapid-x86_64.

Mounting /Users/.../sam-app/... as /var/task:ro,delegated, inside runtime container
START RequestId: fdc12255-79a3-4365-97e9-9459d06446ff Version: $LATEST
END RequestId: fdc12255-79a3-4365-97e9-9459d06446ff
REPORT RequestId: fdc12255-79a3-4365-97e9-9459d06446ff    Init Duration: 0.95 ms
 Duration: 659.13 msBilled Duration: 660 ms    Memory Size: 128 MB    Max Memory Used:
 128 MB
No Content-Type given. Defaulting to 'application/json'.
2023-04-17 15:03:03 127.0.0.1 - - [17/Apr/2023 15:03:03] "GET /authorized-request
 HTTP/1.1" 200 -
```

Options

Continuously reuse containers to speed up local function invokes

By default, the AWS SAM CLI creates a new container each time your function is invoked through the local HTTP server. Use the `--warm-containers` option to automatically reuse your container for function invokes. This speeds up the time it takes for the AWS SAM CLI to prepare your Lambda function for local invocation. You can customize this option further by providing the `eager` or `lazy` argument.

- `eager` – Containers for all functions are loaded at startup and persist between invocations.

- `lazy` – Containers are only loaded when each function is first invoked. They then persist for additional invocations.

The following is an example:

```
$ sam local start-api --warm-containers eager
```

When using `--warm-containers` and modifying your Lambda function code:

- If your application has a `.aws-sam` directory, run `sam build` to update your function code in your application's build artifacts.
- When a code change is detected, the AWS SAM CLI automatically shuts down the Lambda function container.
- When you invoke the function again, the AWS SAM CLI automatically creates a new container.

Specify a container image to use for your Lambda functions

By default, the AWS SAM CLI uses Lambda base images from Amazon Elastic Container Registry (Amazon ECR) to invoke your functions locally. Use the `--invoke-image` option to reference a custom container image. The following is an example:

```
$ sam local start-api --invoke-image public.ecr.aws/sam/emu-python3.8
```

You can specify the function to use with the custom container image. The following is an example:

```
$ sam local start-api --invoke-image Function1=amazon/aws/sam-cli-emulation-image-python3.8
```

Specify a template to locally test

To specify a template for the AWS SAM CLI to reference, use the `--template` option. The AWS SAM CLI will load just that AWS SAM template and the resources it points to. The following is an example:

```
$ sam local start-api --template myTemplate.yaml
```

Specify the host development environment of your Lambda function

By default, the `sam local start-api` subcommand creates an HTTP server using `localhost` with IP address `127.0.0.1`. You can customize these values if your local development environment is isolated from your local machine.

Use the `--container-host` option to specify a host. The following is an example:

```
$ sam local start-api --container-host host.docker.internal
```

Use the `--container-host-interface` option to specify the IP address of the host network that container ports should bind to. The following is an example:

```
$ sam local start-api --container-host-interface 0.0.0.0
```

Best practices

If your application has a `.aws-sam` directory from running `sam build`, be sure to run `sam build` every time you update your function code. Then, run `sam local start-api` to locally test your updated function code.

Local testing is a great solution for quick development and testing before deploying to the cloud. However, local testing doesn't validate everything, such as permissions between your resources in the cloud. As much as possible, test your applications in the cloud. We recommend using `sam` `sync` to speed up your cloud testing workflows.

Learn more

For a list of all `sam local start-api` options, see sam local start-api.

Using sam local start-lambda

Use the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam local start-lambda` subcommand to invoke your AWS Lambda function through the AWS Command Line Interface (AWS CLI) or SDKs. This command starts a local endpoint that emulates AWS Lambda.

- For an introduction to the AWS SAM CLI, see What is the AWS SAM CLI?.

- For a list of `sam local start-lambda` command options, see sam local start-lambda.

Prerequisites

To use `sam local start-lambda`, install the AWS SAM CLI by completing the following:

- [AWS SAM prerequisites](#).
- [Installing the AWS SAM CLI](#).

Before using `sam local start-lambda`, we recommend a basic understanding of the following:

- [Configuring the AWS SAM CLI](#).
- [Using sam init](#).
- [Using sam build](#).
- [Using sam deploy](#).

Using sam local start-lambda

When you run `sam local start-lambda`, the AWS SAM CLI assumes that your current working directory is your project's root directory. The AWS SAM CLI will first look for a `template.[yaml|yml]` file within a `.aws-sam` subfolder. If not found, the AWS SAM CLI will look for a `template.[yaml|yml]` file within your current working directory.

To use sam local start-lambda

1. From the root directory of your project, run the following:

   ```
   $ sam local start-lambda <options>
   ```

2. The AWS SAM CLI builds your Lambda functions in a local Docker container. It then outputs the local address to your HTTP server endpoint. The following is an example:

   ```
   $ sam local start-lambda
   Initializing the lambda functions containers.
   Local image is up-to-date
   Using local image: public.ecr.aws/lambda/python:3.9-rapid-x86_64.

   Mounting /Users/.../sam-app/hello_world as /var/task:ro,delegated, inside runtime
     container
   Containers Initialization is done.
   ```

```
Starting the Local Lambda Service. You can now invoke your Lambda Functions defined
 in your template through the endpoint.
2023-04-13 07:25:43 WARNING: This is a development server. Do not use it in a
 production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:3001
2023-04-13 07:25:43 Press CTRL+C to quit
```

3. Use the AWS CLI or SDKs to invoke your Lambda function locally.

    The following is an example using the AWS CLI:

    ```
    $ aws lambda invoke --function-name "HelloWorldFunction" --endpoint-
    url "http://127.0.0.1:3001" --no-verify-ssl out.txt

    StatusCode: 200
    (END)
    ```

    The following is an example using the AWS SDK for Python:

    ```
    import boto3
    from botocore.config import Config
    from botocore import UNSIGNED

    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=1,
                                               retries={'max_attempts': 0}
                                               )
                                 )
    lambda_client.invoke(FunctionName="HelloWorldFunction")
    ```

Options

Specify a template

To specify a template for the AWS SAM CLI to reference, use the `--template` option. The AWS
SAM CLI will load just that AWS SAM template and the resources it points to. The following is an
example:

```
$ sam local start-lambda --template myTemplate.yaml
```

Best practices

If your application has a `.aws-sam` directory from running `sam build`, be sure to run `sam build` every time you update your function code. Then, run `sam local start-lambda` to locally test your updated function code.

Local testing is a great solution for quick development and testing before deploying to the cloud. However, local testing doesn't validate everything, such as permissions between your resources in the cloud. As much as possible, test your applications in the cloud. We recommend using sam sync to speed up your cloud testing workflows.

Learn more

For a list of all `sam local start-lambda` options, see sam local start-lambda.

Using sam remote invoke

Use the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam remote invoke` command to interact with supported AWS resources in the AWS Cloud. You can use `sam remote invoke` to invoke the following resources:

- **Amazon Kinesis Data Streams** – Send data records to Kinesis Data Streams applications.
- **AWS Lambda** – Invoke and pass events to your Lambda functions.
- **Amazon Simple Queue Service (Amazon SQS)** – Send messages to Amazon SQS queues.
- **AWS Step Functions** – Invoke Step Functions state machines to start execution.

For an introduction to the AWS SAM CLI, see What is the AWS SAM CLI?.

For an example of using `sam remote invoke` during a typical development workflow, see Step 5: Interact with your function in the AWS Cloud.

Topics

- Using the sam remote invoke command
- Using sam remote invoke command options
- Configure your project configuration file

- [Examples](#)
- [Related links](#)

Prerequisites

To use `sam remote invoke`, install the AWS SAM CLI by completing the following:

- [AWS SAM prerequisites](#).
- [Installing the AWS SAM CLI](#).

We also recommend upgrading to the latest version of the AWS SAM CLI. To learn more, see [Upgrading the AWS SAM CLI](#).

Before using `sam remote invoke`, we recommend a basic understanding of the following:

- [Configuring the AWS SAM CLI](#).
- [Using sam init](#).
- [Using sam build](#).
- [Using sam deploy](#).
- [Using sam sync](#).

Using the sam remote invoke command

Before using this command, your resource must be deployed to the AWS Cloud.

Use the following command structure and run from your project's root directory:

```
$ sam remote invoke <arguments> <options>
```

> **ⓘ Note**
>
> This page will show options being provided at the command prompt. You can also configure options in your project's configuration file instead of passing them at the command prompt. To learn more, [Configure project settings](#).

For a description of `sam remote invoke` arguments and options, see [sam remote invoke](#).

Using with Kinesis Data Streams

You can send data records to a Kinesis Data Streams application. The AWS SAM CLI will send your data record and return a shard ID and sequence number. The following is an example:

```
$ sam remote invoke KinesisStream --stack-name kinesis-example --event hello-world

 Putting record to Kinesis data stream KinesisStream

 Auto converting value 'hello-world' into JSON '"hello-world"'. If you don't want auto-
conversion, please provide
 a JSON string as event

 {
   "ShardId": "shardId-000000000000",
   "SequenceNumber": "49646251411914806775980850790050483811301135051202232322"
 }%
```

To send a data record

1. Provide a resource ID value as an argument for your Kinesis Data Streams application. For information on valid resource IDs, see [Resource ID](#).

2. Provide the data record as an event to send to your Kinesis Data Streams application. You can provide the event at the command line using the `--event` option, or from a file using `--event-file`. If you don't provide an event, the AWS SAM CLI sends an empty event.

Using with Lambda functions

You can invoke a Lambda function in the cloud and pass an empty event or provide an event at the command line or from a file. The AWS SAM CLI will invoke your Lambda function and return its response. The following is an example:

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app

Invoking Lambda Function HelloWorldFunction
START RequestId: d5ef494b-5f45-4086-86fd-d7322fa1a1f9 Version: $LATEST
END RequestId: d5ef494b-5f45-4086-86fd-d7322fa1a1f9
REPORT RequestId: d5ef494b-5f45-4086-86fd-d7322fa1a1f9  Duration: 6.62 ms        Billed
 Duration: 7 ms      Memory Size: 128 MB     Max Memory Used: 67 MB  Init Duration:
  164.06 ms
```

```
{"statusCode":200,"body":"{\"message\":\"hello world\"}"}%
```

To invoke a Lambda function

1. Provide a resource ID value as an argument for your Lambda function. For information on valid resource IDs, see Resource ID.

2. Provide an event to send to your Lambda function. You can provide the event at the command line using the `--event` option, or from a file using `--event-file`. If you don't provide an event, the AWS SAM CLI sends an empty event.

Lambda functions configured with response streaming

The `sam remote invoke` command supports Lambda functions that are configured to stream responses. You can configure a Lambda function to stream responses using the `FunctionUrlConfig` property in your AWS SAM templates. When you use `sam remote invoke`, the AWS SAM CLI will automatically detect your Lambda configuration and invoke with response streaming.

For an example, see Invoke a Lambda function configured to stream responses.

Pass shareable test events to a Lambda function in the cloud

Shareable test events are test events that you can share with others in the same AWS account. To learn more, see Shareable test events in the *AWS Lambda Developer Guide*.

Accessing and managing shareable test events

You can use the AWS SAM CLI `sam remote test-event` command to access and manage shareable test events. For example, you can use `sam remote test-event` to do the following:

*   Retrieve shareable test events from the Amazon EventBridge schema registry.

*   Modify shareable test events locally and upload them to the EventBridge schema registry.

*   Delete shareable test events from the EventBridge schema registry.

To learn more, see Using sam remote test-event.

Pass a shareable test event to a Lambda function in the cloud

To pass a shareable test event from the EventBridge schema registry to your Lambda function in the cloud, use the `--test-event-name` option and provide the name of the shareable test event. The following is an example:

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --test-event-name demo-
event
```

If you save the shareable test event locally, you can use the `--event-file` option and provide the file path and name of the local test event. The following is an example:

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --event-file demo-
event.json
```

Using with Amazon SQS

You can send messages to Amazon SQS queues. The AWS SAM CLI returns the following:

- Message ID
- MD5 of message body
- Response metadata

The following is an example:

```
$ sam remote invoke MySqsQueue --stack-name sqs-example -event hello

Sending message to SQS queue MySqsQueue

{
  "MD5OfMessageBody": "5d41402abc4b2a76b9719d911017c592",
  "MessageId": "05c7af65-9ae8-4014-ae28-809d6d8ec652"
}%
```

To send a message

1. Provide a resource ID value as an argument for the Amazon SQS queue. For information on valid resource IDs, see Resource ID.

2.   Provide an event to send to your Amazon SQS queue. You can provide the event at the
     command line using the `--event` option, or from a file using `--event-file`. If you don't
     provide an event, the AWS SAM CLI sends an empty event.

Using with Step Functions

You can invoke a Step Functions state machine to start execution. The AWS SAM CLI will wait for
the state machine workflow to complete and return an output of the last step in the execution. The
following is an example:

```
$ sam remote invoke HelloWorldStateMachine --stack-name state-machine-example --
event '{"is_developer": true}'

Invoking Step Function HelloWorldStateMachine

"Hello Developer World"%
```

To invoke a state machine

1.   Provide a resource ID value as an argument for the Step Functions state machine. For
     information on valid resource IDs, see Resource ID.

2.   Provide an event to send to your state machine. You can provide the event at the command
     line using the `--event` option, or from a file using `--event-file`. If you don't provide an
     event, the AWS SAM CLI sends an empty event.

Using sam remote invoke command options

This section covers some of the main options that you can use with the `sam remote invoke`
command. For a full list of options, see sam remote invoke.

Pass an event to your resource

Use the following options to pass events to your resources in the cloud:

- `--event` – Pass an event at the command line.

- `--event-file` – Pass an event from a file.

Lambda examples

Use `--event` to pass an event at the command line as a string value:

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --event '{"message":
 "hello!"}'

Invoking Lambda Function HelloWorldFunction


START RequestId: b992292d-1fac-4aa2-922a-c9dc5c6fceab Version: $LATEST
END RequestId: b992292d-1fac-4aa2-922a-c9dc5c6fceab
REPORT RequestId: b992292d-1fac-4aa2-922a-c9dc5c6fceab  Duration: 16.41 ms      Billed
 Duration: 17 ms  Memory Size: 128 MB     Max Memory Used: 67 MB  Init Duration: 185.96
 ms
{"statusCode":200,"body":"{\"message\":\"hello!\"}"}%
```

Use `--event-file` to pass an event from a file and provide the path to the file:

```
$ cat event.json

{"message": "hello from file"}%

$ sam remote invoke HelloWorldFunction --stack-name sam-app --event-file event.json


Invoking Lambda Function HelloWorldFunction


START RequestId: 3bc71f7d-153a-4b1e-8c9a-901d91b1bec9 Version: $LATEST
END RequestId: 3bc71f7d-153a-4b1e-8c9a-901d91b1bec9
REPORT RequestId: 3bc71f7d-153a-4b1e-8c9a-901d91b1bec9  Duration: 21.15 ms      Billed
 Duration: 22 ms  Memory Size: 128 MB     Max Memory Used: 67 MB
{"statusCode":200,"body":"{\"message\":\"hello from file\"}"}%
```

Pass an event using `stdin`:

```
$ cat event.json

{"message": "hello from file"}%

$ cat event.json | sam remote invoke HelloWorldFunction --stack-name sam-app --event-
file -
```

```
Reading event from stdin (you can also pass it from file with --event-file)

Invoking Lambda Function HelloWorldFunction

START RequestId: 85ecc902-8ad0-4a2b-a8c8-9bb4f65f5a7a Version: $LATEST
END RequestId: 85ecc902-8ad0-4a2b-a8c8-9bb4f65f5a7a
REPORT RequestId: 85ecc902-8ad0-4a2b-a8c8-9bb4f65f5a7a  Duration: 1.36 ms      Billed
 Duration: 2 ms   Memory Size: 128 MB       Max Memory Used: 67 MB
{"statusCode":200,"body":"{\"message\":\"hello from file\"}"}%
```

Configure the AWS SAM CLI response output

When you invoke a supported resource with `sam remote invoke`, the AWS SAM CLI returns a response that contains the following:

- **Request metadata** – Metadata associated with the request. This includes a request ID and request start time.
- **Resource response** – The response from your resource after being invoked in the cloud.

You can use the `--output` option to configure the AWS SAM CLI output response. The following option values are available:

- `json` – Metadata and resource response are returned in a JSON structure. The response contains the full SDK output.
- `text` – Metadata is returned in text structure. The resource response is returned in the output format of the resource.

The following is an example of a `json` output:

```
$ sam remote invoke --stack-name sam-app --output json

Invoking Lambda Function HelloWorldFunction


{
  "ResponseMetadata": {
    "RequestId": "3bdf9a30-776d-4a90-94a6-4cccc0fc7b41",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
```

```
      "date": "Mon, 19 Jun 2023 17:15:46 GMT",
      "content-type": "application/json",
      "content-length": "57",
      "connection": "keep-alive",
      "x-amzn-requestid": "3bdf9a30-776d-4a90-94a6-4cccc0fc7b41",
      "x-amzn-remapped-content-length": "0",
      "x-amz-executed-version": "$LATEST",
      "x-amz-log-result":
 "U1RBUlQgUmVxdWVzdElkOiAzYmRmOWEzMC03NzZkLTRhOTAtOTRhNi00Y2NjYzBmYzdiNDEgVmVyc2lvbjogJExBVEVVTV
      "x-amzn-trace-id":
 "root=1-64908d42-17dab270273fcc6b527dd6b8;sampled=0;lineage=2301f8dc:0"
    },
    "RetryAttempts": 0
  },
  "StatusCode": 200,
  "LogResult":
 "U1RBUlQgUmVxdWVzdElkOiAzYmRmOWEzMC03NzZkLTRhOTAtOTRhNi00Y2NjYzBmYzdiNDEgVmVyc2lvbjogJExBVEVVTV
  "ExecutedVersion": "$LATEST",
  "Payload": "{\"statusCode\":200,\"body\":\"{\\\"message\\\":\\\"hello world\\\"}\"}"
}%
```

When you specify a `json` output, the entire response is returned to `stdout`. The following is an example:

```
$ sam remote invoke --stack-name sam-app --output json 1> stdout.log

Invoking Lambda Function HelloWorldFunction


$ cat stdout.log

{
  "ResponseMetadata": {
    "RequestId": "d30d280f-8188-4372-bc94-ce0f1603b6bb",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "date": "Mon, 19 Jun 2023 17:35:56 GMT",
      "content-type": "application/json",
      "content-length": "57",
      "connection": "keep-alive",
      "x-amzn-requestid": "d30d280f-8188-4372-bc94-ce0f1603b6bb",
      "x-amzn-remapped-content-length": "0",
```

```
            "x-amz-executed-version": "$LATEST",
            "x-amz-log-result":
 "U1RBUlQgUmVxdWVzdElkOiBkMzBkMjgwZi04MTg4LTQzNzItYmM5NC1jZTBmMTYwM2I2YmIgVmVyc2lvbjogJExBVEVTTV
            "x-amzn-trace-id":
 "root=1-649091fc-771473c7778689627a6122b7;sampled=0;lineage=2301f8dc:0"
        },
        "RetryAttempts": 0
    },
    "StatusCode": 200,
    "LogResult":
 "U1RBUlQgUmVxdWVzdElkOiBkMzBkMjgwZi04MTg4LTQzNzItYmM5NC1jZTBmMTYwM2I2YmIgVmVyc2lvbjogJExBVEVTTV
    "ExecutedVersion": "$LATEST",
    "Payload": "{\"statusCode\":200,\"body\":\"{\\\\"message\\\":\\\"hello world\\\"}\"}"
}%
```

The following is an example of a `text` output:

```
$ sam remote invoke --stack-name sam-app --output text

Invoking Lambda Function HelloWorldFunction


START RequestId: 4dbacc43-1ec6-47c2-982b-9dc4620144d6 Version: $LATEST
END RequestId: 4dbacc43-1ec6-47c2-982b-9dc4620144d6
REPORT RequestId: 4dbacc43-1ec6-47c2-982b-9dc4620144d6  Duration: 9.13 ms       Billed
 Duration: 10 ms  Memory Size: 128 MB     Max Memory Used: 67 MB  Init Duration: 165.50
  ms
{"statusCode":200,"body":"{\"message\":\"hello world\"}"}%
```

When you specify a `text` output, the Lambda function runtime output (for example, logs) is returned to `stderr`. The Lambda function payload is returned to `stdout`. The following is an example:

```
$ sam remote invoke --stack-name sam-app --output text 2> stderr.log

{"statusCode":200,"body":"{\"message\":\"hello world\"}"}%


$ cat stderr.log

Invoking Lambda Function HelloWorldFunction
```

```
START RequestId: 82273c3b-aa3a-4d16-8f1c-1d2ad3ace891 Version: $LATEST
END RequestId: 82273c3b-aa3a-4d16-8f1c-1d2ad3ace891
REPORT RequestId: 82273c3b-aa3a-4d16-8f1c-1d2ad3ace891  Duration: 40.62 ms     Billed
 Duration: 41 ms  Memory Size: 128 MB     Max Memory Used: 68 MB


$ sam remote invoke --stack-name sam-app --output text 1> stdout.log


Invoking Lambda Function HelloWorldFunction


START RequestId: 74acaa9f-5b80-4a5c-b3b8-ffaccb84cbbd Version: $LATEST
END RequestId: 74acaa9f-5b80-4a5c-b3b8-ffaccb84cbbd
REPORT RequestId: 74acaa9f-5b80-4a5c-b3b8-ffaccb84cbbd  Duration: 2.31 ms     Billed
 Duration: 3 ms   Memory Size: 128 MB     Max Memory Used: 67 MB


$ cat stdout.log


{"statusCode":200,"body":"{\"message\":\"hello world\"}"}%
```

Customize Boto3 parameters

For `sam remote invoke`, the AWS SAM CLI utilizes the AWS SDK for Python (Boto3) to interact with your resources in the cloud. You can use the `--parameter` option to customize Boto3 parameters. For a list of supported parameters that you can customize, see [--parameter](#).

Examples

Invoke a Lambda function to validate parameter values and verify permissions:

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --
parameter InvocationType="DryRun"
```

Use the `--parameter` option multiple times in a single command to provide multiple parameters:

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --
parameter InvocationType="Event" --parameter LogType="None"
```

Other options

For a full list of `sam remote invoke` options, see [sam remote invoke](#).

Configure your project configuration file

To configure `sam remote invoke` in your configuration file, use `remote_invoke` in your table. The following is an example of a `samconfig.toml` file that configures default values for the `sam remote invoke` command.

```
...
version =0.1

[default]
...
[default.remote_invoke.parameters]
stack_name = "cloud-app"
event = '{"message": "Hello!"}'
```

Examples

For a basic example of using `sam remote invoke`, see [Testing AWS Lambda functions with AWS SAM remote](#) in the *AWS Compute Blog*.

Kinesis Data Streams examples

Basic examples

Send a data record to a Kinesis Data Streams application from a file. The Kinesis Data Streams application is identified by providing an ARN for the resource ID:

```
$ sam remote invoke arn:aws:kinesis:us-west-2:01234567890:stream/kinesis-example-
KinesisStream-BgnLcAey4xUQ --event-file event.json
```

Send an event provided at the command line to a Kinesis Data Streams application:

```
$ sam remote invoke KinesisStream --stack-name kinesis-example --event hello-world

Putting record to Kinesis data stream KinesisStream

Auto converting value 'hello-world' into JSON '"hello-world"'. If you don't want auto-
conversion, please provide
a JSON string as event

{
```

```
    "ShardId": "shardId-000000000000",
    "SequenceNumber": "49646251411914806775980903986194508740483329854174920706"
}%
```

Obtain the physical ID of the Kinesis Data Streams application. Then, provide an event at the command line:

```
$ sam list resources --stack-name kinesis-example --output json

[
  {
    "LogicalResourceId": "KinesisStream",
    "PhysicalResourceId": "kinesis-example-KinesisStream-ZgnLcQey4xUQ"
  }
]

$ sam remote invoke kinesis-example-KinesisStream-ZgnLcQey4xUQ --event hello

Putting record to Kinesis data stream KinesisStream

Auto converting value 'hello' into JSON '"hello"'. If you don't want auto-conversion,
 please provide a JSON
string as event

{
  "ShardId": "shardId-000000000000",
  "SequenceNumber": "49646251411914806775980904340716841045751814812900261890"
}%
```

Provide a JSON string at the command line as an event:

```
$ sam remote invoke KinesisStream --stack-name kinesis-example --event '{"method":
 "GET", "body": ""}'

Putting record to Kinesis data stream KinesisStream

{
  "ShardId": "shardId-000000000000",
  "SequenceNumber": "49646251411914806775980904492868617924990209230536441858"
}%
```

Send an empty event to the Kinesis Data Streams application:

```
$ sam remote invoke KinesisStream --stack-name kinesis-example

Putting record to Kinesis data stream KinesisStream

{
  "ShardId": "shardId-000000000000",
  "SequenceNumber": "49646251411914806775980904866469008589597168190416224258"
}%
```

Return the AWS SAM CLI response in JSON format:

```
$ sam remote invoke KinesisStream --stack-name kinesis-example --event '{"hello":
 "world"}' --output json

Putting record to Kinesis data stream KinesisStream

{
  "ShardId": "shardId-000000000000",
  "SequenceNumber": "49646251411914806775980905078409420803696667195489648642",
  "ResponseMetadata": {
    "RequestId": "ebbbd307-3e9f-4431-b67c-f0715e9e353e",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "x-amzn-requestid": "ebbbd307-3e9f-4431-b67c-f0715e9e353e",
      "x-amz-id-2": "Q3yBcgTwtPaQTV26IKclbECmZikUYOzKY+CzcxA84ZHgCkc5T2N/
ITWg6RPOQcWw8Gn0tNPcEJBEHyVVqboJAPgCritqsvCu",
      "date": "Thu, 09 Nov 2023 18:13:10 GMT",
      "content-type": "application/x-amz-json-1.1",
      "content-length": "110"
    },
    "RetryAttempts": 0
  }
}%
```

Return the JSON output to stdout:

```
$ sam remote invoke KinesisStream --stack-name kinesis-example --event '{"hello":
 "world"}' --output json 1> stdout.log

Putting record to Kinesis data stream KinesisStream
```

```
$ cat stdout.log
{
  "ShardId": "shardId-000000000000",
  "SequenceNumber": "49646251411914806775980906397778675950399883490067742274",
  "ResponseMetadata": {
    "RequestId": "f4290006-d84b-b1cd-a9ee-28306eeb2939",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "x-amzn-requestid": "f4290006-d84b-b1cd-a9ee-28306eeb2939",
      "x-amz-id-2": "npCqz
+IBKpoL4sQ1ClbUmxuJlbeA24Fx1UgpIrS6mm2NoIeV2qdZSN5AhNurdssykXajBrXaC9anMhj2eG/h7Hnbf
+bPuotU",
      "date": "Thu, 09 Nov 2023 18:33:26 GMT",
      "content-type": "application/x-amz-json-1.1",
      "content-length": "110"
    },
    "RetryAttempts": 0
  }
}%
```

Lambda examples

Basic examples

Invoke a Lambda function by providing the ARN as a resource ID:

```
$ sam remote invoke arn:aws:lambda:us-west-2:012345678910:function:sam-app-
HelloWorldFunction-ohRFEn2RuAvp
```

Invoke a Lambda function by providing the logical ID as a resource ID:

You must also provide the AWS CloudFormation stack name using the `--stack-name` option. The following is an example:

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app
```

If your application contains a single Lambda function, you don't have to specify it's logical ID. You can provide the `--stack-name` option only. The following is an example:

```
$ sam remote invoke --stack-name sam-app
```

Invoke a Lambda function by providing the physical ID as a resource ID:

The physical ID gets created when you deploy using AWS CloudFormation.

```
$ sam remote invoke sam-app-HelloWorldFunction-TZvxQRFNv0k4
```

Invoke a Lambda function of a child stack:

For this example, our application contains the following directory structure:

```
lambda-example
### childstack
#   ### function
#   #   ### __init__.py
#   #   ### app.py
#   #   ### requirements.txt
#   ### template.yaml
### events
#   ### event.json
### samconfig.toml
### template.yaml
```

To invoke the Lambda function of our `childstack`, we run the following:

```
$ sam remote invoke ChildStack/HelloWorldFunction --stack-name lambda-example

Invoking Lambda Function HelloWorldFunction

START RequestId: 207a864b-e67c-4307-8478-365b004d4bcd Version: $LATEST
END RequestId: 207a864b-e67c-4307-8478-365b004d4bcd
REPORT RequestId: 207a864b-e67c-4307-8478-365b004d4bcd  Duration: 1.27 ms        Billed
 Duration: 2 ms   Memory Size: 128 MB     Max Memory Used: 36 MB  Init Duration: 111.07
 ms
{"statusCode": 200, "body": "{\"message\": \"Hello\", \"received_event\": {}}"}%
```

Invoke a Lambda function configured to stream responses

In this example, we use the AWS SAM CLI to initialize a new serverless application that contains a Lambda function configured to stream its response. We deploy our application to the AWS Cloud and use the `sam remote invoke` to interact with our function in the cloud.

We start by running the `sam init` command to create a new serverless application. We select the **Lambda Response Streaming** quick start template and name our application **lambda-streaming-nodejs-app**.

```
$ sam init

You can preselect a particular runtime or package type when using the `sam init`
experience.
Call `sam init --help` to learn more.

Which template source would you like to use?
        1 - AWS Quick Start Templates
        2 - Custom Template Location
Choice: 1

Choose an AWS Quick Start application template
        1 - Hello World Example
        ...
        9 - Lambda Response Streaming
        ...
        15 - Machine Learning
Template: 9

Which runtime would you like to use?
        1 - go (provided.al2)
        2 - nodejs18.x
        3 - nodejs16.x
Runtime: 2

Based on your selections, the only Package type available is Zip.
We will proceed to selecting the Package type as Zip.

Based on your selections, the only dependency manager available is npm.
We will proceed copying the template using npm.

Would you like to enable X-Ray tracing on the function(s) in your application?  [y/
N]: ENTER

Would you like to enable monitoring using CloudWatch Application Insights?
 For more info, please view https://docs.aws.amazon.com/AmazonCloudWatch/latest/
monitoring/cloudwatch-application-insights.html [y/N]: ENTER

Project name [sam-app]: lambda-streaming-nodejs-app

    -----------------------
    Generating application:
    -----------------------
```

```
      Name: lambda-streaming-nodejs-app
      Runtime: nodejs18.x
      Architectures: x86_64
      Dependency Manager: npm
      Application Template: response-streaming
      Output Directory: .
      Configuration file: lambda-streaming-nodejs-app/samconfig.toml

      Next steps can be found in the README file at lambda-streaming-nodejs-app/
README.md


 Commands you can use next
 =========================
 [*] Create pipeline: cd lambda-streaming-nodejs-app && sam pipeline init --bootstrap
 [*] Validate SAM template: cd lambda-streaming-nodejs-app && sam validate
 [*] Test Function in the Cloud: cd lambda-streaming-nodejs-app && sam sync --stack-
name {stack-name} --watch
```

The AWS SAM CLI creates our project with the following structure:

```
lambda-streaming-nodejs-app
 ### README.md
 ### __tests__
 #    ### unit
 #        ### index.test.js
 ### package.json
 ### samconfig.toml
 ### src
 #    ### index.js
 ### template.yaml
```

The following is an example of our Lambda function code:

```
exports.handler = awslambda.streamifyResponse(
    async (event, responseStream, context) => {
      const httpResponseMetadata = {
        statusCode: 200,
        headers: {
          "Content-Type": "text/html",
          "X-Custom-Header": "Example-Custom-Header"
        }
      };
```

```
    responseStream = awslambda.HttpResponseStream.from(responseStream,
 httpResponseMetadata);
    // It's recommended to use a `pipeline` over the `write` method for more complex
 use cases.
    // Learn more: https://docs.aws.amazon.com/lambda/latest/dg/configuration-
 response-streaming.html
    responseStream.write("<html>");
    responseStream.write("<p>First write!</p>");

    responseStream.write("<h1>Streaming h1</h1>");
    await new Promise(r => setTimeout(r, 1000));
    responseStream.write("<h2>Streaming h2</h2>");
    await new Promise(r => setTimeout(r, 1000));
    responseStream.write("<h3>Streaming h3</h3>");
    await new Promise(r => setTimeout(r, 1000));

    // Long strings will be streamed
    const loremIpsum1 = "Lorem ipsum dolor sit amet, consectetur adipiscing elit.
 Quisque vitae mi tincidunt tellus ultricies dignissim id et diam. Morbi pharetra eu
 nisi et finibus. Vivamus diam nulla, vulputate et nisl cursus, pellentesque vehicula
 libero. Cras imperdiet lorem ante, non posuere dolor sollicitudin a. Vestibulum ipsum
 lacus, blandit nec augue id, lobortis dictum urna. Vestibulum ante ipsum primis in
 faucibus orci luctus et ultrices posuere cubilia curae; Morbi auctor orci eget tellus
 aliquam, non maximus massa porta. In diam ante, pulvinar aliquam nisl non, elementum
 hendrerit sapien. Vestibulum massa nunc, mattis non congue vitae, placerat in quam.
 Nam vulputate lectus metus, et dignissim erat varius a.";
    responseStream.write(`<p>${loremIpsum1}</p>`);
    await new Promise(r => setTimeout(r, 1000));

    responseStream.write("<p>DONE!</p>");
    responseStream.write("</html>");
    responseStream.end();
  }
);
```

The following is an example of our `template.yaml` file. Response streaming for our Lambda function is configured using the `FunctionUrlConfig` property.

```yaml
AWSTemplateFormatVersion: '2010-09-09'
 Transform: AWS::Serverless-2016-10-31

 Description: >
```

```
   Sample SAM Template for lambda-streaming-nodejs-app

Resources:
  StreamingFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: src/
      Handler: index.handler
      Runtime: nodejs18.x
      Architectures:
        - x86_64
      Timeout: 10
      FunctionUrlConfig:
        AuthType: AWS_IAM
        InvokeMode: RESPONSE_STREAM

Outputs:
  StreamingFunction:
    Description: "Streaming Lambda Function ARN"
    Value: !GetAtt StreamingFunction.Arn
  StreamingFunctionURL:
    Description: "Streaming Lambda Function URL"
    Value: !GetAtt StreamingFunctionUrl.FunctionUrl
```

Typically, you can use `sam build` and `sam deploy --guided` to build and deploy a production application. In this example, we'll assume a development environment and use the `sam sync` command to build and deploy our application.

> ⓘ **Note**
>
> The `sam sync` command is recommended for development environments. To learn more, see [Using sam sync](#).

Before running `sam sync`, we verify that our project is configured correctly in our `samconfig.toml` file. Most importantly, we verify the values for `stack_name` and `watch`. With these values specified in our configuration file, we don't have to provide them at the command line.

```
version = 0.1
```

```
[default]
[default.global.parameters]
stack_name = "lambda-streaming-nodejs-app"

[default.build.parameters]
cached = true
parallel = true

[default.validate.parameters]
lint = true

[default.deploy.parameters]
capabilities = "CAPABILITY_IAM"
confirm_changeset = true
resolve_s3 = true
s3_prefix = "lambda-streaming-nodejs-app"
region = "us-west-2"
image_repositories = []

[default.package.parameters]
resolve_s3 = true

[default.sync.parameters]
watch = true

[default.local_start_api.parameters]
warm_containers = "EAGER"

[default.local_start_lambda.parameters]
warm_containers = "EAGER"
```

Next, we run `sam sync` to build and deploy our application. Since the `--watch` option is configured in our configuration file, the AWS SAM CLI will build our application, deploy our application, and watch for changes.

```
$ sam sync

The SAM CLI will use the AWS Lambda, Amazon API Gateway, and AWS StepFunctions APIs to
upload your code
without

performing a CloudFormation deployment. This will cause drift in your CloudFormation
stack.
```

The sync command should only be used against a development stack.

Queued infra sync. Waiting for in progress code syncs to complete...

Starting infra sync.

Building codeuri:

/Users/.../lambda-streaming-nodejs-app/src runtime: nodejs18.x metadata: {}
architecture: x86_64 functions: StreamingFunction
package.json file not found. Continuing the build without dependencies.

Running NodejsNpmBuilder:CopySource

Build Succeeded

Successfully packaged artifacts and wrote output template to file /var/
folders/45/5ct135bx3fn2551_ptl5g6_80000gr/T/tmpavrzdhgp.
Execute the following command to deploy the packaged template
sam deploy --template-file /var/folders/45/5ct135bx3fn2551_ptl5g6_80000gr/T/
tmpavrzdhgp --stack-name <YOUR STACK NAME>

        Deploying with following values
        ===============================
        Stack name   : lambda-streaming-nodejs-app
        Region   : us-west-2
        Disable rollback   : False
        Deployment s3 bucket   : aws-sam-cli-managed-default-
samclisourcebucket-1a4x26zbcdkqr
        Capabilities   : ["CAPABILITY_NAMED_IAM",
"CAPABILITY_AUTO_EXPAND"]
        Parameter overrides   : {}
        Signing Profiles   : null

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

2023-06-20 12:11:16 - Waiting for stack create/update to complete

```
CloudFormation events from stack operations (refresh every 0.5 seconds)
-------------------------------------------------------------------------
ResourceStatus          ResourceType            LogicalResourceId
ResourceStatusReason
-------------------------------------------------------------------------
CREATE_IN_PROGRESS      AWS::CloudFormation::St  lambda-streaming-
Transformation
                        ack                      nodejs-app
succeeded
CREATE_IN_PROGRESS      AWS::IAM::Role           StreamingFunctionRole    -

CREATE_IN_PROGRESS      AWS::CloudFormation::St  AwsSamAutoDependencyLay   -

                        ack                      erNestedStack

CREATE_IN_PROGRESS      AWS::IAM::Role           StreamingFunctionRole    Resource
creation

Initiated
CREATE_IN_PROGRESS      AWS::CloudFormation::St  AwsSamAutoDependencyLay   Resource
creation
                        ack                      erNestedStack
Initiated
CREATE_COMPLETE         AWS::IAM::Role           StreamingFunctionRole    -

CREATE_COMPLETE         AWS::CloudFormation::St  AwsSamAutoDependencyLay   -

                        ack                      erNestedStack

CREATE_IN_PROGRESS      AWS::Lambda::Function    StreamingFunction        -

CREATE_IN_PROGRESS      AWS::Lambda::Function    StreamingFunction        Resource
creation

Initiated
CREATE_COMPLETE         AWS::Lambda::Function    StreamingFunction        -

CREATE_IN_PROGRESS      AWS::Lambda::Url         StreamingFunctionUrl     -

CREATE_IN_PROGRESS      AWS::Lambda::Url         StreamingFunctionUrl     Resource
creation

Initiated
```

```
CREATE_COMPLETE              AWS::Lambda::Url          StreamingFunctionUrl        -

CREATE_COMPLETE              AWS::CloudFormation::St    lambda-streaming-           -

                             ack                        nodejs-app


-----------------------------------------------------------------------------------

CloudFormation outputs from deployed stack
-----------------------------------------------------------------------------------
Outputs


-----------------------------------------------------------------------------------
Key                 StreamingFunction

Description         Streaming Lambda Function ARN

Value               arn:aws:lambda:us-west-2:012345678910:function:lambda-streaming-
nodejs-app-
StreamingFunction-gUmhO833A0vZ



Key                 StreamingFunctionURL

Description         Streaming Lambda Function URL

Value               https://wxgkcc2dyntgtrwhf2dgdcvylu0rnnof.lambda-url.us-
west-2.on.aws/
-----------------------------------------------------------------------------------



Stack creation succeeded. Sync infra completed.



Infra sync completed.
```

Now that our function is deployed to the cloud, we can use `sam remote invoke` to interact with our function. The AWS SAM CLI automatically detects that our function is configured for response streaming and immediately begins outputting a streamed response of our function in real time.

```
$ sam remote invoke StreamingFunction
```

```
Invoking Lambda Function StreamingFunction

{"statusCode":200,"headers":{"Content-Type":"text/html","X-Custom-Header":"Example-
Custom-Header"}}<html><p>First write!</p><h1>Streaming h1</h1><h2>Streaming h2</
h2><h3>Streaming h3</h3><p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
 Quisque vitae mi tincidunt tellus ultricies dignissim id et diam. Morbi pharetra eu
 nisi et finibus. Vivamus diam nulla, vulputate et nisl cursus, pellentesque vehicula
 libero. Cras imperdiet lorem ante, non posuere dolor sollicitudin a. Vestibulum ipsum
 lacus, blandit nec augue id, lobortis dictum urna. Vestibulum ante ipsum primis in
 faucibus orci luctus et ultrices posuere cubilia curae; Morbi auctor orci eget tellus
 aliquam, non maximus massa porta. In diam ante, pulvinar aliquam nisl non, elementum
 hendrerit sapien. Vestibulum massa nunc, mattis non congue vitae, placerat in quam.
 Nam vulputate lectus metus, et dignissim erat varius a.</p><p>DONE!</p></html>START
 RequestId: 1e4cdf04-60de-4769-b3a2-c1481982deb4 Version: $LATEST
 END RequestId: 1e4cdf04-60de-4769-b3a2-c1481982deb4
 REPORT RequestId: 1e4cdf04-60de-4769-b3a2-c1481982deb4  Duration: 4088.66 ms
 Billed Duration: 4089 ms        Memory Size: 128 MB     Max Memory Used: 68 MB  Init
 Duration: 168.45 ms
```

When we modify our function code, the AWS SAM CLI instantly detects and immediately deploys our changes. Here is an example of the AWS SAM CLI output after changes are made to our function code:

```
Syncing Lambda Function StreamingFunction...

 Building codeuri:

 /Users/.../lambda-streaming-nodejs-app/src runtime: nodejs18.x metadata: {}
 architecture:
 x86_64 functions: StreamingFunction

 package.json file not found. Continuing the build without dependencies.

 Running NodejsNpmBuilder:CopySource

 Finished syncing Lambda Function StreamingFunction.

 Syncing Layer StreamingFunctione9cfe924DepLayer...

 SyncFlow [Layer StreamingFunctione9cfe924DepLayer]: Skipping resource update as the
```

```
content didn't change

Finished syncing Layer StreamingFunctione9cfe924DepLayer.
```

We can now use `sam remote invoke` again to interact with our function in the cloud and test our changes.

SQS examples

Basic examples

Invoke an Amazon SQS queue by providing the ARN as a resource ID:

```
$ sam remote invoke arn:aws:sqs:us-west-2:01234567890:sqs-example-4DonhBsjsW1b --
event '{"hello": "world"}' --output json

Sending message to SQS queue MySqsQueue

{
  "MD5OfMessageBody": "49dfdd54b01cbcd2d2ab5e9e5ee6b9b9",
  "MessageId": "4f464cdd-15ef-4b57-bd72-3ad225d80adc",
  "ResponseMetadata": {
    "RequestId": "95d39377-8323-5ef0-9223-ceb198bd09bd",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "x-amzn-requestid": "95d39377-8323-5ef0-9223-ceb198bd09bd",
      "date": "Wed, 08 Nov 2023 23:27:26 GMT",
      "content-type": "application/x-amz-json-1.0",
      "content-length": "106",
      "connection": "keep-alive"
    },
    "RetryAttempts": 0
  }
}%
```

Step Functions examples

Basic examples

Invoke a state machine by providing its physical ID as a resource ID:

First, we use `sam list resources` to obtain our physical ID:

```
$ sam list resources --stack-name state-machine-example --output json
```

```
[
  {
    "LogicalResourceId": "HelloWorldStateMachine",
    "PhysicalResourceId": "arn:aws:states:us-
west-2:513423067560:stateMachine:HelloWorldStateMachine-z69tFEUx0F66"
  },
  {
    "LogicalResourceId": "HelloWorldStateMachineRole",
    "PhysicalResourceId": "simple-state-machine-HelloWorldStateMachineRole-
PduA0BDGuFXw"
  }
]
```

Next, we invoke our state machine using the physical ID as a resource ID. We pass in an event at the command line with the `--event` option:

```
$ sam remote invoke arn:aws:states:us-
west-2:01234567890:stateMachine:HelloWorldStateMachine-z69tFEUx0F66 --
event '{"is_developer": true}'

Invoking Step Function arn:aws:states:us-
west-2:01234567890:stateMachine:HelloWorldStateMachine-z69tFEUx0F66
"Hello Developer World"%
```

Invoke a state machine by passing an empty event:

```
$ sam remote invoke HelloWorldStateMachine --stack-name state-machine-example

Invoking Step Function HelloWorldStateMachine

"Hello World"%
```

Related links

For documentation related to `sam remote invoke` and using the AWS SAM CLI, see the following:

- [sam remote invoke](#)
- [AWS SAM CLI troubleshooting](#)

Using sam remote test-event

Use the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam remote test-event` command to access and manage shareable test events for your AWS Lambda functions.

To learn more about shareable test events, see [Shareable test events](#) in the *AWS Lambda Developer Guide*.

Topics

- [Set up the AWS SAM CLI to use sam remote test-event](#)
- [Using the sam remote test-event command](#)
- [Using shareable test events](#)
- [Managing shareable test events](#)

Prerequisites

To use `sam remote test-event`, install the AWS SAM CLI by completing the following:

- [AWS SAM prerequisites](#).
- [Installing the AWS SAM CLI](#).

If you already have the AWS SAM CLI installed, we recommend upgrading to the latest version of the AWS SAM CLI version. To learn more, see [Upgrading the AWS SAM CLI](#).

Before using `sam remote test-event`, we recommend a basic understanding of the following:

- [Configuring the AWS SAM CLI](#).
- [Using sam init](#).
- [Using sam build](#).
- [Using sam deploy](#).
- [Using sam sync](#).

Set up the AWS SAM CLI to use sam remote test-event

Complete the following set up steps to use the AWS SAM CLI `sam remote test-event` command:

1. **Configure the AWS SAM CLI to use your AWS account** – Shareable test events for Lambda can be accessed and managed by users within the same AWS account. To configure the AWS SAM CLI to use your AWS account, see Configuring the AWS SAM CLI.
2. **Configure permissions for shareable test events** – To access and manage shareable test events, you must have the proper permissions. To learn more, see Shareable test events in the *AWS Lambda Developer Guide*.

Using the sam remote test-event command

The AWS SAM CLI `sam remote test-event` command provides the following subcommands that you can use to access and manage your shareable test events:

- `delete` – Delete a shareable test event from the Amazon EventBridge schema registry.
- `get` – Get a shareable test event from the EventBridge schema registry.
- `list` – List the existing shareable test events for a function from the EventBridge schema registry.
- `put` – Save an event from a local file to the EventBridge schema registry.

To list these subcommands using the AWS SAM CLI, run the following:

```
$ sam remote test-event --help
```

Deleting shareable test events

You can delete a shareable test event by using the `delete` subcommand along with the following:

- Provide the name of the shareable test event to delete.
- Provide an acceptable ID of the Lambda function associated with the event.
- If you are providing the Lambda function logical ID, you must also provide the AWS CloudFormation stack name associated with the Lambda function.

The following is an example:

```
$ sam remote test-event delete HelloWorldFunction --stack-name sam-app --name demo-
event
```

For a list of options to use with the `delete` subcommand, see [sam remote test-event delete](). You can also run the following from the AWS SAM CLI:

```
$ sam remote test-event delete --help
```

Getting shareable test events

You can get a shareable test event from the EventBridge schema registry by using the `get` subcommand along with the following:

- Provide the name of the shareable test event to get.

- Provide an acceptable ID of the Lambda function associated with the event.

- If you are providing the Lambda function logical ID, you must also provide the AWS CloudFormation stack name associated with the Lambda function.

The following is an example that gets a shareable test event named `demo-event` that is associated with the `HelloWorldFunction` Lambda function of the `sam-app` stack. This command will print the event to your console.

```
$ sam remote test-event get HelloWorldFunction --stack-name sam-app --name demo-event
```

To get a shareable test event and save it to your local machine, use the `--output-file` option and provide a file path and name. The following is an example that saves demo-event as demo-event.json in the current working directory:

```
$ sam remote test-event get HelloWorldFunction --stack-name sam-app --name demo-event
  --output-file demo-event.json
```

For a list of options to use with the `get` subcommand, see [sam remote test-event get](). You can also run the following from the AWS SAM CLI:

```
$ sam remote test-event get --help
```

Listing shareable test events

You can list all shareable test events for a particular Lambda function from the schema registry. Use the `list` subcommand along with the following:

- Provide an acceptable ID of the Lambda function associated with the events.

- If you are providing the Lambda function logical ID, you must also provide the AWS CloudFormation stack name associated with the Lambda function.

The following is an example that obtains a list of all shareable test events associated with the `HelloWorldFunction` Lambda function of the `sam-app` stack:

```
$ sam remote test-event list HelloWorldFunction --stack-name sam-app
```

For a list of options to use with the `list` subcommand, see [sam remote test-event list](). You can also run the following from the AWS SAM CLI:

```
$ sam remote test-event list --help
```

Saving shareable test events

You can save shareable test events to the EventBridge schema registry. Use the `put` subcommand along with the following:

- Provide an acceptable ID of the Lambda function associated with the shareable test event.

- Provide a name for the shareable test event.

- Provide the file path and name to the local event to upload.

The following is an example that saves the local `demo-event.json` event as `demo-event` and associates it with the `HelloWorldFunction` Lambda function of the `sam-app` stack:

```
$ sam remote test-event put HelloWorldFunction --stack-name sam-app --name demo-event
  --file demo-event.json
```

If a shareable test event with the same name exists in the EventBridge schema registry, the AWS SAM CLI will not overwrite it. To overwrite, add the `--force` option to your command.

For a list of options to use with the `put` subcommand, see [sam remote test-event put](#). You can also run the following from the AWS SAM CLI:

```
$ sam remote test-event put --help
```

Using shareable test events

Use shareable test events to test your Lambda functions in the AWS Cloud with the `sam remote invoke` command. To learn more, see [Pass shareable test events to a Lambda function in the cloud](#).

Managing shareable test events

This topic contains examples on how you can manage and use shareable test events.

Get a shareable test event, modify it, and use it

You can get a shareable test event from the EventBridge schema registry, modify it locally, and use the local test event with your Lambda function in the AWS Cloud. The following is an example:

1. **Retrieve the shareable test event** – Use the `sam remote test-event get` subcommand to retrieve a shareable test event for a specific Lambda function and save it locally:

   ```
   $ sam remote test-event get HelloWorldFunction --stack-name sam-app --name demo-event
     --output-file demo-event.json
   ```

2. **Modify the shareable test event** – Use a text editor of your choice to modify the shareable test event.

3. **Use the shareable test event** – Use the `sam remote invoke` command and provide the file path and name of the event with `--event-file`:

   ```
   $ sam remote invoke HelloWorldFunction --stack-name sam-app --event-file demo-event.json
   ```

Get a shareable test event, modify it, upload it, and use it

You can get a shareable test event from the EventBridge schema registry, modify it locally, and upload it. Then, you can pass the shareable test event directly to your Lambda function in the AWS Cloud. The following is an example:

1. **Retrieve the shareable test event** – Use the `sam remote test-event get` subcommand to retrieve a shareable test event for a specific Lambda function and save it locally:

   ```
   $ sam remote test-event get HelloWorldFunction --stack-name sam-app --name demo-event
    --output-file demo-event.json
   ```

2. **Modify the shareable test event** – Use a text editor of your choice to modify the shareable test event.

3. **Upload the shareable test event** – Use the `sam remote test-event put` subcommand to upload and save the shareable test event to the EventBridge schema registry. In this example, we use the `--force` option to overwrite an older version of our shareable test:

   ```
   $ sam remote test-event put HelloWorldFunction --stack-name sam-app --name demo-event
    --file demo-event.json --force
   ```

4. **Pass the shareable test event to your Lambda function** – Use the `sam remote invoke` command to pass the shareable test event directly to your Lambda function in the AWS Cloud:

   ```
   $ sam remote invoke HelloWorldFunction --stack-name sam-app --test-event-name demo-
   event
   ```

Using sam sync

The AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam sync` command provides options to quickly sync local application changes to the AWS Cloud. Use `sam sync` when developing your applications to:

1. Automatically detect and sync local changes to the AWS Cloud.

2. Customize what local changes are synced to the AWS Cloud.

3. Prepare your application in the cloud for testing and validation.

With `sam sync`, you can create a rapid development workflow that shortens the time it takes to sync your local changes to the cloud for testing and validation.

> **ⓘ Note**
>
> The `sam sync` command is recommended for development environments. For production environments, we recommend using `sam deploy` or configuring a *continuous integration and delivery (CI/CD)* pipeline. To learn more, see [Deploying serverless applications](#).

The `sam sync` command is a part of AWS SAM Accelerate. *AWS SAM Accelerate* provides tools that you can use to speed up the experience of developing and testing serverless applications in the AWS Cloud.

Topics

- [Automatically detect and sync local changes to the AWS Cloud](#)
- [Customize what local changes are synced to the AWS Cloud](#)
- [Prepare your application in the cloud for testing and validation](#)
- [Options for the sam sync command](#)
- [Troubleshooting](#)
- [Examples](#)
- [Learn more](#)

Automatically detect and sync local changes to the AWS Cloud

Run `sam sync` with the `--watch` option to begin syncing your application to the AWS Cloud. This does the following:

1. **Build your application** – This process is similar to using the `sam build` command.

2. **Deploy your application** – The AWS SAM CLI deploys your application to AWS CloudFormation using your default settings. The following default values are used:

   a. AWS credentials and general configuration settings found in your `.aws` user folder.

   b. Application deployment settings found in your application's `samconfig.toml` file.

   If default values can't be found, the AWS SAM CLI will inform you and exit the sync process.

3. **Watch for local changes** – The AWS SAM CLI remains running and watches for local changes to your application. This is what the `--watch` option provides.

This option may be turned on by default. For default values, see your application's `samconfig.toml` file. The following is an example file:

```
...
[default.sync]
[default.sync.parameters]
watch = true
...
```

4. **Sync local changes to the AWS Cloud** – When you make local changes, the AWS SAM CLI detects and syncs those changes to the AWS Cloud through the quickest method available. Depending on the type of change, the following may occur:

   a. If your updated resource supports AWS service APIs, the AWS SAM CLI will use it to deploy your changes. This results in a quick sync to update your resource in the AWS Cloud.

   b. If your updated resource doesn't support AWS service APIs, the AWS SAM CLI will perform an AWS CloudFormation deployment. This updates your entire application in the AWS Cloud. While not as quick, it does prevent you from having to manually initiate a deployment.

Since the `sam sync` command automatically updates your application in the AWS Cloud, it is recommended for development environments only. When you run `sam sync`, you will be asked to confirm:

```
**The sync command should only be used against a development stack**.

Confirm that you are synchronizing a development stack.

Enter Y to proceed with the command, or enter N to cancel:
 [Y/n]: ENTER
```

Customize what local changes are synced to the AWS Cloud

Provide options to customize what local changes are synced to the AWS Cloud. This can speed up the time it takes to see your local changes in the cloud for testing and validation.

For example, provide the `--code` option to sync only code changes, such as AWS Lambda function code. During development, if you are focused specifically on Lambda code, this will get your changes into the cloud quickly for testing and validation. The following is an example:

```
$ sam sync --code --watch
```

To sync only code changes for a specific Lambda function or layer, use the `--resource-id` option. The following is an example:

```
$ sam sync --code --resource-id HelloWorldFunction --resource-id HelloWorldLayer
```

Prepare your application in the cloud for testing and validation

The `sam sync` command automatically finds the quickest method available to update your application in the AWS Cloud. This can speed up your development and cloud testing workflows. By utilizing AWS service APIs, you can quickly develop, sync, and test supported resources. For a hands-on example, see [Module 6 - AWS SAM Accelerate](#) in *The Complete AWS SAM Workshop*.

Options for the sam sync command

The following are some of the main options you can use to modify the `sam sync` command. For a list of all options, see [sam sync](#).

Perform a one-time AWS CloudFormation deployment

Use the `--no-watch` option to turn off automatic syncing. The following is an example:

```
$ sam sync --no-watch
```

The AWS SAM CLI will perform a one-time AWS CloudFormation deployment. This command groups together the actions performed by the `sam build` and `sam deploy` commands.

Skip the initial AWS CloudFormation deployment

You can customize whether an AWS CloudFormation deployment is required each time `sam sync` is run.

- Provide `--no-skip-deploy-sync` to require an AWS CloudFormation deployment each time `sam sync` is run. This ensures that your local infrastructure is synced to AWS CloudFormation, preventing drift. Using this option does add additional time to your development and testing workflow.

- Provide `--skip-deploy-sync` to make AWS CloudFormation deployment optional. The AWS SAM CLI will compare your local AWS SAM template with your deployed AWS CloudFormation

template and will skip the initial AWS CloudFormation deployment if a change isn't detected. Skipping AWS CloudFormation deployment can save you time when syncing local changes to the AWS Cloud.

If no change is detected, the AWS SAM CLI will still perform an AWS CloudFormation deployment in the following scenarios:

- If its been 7 days or more since your last AWS CloudFormation deployment.

- If a large number of Lambda function code changes are detected, making AWS CloudFormation deployment the quickest method to update your application.

The following is an example:

```
$ sam sync --skip-deploy-sync
```

Sync a resource from a nested stack

To sync a resource from a nested stack

1. Provide the root stack using `--stack-name`.

2. Identify the resource in the nested stack using the following format: *nestedStackId/resourceId*.

3. Provide the resource in the nested stack using `--resource-id`.

   The following is an example:

   ```
   $ sam sync --code --stack-name sam-app --resource-id myNestedStack/
   HelloWorldFunction
   ```

For more information about creating nested applications, see [Using nested applications](#).

Specify a specific AWS CloudFormation stack to update

To specify a specific AWS CloudFormation stack to update, provide the `--stack-name` option. The following is an example:

```
$ sam sync --stack-name dev-sam-app
```

Speed up build times by building your project in the source folder

For supported runtimes and build methods, you can use the `--build-in-source` option to build your project directly in the source folder. By default, the AWS SAM CLI builds in a temporary directory, which involves copying over source code and project files. With `--build-in-source`, the AWS SAM CLI builds directly in your source folder, which speeds up the build process by removing the need to copy files to a temporary directory.

For a list of supported runtimes and build methods, see `--build-in-source`.

Specify files and folders that won't initiate a sync

Use the `--watch-exclude` option to specify any file or folder that won't initiate a sync when updated. For more information about this option, see `--watch-exclude`.

The following is an example that excludes the `package-lock.json` file associated with our `HelloWorldFunction` function:

```
$ sam sync --watch --watch-exclude HelloWorldFunction=package-lock.json
```

When this command is run, the AWS SAM CLI will initiate the sync process. This includes the following:

- Run `sam build` to build your functions and prepare your application for deployment.
- Run `sam deploy` to deploy your application.
- Watch for changes to your application.

When we modify the `package-lock.json` file, the AWS SAM CLI won't initiate a sync. When another file is updated, the AWS SAM CLI will initiate a sync, which will include the `package-lock.json` file.

The following is an example of specifying a Lambda function of a child stack:

```
$ sam sync --watch --watch-exclude ChildStackA/MyFunction=database.sqlite3
```

Troubleshooting

To troubleshoot the AWS SAM CLI, see AWS SAM CLI troubleshooting.

Examples

Using sam sync to update the Hello World application

In this example, we start by initializing the sample Hello World application. To learn more about this application, see [Tutorial: Deploying a Hello World application](#).

Running `sam sync` begins the build and deployment process.

```
$ sam sync

The SAM CLI will use the AWS Lambda, Amazon API Gateway, and AWS StepFunctions APIs to
 upload your code without
performing a CloudFormation deployment. This will cause drift in your CloudFormation
 stack.
**The sync command should only be used against a development stack**.

Confirm that you are synchronizing a development stack.

Enter Y to proceed with the command, or enter N to cancel:
 [Y/n]:
Queued infra sync. Waiting for in progress code syncs to complete...
Starting infra sync.
Manifest file is changed (new hash: 3298f13049d19cffaa37ca931dd4d421) or dependency
 folder (.aws-sam/deps/0663e6fe-a888-4efb-b908-e2344261e9c7) is missing for
 (HelloWorldFunction), downloading dependencies and copying/building source
Building codeuri: /Users/.../Demo/sync/sam-app/hello_world runtime: python3.9 metadata:
 {} architecture: x86_64 functions: HelloWorldFunction
Running PythonPipBuilder:CleanUp
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource

Build Succeeded

Successfully packaged artifacts and wrote output template to file /var/
folders/45/5ct135bx3fn2551_ptl5g6_80000gr/T/tmpx_5t4u3f.
Execute the following command to deploy the packaged template
sam deploy --template-file /var/folders/45/5ct135bx3fn2551_ptl5g6_80000gr/T/tmpx_5t4u3f
 --stack-name <YOUR STACK NAME>


    Deploying with following values
    ===============================
    Stack name                   : sam-app
```

```
    Region                      : us-west-2
    Disable rollback            : False
    Deployment s3 bucket        : aws-sam-cli-managed-default-
samclisourcebucket-1a4x26zbcdkqr
    Capabilities                : ["CAPABILITY_NAMED_IAM", "CAPABILITY_AUTO_EXPAND"]
    Parameter overrides         : {}
    Signing Profiles            : null


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


2023-03-17 11:17:19 - Waiting for stack create/update to complete

CloudFormation events from stack operations (refresh every 0.5 seconds)
-----------------------------------------------------------------------------------------
ResourceStatus                     ResourceType
 LogicalResourceId                   ResourceStatusReason
-----------------------------------------------------------------------------------------
CREATE_IN_PROGRESS                 AWS::CloudFormation::Stack          sam-app
                    Transformation succeeded
CREATE_IN_PROGRESS                 AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt   -
                                                                            ack
CREATE_IN_PROGRESS                 AWS::IAM::Role
 HelloWorldFunctionRole              -
CREATE_IN_PROGRESS                 AWS::IAM::Role
 HelloWorldFunctionRole              Resource creation Initiated
CREATE_IN_PROGRESS                 AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt   Resource creation Initiated
                                                                            ack
CREATE_COMPLETE                    AWS::IAM::Role
 HelloWorldFunctionRole              -
CREATE_COMPLETE                    AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt   -
                                                                            ack
CREATE_IN_PROGRESS                 AWS::Lambda::Function
 HelloWorldFunction                  -
CREATE_IN_PROGRESS                 AWS::Lambda::Function
 HelloWorldFunction                  Resource creation Initiated
CREATE_COMPLETE                    AWS::Lambda::Function
 HelloWorldFunction                  -
CREATE_IN_PROGRESS                 AWS::ApiGateway::RestApi
 ServerlessRestApi                   -
```

```
CREATE_IN_PROGRESS                      AWS::ApiGateway::RestApi
 ServerlessRestApi                        Resource creation Initiated
CREATE_COMPLETE                         AWS::ApiGateway::RestApi
 ServerlessRestApi                        -
CREATE_IN_PROGRESS                      AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment47fc2d        -
                                                                      5f9d
CREATE_IN_PROGRESS                      AWS::Lambda::Permission
 HelloWorldFunctionHelloWorldPermi        -
                                                                      ssionProd
CREATE_IN_PROGRESS                      AWS::Lambda::Permission
 HelloWorldFunctionHelloWorldPermi        Resource creation Initiated
                                                                      ssionProd
CREATE_IN_PROGRESS                      AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment47fc2d        Resource creation Initiated
                                                                      5f9d
CREATE_COMPLETE                         AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment47fc2d        -
                                                                      5f9d
CREATE_IN_PROGRESS                      AWS::ApiGateway::Stage
 ServerlessRestApiProdStage               -
CREATE_IN_PROGRESS                      AWS::ApiGateway::Stage
 ServerlessRestApiProdStage               Resource creation Initiated
CREATE_COMPLETE                         AWS::ApiGateway::Stage
 ServerlessRestApiProdStage               -
CREATE_COMPLETE                         AWS::Lambda::Permission
 HelloWorldFunctionHelloWorldPermi        -
                                                                      ssionProd
CREATE_COMPLETE                         AWS::CloudFormation::Stack    sam-app
                       -
-------------------------------------------------------------------------------


CloudFormation outputs from deployed stack
-------------------------------------------------------------------------------
Outputs
-------------------------------------------------------------------------------
Key                 HelloWorldFunctionIamRole
Description          Implicit IAM Role created for Hello World function
Value               arn:aws:iam::012345678910:role/sam-app-HelloWorldFunctionRole-
BUFVMO2PJIYF

Key                 HelloWorldApi
Description          API Gateway endpoint URL for Prod stage for Hello World function
Value               https://pcrx5gdaof.execute-api.us-west-2.amazonaws.com/Prod/hello/
```

```
Key                    HelloWorldFunction
Description            Hello World Lambda Function ARN
Value                  arn:aws:lambda:us-west-2:012345678910:function:sam-app-
HelloWorldFunction-2PlN6TPTQoco
------------------------------------------------------------------------------------
Stack creation succeeded. Sync infra completed.


Infra sync completed.
CodeTrigger not created as CodeUri or DefinitionUri is missing for ServerlessRestApi.
```

Once deployment is complete, we modify the `HelloWorldFunction` code. The AWS SAM CLI detects this change and syncs our application to the AWS Cloud. Since AWS Lambda supports AWS service APIs, a quick sync is performed.

```
Syncing Lambda Function HelloWorldFunction...
Manifest is not changed for (HelloWorldFunction), running incremental build
Building codeuri: /Users/.../Demo/sync/sam-app/hello_world runtime: python3.9 metadata:
 {} architecture: x86_64 functions: HelloWorldFunction
Running PythonPipBuilder:CopySource
Finished syncing Lambda Function HelloWorldFunction.
```

Next, we modify our API endpoint in the application's AWS SAM template. We change `/hello` to `/helloworld`.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  HelloWorldFunction:
    ...
    Properties:
      ...
      Events:
        HelloWorld:
          Type: Api
          Properties:
            Path: /helloworld
            Method: get
```

Since the Amazon API Gateway resource doesn't support the AWS service API, the AWS SAM CLI automatically performs an AWS CloudFormation deployment. The following is an example output:

```
Queued infra sync. Waiting for in progress code syncs to complete...
Starting infra sync.
Manifest is not changed for (HelloWorldFunction), running incremental build
Building codeuri: /Users/.../Demo/sync/sam-app/hello_world runtime: python3.9 metadata:
 {} architecture: x86_64 functions: HelloWorldFunction
Running PythonPipBuilder:CopySource


Build Succeeded


Successfully packaged artifacts and wrote output template to file /var/
folders/45/5ct135bx3fn2551_ptl5g6_80000gr/T/tmpuabo0jb9.
Execute the following command to deploy the packaged template
sam deploy --template-file /var/folders/45/5ct135bx3fn2551_ptl5g6_80000gr/T/tmpuabo0jb9
 --stack-name <YOUR STACK NAME>



    Deploying with following values
    ==============================
    Stack name                    : sam-app
    Region                        : us-west-2
    Disable rollback              : False
    Deployment s3 bucket          : aws-sam-cli-managed-default-
samclisourcebucket-1a4x26zbcdkqr
    Capabilities                  : ["CAPABILITY_NAMED_IAM", "CAPABILITY_AUTO_EXPAND"]
    Parameter overrides           : {}
    Signing Profiles              : null


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



2023-03-17 14:41:18 - Waiting for stack create/update to complete

CloudFormation events from stack operations (refresh every 0.5 seconds)
-------------------------------------------------------------------------------------
ResourceStatus                         ResourceType
 LogicalResourceId                       ResourceStatusReason
-------------------------------------------------------------------------------------
UPDATE_IN_PROGRESS                     AWS::CloudFormation::Stack          sam-app
                   Transformation succeeded
UPDATE_IN_PROGRESS                     AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt    -
                                                          ack
```

```
UPDATE_COMPLETE                        AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt     -
                                                                              ack
UPDATE_IN_PROGRESS                     AWS::ApiGateway::RestApi
 ServerlessRestApi                     -
UPDATE_COMPLETE                        AWS::ApiGateway::RestApi
 ServerlessRestApi                     -
CREATE_IN_PROGRESS                     AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment8cf30e     -
                                                                              d3cd
UPDATE_IN_PROGRESS                     AWS::Lambda::Permission
 HelloWorldFunctionHelloWorldPermi       Requested update requires the
                                                                              ssionProd
                       creation of a new physical

                       resource; hence creating one.
UPDATE_IN_PROGRESS                     AWS::Lambda::Permission
 HelloWorldFunctionHelloWorldPermi       Resource creation Initiated
                                                                              ssionProd
CREATE_IN_PROGRESS                     AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment8cf30e       Resource creation Initiated
                                                                              d3cd
CREATE_COMPLETE                        AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment8cf30e     -
                                                                              d3cd
UPDATE_IN_PROGRESS                     AWS::ApiGateway::Stage
 ServerlessRestApiProdStage            -
UPDATE_COMPLETE                        AWS::ApiGateway::Stage
 ServerlessRestApiProdStage            -
UPDATE_COMPLETE                        AWS::Lambda::Permission
 HelloWorldFunctionHelloWorldPermi     -
                                                                              ssionProd
UPDATE_COMPLETE_CLEANUP_IN_PROGRE      AWS::CloudFormation::Stack    sam-app
                       -
SS
DELETE_IN_PROGRESS                     AWS::Lambda::Permission
 HelloWorldFunctionHelloWorldPermi     -
                                                                              ssionProd
DELETE_IN_PROGRESS                     AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment47fc2d     -
                                                                              5f9d
DELETE_COMPLETE                        AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment47fc2d     -
                                                                              5f9d
```

```
UPDATE_COMPLETE                    AWS::CloudFormation::Stack
  AwsSamAutoDependencyLayerNestedSt    -
                                                                      ack
DELETE_COMPLETE                    AWS::Lambda::Permission
  HelloWorldFunctionHelloWorldPermi    -
                                                                      ssionProd
UPDATE_COMPLETE                    AWS::CloudFormation::Stack    sam-app
                          -
-------------------------------------------------------------------------------

CloudFormation outputs from deployed stack
-------------------------------------------------------------------------------
Outputs
-------------------------------------------------------------------------------
Key                 HelloWorldFunctionIamRole
Description         Implicit IAM Role created for Hello World function
Value               arn:aws:iam::012345678910:role/sam-app-HelloWorldFunctionRole-
BUFVMO2PJIYF

Key                 HelloWorldApi
Description         API Gateway endpoint URL for Prod stage for Hello World function
Value               https://pcrx5gdaof.execute-api.us-west-2.amazonaws.com/Prod/hello/

Key                 HelloWorldFunction
Description         Hello World Lambda Function ARN
Value               arn:aws:lambda:us-west-2:012345678910:function:sam-app-
HelloWorldFunction-2PlN6TPTQoco
-------------------------------------------------------------------------------


Stack update succeeded. Sync infra completed.

Infra sync completed.
```

Learn more

For a description of all sam sync options, see [sam sync](#).

AWS Serverless Application Model (AWS SAM) specification

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

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

Topics

- AWS SAM template anatomy
- AWS SAM resource and property reference
- Resource attributes
- Intrinsic functions
- Generated AWS CloudFormation resources
- API Gateway extensions

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](#).

- **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](#).

- **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](#) 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.

```
Transform: AWS::Serverless-2016-10-31

Globals:
  set of globals

Description:
  String

Metadata:
  template metadata

Parameters:
  set of parameters

Mappings:
  set of mappings

Conditions:
  set of conditions

Resources:
```

```
    set of resources


Outputs:
    set of outputs
```

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, if using language extensions, you should add `AWS::LanguageExtensions` *before* the serverless transform (that is, before `AWS::Serverless-2016-10-31`) as shown in the following example:

```
Transform:
  - AWS::LanguageExtensions
  - AWS::Serverless-2016-10-31
```

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)

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.

This section corresponds directly with the `Metadata` section of AWS CloudFormation templates.

Parameters (optional)

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

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

Mappings (optional)

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

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

Conditions (optional)

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

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

Resources (required)

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

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

Outputs (optional)

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

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

Next steps

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

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 supports the following AWS SAM resource types:

- `AWS::Serverless::Api`
- `AWS::Serverless::Function`
- `AWS::Serverless::HttpApi`
- `AWS::Serverless::SimpleTable`
- `AWS::Serverless::StateMachine`

Example:

```
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.

```
 Globals:
   Api:
     AccessLogSetting:
     Auth:
     BinaryMediaTypes:
     CacheClusterEnabled:
     CacheClusterSize:
     CanarySetting:
     Cors:
     DefinitionUri:
```

```
      Domain:
      EndpointConfiguration:
      GatewayResponses:
      MethodSettings:
      MinimumCompressionSize:
      Name:
      OpenApiVersion:
      PropagateTags:
      TracingEnabled:
      Variables:

  Function:
      Architectures:
      AssumeRolePolicyDocument:
      AutoPublishAlias:
      CodeUri:
      DeadLetterQueue:
      DeploymentPreference:
      Description:
      Environment:
      EphemeralStorage:
      EventInvokeConfig:
      Handler:
      KmsKeyArn:
      Layers:
      MemorySize:
      PermissionsBoundary:
      PropagateTags:
      ProvisionedConcurrencyConfig:
      ReservedConcurrentExecutions:
      Runtime:
      Tags:
      Timeout:
      Tracing:
      VpcConfig:

  HttpApi:
      AccessLogSettings:
      Auth:
      PropagateTags:
      StageVariables:
      Tags:

  SimpleTable:
```

```
      SSESpecification:

  StateMachine:
    PropagateTags:
```

> **ⓘ 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:

```
Globals:
  Function:
```

```
    Runtime: nodejs12.x

Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      Runtime: python3.9
```

The `Runtime` for `MyFunction` is set to `python3.9`.

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:

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

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

The environment variables of `MyFunction` are set to the following:

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

Lists are additive

Lists are also known as arrays.

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

Example:

```
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
- AWS::Serverless::Application
- AWS::Serverless::Connector
- AWS::Serverless::Function
- AWS::Serverless::GraphQLApi
- AWS::Serverless::HttpApi
- AWS::Serverless::LayerVersion

- [AWS::Serverless::SimpleTable](#)
- [AWS::Serverless::StateMachine](#)

AWS::Serverless::Api

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

An [AWS::Serverless::Api](#) 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](#) resources defined in the template that do not refer to an [AWS::Serverless::Api](#) resource.

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

We recommend that you use AWS CloudFormation hooks or IAM policies to verify that API Gateway resources have authorizers attached to them to control access to them.

For more information about using AWS CloudFormation hooks, see [Registering hooks](#) in the *AWS CloudFormation CLI user guide* and the [apigw-enforce-authorizer](#) GitHub repository.

For more information about using IAM policies, see [Require that API routes have authorization](#) in the *API Gateway Developer Guide*.

> **ⓘ Note**
>
> When you deploy to AWS CloudFormation, AWS SAM transforms your AWS SAM resources into AWS CloudFormation resources. For more information, see [Generated AWS CloudFormation 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
  AlwaysDeploy: Boolean
  ApiKeySourceType: String
  Auth: ApiAuth
  BinaryMediaTypes: List
  CacheClusterEnabled: Boolean
  CacheClusterSize: String
  CanarySetting: CanarySetting
  Cors: String | CorsConfiguration
  DefinitionBody: JSON
  DefinitionUri: String | ApiDefinition
  Description: String
  DisableExecuteApiEndpoint: Boolean
  Domain: DomainConfiguration
  EndpointConfiguration: EndpointConfiguration
  FailOnWarnings: Boolean
  GatewayResponses: Map
  MergeDefinitions: Boolean
  MethodSettings: MethodSettings
  MinimumCompressionSize: Integer
  Mode: String
  Models: Map
  Name: String
  OpenApiVersion: String
  PropagateTags: Boolean
  StageName: String
  Tags: Map
  TracingEnabled: Boolean
  Variables: Map
```

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.

AlwaysDeploy

Always deploys the API, even when no changes to the API have been detected.

Type: Boolean

Required: No

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

ApiKeySourceType

The source of the API key for metering requests according to a usage plan. Valid values are HEADER and AUTHORIZER.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [ApiKeySourceType](#) property of an AWS::ApiGateway::RestApi 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](#).

Type: [ApiAuth](#)

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 caching is enabled for the stage. To cache responses, you must also set CachingEnabled to `true` under MethodSettings.

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. Create an inline OpenAPI definition in the `DefinitionBody` to turn on CORS.

For more information about CORS, see [Enable CORS for an API Gateway REST API Resource](#) in the *API Gateway Developer Guide*.

Type: String | [CorsConfiguration](#)

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.

To reference a local OpenAPI file that defines your API, use the `AWS::Include` transform. To learn more, see [Upload local files at deployment](#).

Type: JSON

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 for a corresponding `AWS::Serverless::Function`.

DefinitionUri

Amazon S3 Uri, local file path, or location object of the the OpenAPI document defining the API. The Amazon 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](#)

Required: No

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

Description

A description of the Api resource.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [Description](#) property of an `AWS::ApiGateway::RestApi` resource.

DisableExecuteApiEndpoint

Specifies whether clients can invoke your API by using the default `execute-api` endpoint. By default, clients can invoke your API with the default `https://{api_id}.execute-api.{region}.amazonaws.com`. To require that clients use a custom domain name to invoke your API, specify `True`.

Type: Boolean

Required: No

AWS CloudFormation compatibility: This property is similar to the [DisableExecuteApiEndpoint](#) property of an `AWS::ApiGateway::RestApi` resource. It is passed directly to the `disableExecuteApiEndpoint` property of an [x-amazon-](#)

`apigateway-endpoint-configuration` extension, which gets added to the `Body` property of an `AWS::ApiGateway::RestApi` resource.

Domain

Configures a custom domain for this API Gateway API.

Type: DomainConfiguration

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

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.

FailOnWarnings

Specifies whether to roll back 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::ApiGateway::RestApi` resource.

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.

MergeDefinitions

AWS SAM generates an OpenAPI specification from your API event source. Specify `true` to have AWS SAM merge this into the inline OpenAPI specification defined in your `AWS::Serverless::Api` resource. Specify `false` to not merge.

`MergeDefinitions` requires the `DefinitionBody` property for `AWS::Serverless::Api` to be defined. `MergeDefinitions` is not compatible with the `DefinitionUri` property for `AWS::Serverless::Api`.

Default value: `false`

Type: Boolean

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: List of [MethodSetting](#)

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.

Mode

This property applies only when you use OpenAPI to define your REST API. The Mode
determines how API Gateway handles resource updates. For more information, see [Mode](#)
property of the [AWS::ApiGateway::RestApi](#) resource type.

Valid values: `overwrite` or `merge`

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [Mode](#) 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**
>
> AWS SAM creates a stage called `Stage` by default. Setting this property to any valid value will prevent the creation of the stage `Stage`.

Type: String

Required: No

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

`PropagateTags`

Indicate whether or not to pass tags from the `Tags` property to your [AWS::Serverless::Api](#) generated resources. Specify `True` to propagate tags in your generated resources.

Type: Boolean

Required: No

Default: `False`

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

`StageName`

The name of the stage, which API Gateway uses as the first path segment in the invoke Uniform Resource Identifier (URI).

To reference the stage resource, use `<api-logical-id>`.`Stage`. For more information about referencing resources generated when an [AWS::Serverless::Api](#) resource is specified, see [AWS CloudFormation resources generated when AWS::Serverless::Api is specified](#). For

general information about generated AWS CloudFormation resources, see Generated AWS
CloudFormation resources.

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. For details
about valid keys and values for tags, see Resource tag in the *AWS CloudFormation User Guide*.

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

```
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 method at the root resource via an Api event
    Type: AWS::Serverless::Function
    Properties:
      Events:
        ApiEvent:
          Type: Api
          Properties:
            Path: /
            Method: get
            RestApiId:
              Ref: ApiGatewayApi
      Runtime: python3.10
      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

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

ApiCognitoAuthExample

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

YAML

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

ApiModelsExample

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

YAML

```
Resources:
  ApiGatewayApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      Models:
        User:
          type: object
          required:
            - username
            - employee_id
          properties:
            username:
              type: string
            employee_id:
```

```
                type: integer
              department:
                type: string
          Item:
            type: object
            properties:
              count:
                type: integer
              category:
                type: string
              price:
                type: integer
```

Caching example

A Hello World AWS SAM template file that contains a Lambda Function with an API endpoint. The API has caching enabled for one resource and method. For more information about caching, see Enabling API caching to enhance responsiveness in the *API Gateway Developer Guide*.

YAML

```yaml
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: AWS SAM template with a simple API definition with caching turned on
Resources:
  ApiGatewayApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: prod
      CacheClusterEnabled: true
      CacheClusterSize: '0.5'
      MethodSettings:
        - ResourcePath: /
          HttpMethod: GET
          CachingEnabled: true
          CacheTtlInSeconds: 300
      Tags:
        CacheMethods: All

  ApiFunction: # Adds a GET method at the root resource via an Api event
    Type: AWS::Serverless::Function
    Properties:
      Events:
```

```
    ApiEvent:
      Type: Api
      Properties:
        Path: /
        Method: get
        RestApiId:
          Ref: ApiGatewayApi
  Runtime: python3.10
  Handler: index.handler
  InlineCode: |
    def handler(event, context):
        return {'body': 'Hello World!', 'statusCode': 200}
```

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.

Syntax

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

YAML

```
AddApiKeyRequiredToCorsPreflight: Boolean
AddDefaultAuthorizerToCorsPreflight: Boolean
ApiKeyRequired: Boolean
Authorizers: CognitoAuthorizer | LambdaTokenAuthorizer | LambdaRequestAuthorizer
DefaultAuthorizer: String
InvokeRole: String
ResourcePolicy: ResourcePolicyStatement
UsagePlan: ApiUsagePlan
```

Properties

AddApiKeyRequiredToCorsPreflight

If the ApiKeyRequired and Cors properties are set, then setting AddApiKeyRequiredToCorsPreflight will cause the API key to be added to the Options property.

Type: Boolean

Required: No

Default: `True`

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

`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](#).

Type: [CognitoAuthorizer](#) | [LambdaTokenAuthorizer](#) | [LambdaRequestAuthorizer](#)

Required: No

Default: None

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

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

DefaultAuthorizer

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

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

Type: String

Required: No

Default: None

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

InvokeRole

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

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

Valid values: `CALLER_CREDENTIALS, NONE, IAMRoleArn`

Type: String

Required: No

Default: `CALLER_CREDENTIALS`

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

`ResourcePolicy`

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

Type: [ResourcePolicyStatement](#)

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](#). 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](#). For general information about generated AWS CloudFormation resources, see [Generated AWS CloudFormation resources](#).

Type: [ApiUsagePlan](#)

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

```
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

```
CreateUsagePlan: String
```

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

Properties

CreateUsagePlan

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

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

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

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

Valid values: PER_API, SHARED, and NONE

Type: String

Required: Yes

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

Description

A description of the usage plan.

Type: String

Required: No

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

Quota

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

Type: QuotaSettings

Required: No

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](#).

Syntax

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

YAML

```
  AuthorizationScopes: List
  Identity: CognitoAuthorizationIdentity
```

```
    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](#)

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

```
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](#).

Syntax

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

YAML

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

Properties

Header

>   Specify the header name for Authorization in the OpenApi definition.
>
>   *Type*: String
>
>   *Required*: No

Default: Authorization

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

ReauthorizeEvery

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

Type: Integer

Required: No

Default: 300

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

ValidationExpression

Specify a validation expression for validating the incoming Identity

Type: String

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.

Syntax

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

YAML

```yaml
DisableFunctionDefaultPermissions: Boolean
FunctionArn: String
FunctionInvokeRole: String
FunctionPayloadType: String
Identity: LambdaRequestAuthorizationIdentity
```

Properties

DisableFunctionDefaultPermissions

Specify `true` to prevent AWS SAM from automatically creating an
`AWS::Lambda::Permissions` resource to provision permissions between your
`AWS::Serverless::Api` resource and authorizer Lambda function.

Default value: `false`

Type: Boolean

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.

> **ⓘ Note**
>
> AWS SAM will automatically create an `AWS::Lambda::Permissions`
> resource when `FunctionArn` is specified for `AWS::Serverless::Api`. The

> `AWS::Lambda::Permissions` resource provisions permissions between your API and authorizer Lambda function.

Type: String

Required: Yes

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

FunctionInvokeRole

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

Type: String

Required: No

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

FunctionPayloadType

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

Valid values: TOKEN or REQUEST

Type: String

Required: No

Default: TOKEN

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

Identity

This property can be used to specify an `IdentitySource` in an incoming request for an authorizer. This property is only required if the `FunctionPayloadType` property is set to `REQUEST`.

Type: LambdaRequestAuthorizationIdentity

Required: Conditional

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

Examples

LambdaRequestAuth

YAML

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

LambdaRequestAuthorizationIdentity

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

Syntax

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

YAML

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

```
    StageVariables: List
```

Properties

Context

> Converts the given context strings to the mapping expressions of format
> `context.contextString`.
>
> *Type*: List
>
> *Required*: No
>
> *AWS CloudFormation compatibility*: This property is unique to AWS SAM and doesn't have an
> AWS CloudFormation equivalent.

Headers

> Converts the headers to comma-separated string of mapping expressions of format
> `method.request.header.name`.
>
> *Type*: List
>
> *Required*: No
>
> *AWS CloudFormation compatibility*: This property is unique to AWS SAM and doesn't have an
> AWS CloudFormation equivalent.

QueryStrings

> Converts the given query strings to comma-separated string of mapping expressions of format
> `method.request.querystring.queryString`.
>
> *Type*: List
>
> *Required*: No
>
> *AWS CloudFormation compatibility*: This property is unique to AWS SAM and doesn't have an
> AWS CloudFormation equivalent.

ReauthorizeEvery

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

Type: Integer

Required: No

Default: 300

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

StageVariables

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

Type: List

Required: No

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

Examples

LambdaRequestIdentity

YAML

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

LambdaTokenAuthorizer

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

For more information and examples, see [Controlling access to API Gateway APIs](#).

Syntax

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

YAML

```
DisableFunctionDefaultPermissions: Boolean
FunctionArn: String
FunctionInvokeRole: String
FunctionPayloadType: String
Identity: LambdaTokenAuthorizationIdentity
```

Properties

`DisableFunctionDefaultPermissions`

Specify `true` to prevent AWS SAM from automatically creating an `AWS::Lambda::Permissions` resource to provision permissions between your `AWS::Serverless::Api` resource and authorizer Lambda function.

Default value: `false`

Type: Boolean

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.

> ⓘ **Note**
>
> AWS SAM will automatically create an `AWS::Lambda::Permissions` resource when `FunctionArn` is specified for `AWS::Serverless::Api`. The `AWS::Lambda::Permissions` resource provisions permissions between your API and authorizer Lambda function.

Type: String

Required: Yes

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

FunctionInvokeRole

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

Type: String

Required: No

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

FunctionPayloadType

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

Valid values: TOKEN or REQUEST

Type: String

Required: No

Default: TOKEN

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

Identity

This property can be used to specify an IdentitySource in an incoming request for an authorizer. This property is only required if the FunctionPayloadType property is set to REQUEST.

Type: LambdaTokenAuthorizationIdentity

Required: Conditional

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

```
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

LambdaTokenIdentity

YAML

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

ResourcePolicyStatement

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

Syntax

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

YAML

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

Properties

AwsAccountBlacklist

The AWS accounts to block.

Type: List of String

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 of String

Required: No

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

CustomStatements

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

Type: List

Required: No

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

IntrinsicVpcBlacklist

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

Type: List

Required: No

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

`IntrinsicVpcWhitelist`

The list of VPCs to allow, where each VPC is specified as a reference such as a [dynamic reference](#) or the `Ref` [intrinsic function](#).

Type: List

Required: No

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

`IntrinsicVpceBlacklist`

The list of VPC endpoints to block, where each VPC endpoint is specified as a reference such as a [dynamic reference](#) or the `Ref` [intrinsic function](#).

Type: List

Required: No

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

`IntrinsicVpceWhitelist`

The list of VPC endpoints to allow, where each VPC endpoint is specified as a reference such as a [dynamic reference](#) or the `Ref` [intrinsic function](#). For an example use of this property, see the Examples section at the bottom of this page.

Type: List

Required: No

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

`IpRangeBlacklist`

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

Type: List

Required: No

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

`IpRangeWhitelist`

The IP addresses or address ranges to allow.

Type: List

Required: No

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

`SourceVpcBlacklist`

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

Type: List

Required: No

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

`SourceVpcWhitelist`

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

Type: List

Required: No

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

Examples

Resource Policy Example

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

YAML

```
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

```
  Bucket: String
  Key: String
```

```
    Version: String
```

Properties

Bucket

> The name of the Amazon S3 bucket where the OpenAPI file is stored.
>
> *Type*: String
>
> *Required*: Yes
>
> *AWS CloudFormation compatibility*: This property is passed directly to the Bucket property of
> the AWS::ApiGateway::RestApi S3Location data type.

Key

> The Amazon S3 key of the OpenAPI file.
>
> *Type*: String
>
> *Required*: Yes
>
> *AWS CloudFormation compatibility*: This property is passed directly to the Key property of the
> AWS::ApiGateway::RestApi S3Location data type.

Version

> For versioned objects, the version of the OpenAPI file.
>
> *Type*: String
>
> *Required*: No
>
> *AWS CloudFormation compatibility*: This property is passed directly to the Version property of
> the AWS::ApiGateway::RestApi S3Location data type.

Examples

Definition Uri example

API Definition example

YAML

```
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 AWS SAM to modify your OpenAPI definition. Create an inline OpenAPI definition in the `DefinitionBody` to turn on CORS. If the `CorsConfiguration` is set in the OpenAPI definition and also at the property level, AWS SAM merges them. The property level takes precedence over the OpenAPI definition.

For more information about CORS, see Enable CORS for an API Gateway REST API Resource 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

```
    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. This can be a comma-separated list in string format.

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 and a [AWS::Serverless::Function](#). If you use a Lambda proxy integration or a HTTP proxy integration, your backend must return the `Access-Control-Allow-Origin`, `Access-Control-Allow-Methods`, and `Access-Control-Allow-Headers` headers.

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
  ApiFunction: # Adds a GET method at the root resource via an Api event
    Type: AWS::Serverless::Function
    Properties:
      Events:
        ApiEvent:
          Type: Api
          Properties:
            Path: /
            Method: get
            RestApiId:
              Ref: ApiGatewayApi
      Runtime: python3.10
      Handler: index.handler
      InlineCode: |
```

```
      import json
      def handler(event, context):
        return {
        'statusCode': 200,
        'headers': {
          'Access-Control-Allow-Headers': 'Content-Type',
          'Access-Control-Allow-Origin': 'www.example.com',
          'Access-Control-Allow-Methods': 'POST, GET'
          },
        'body': json.dumps('Hello from Lambda!')
        }
```

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

```
BasePath: List
NormalizeBasePath: Boolean
CertificateArn: String
DomainName: String
EndpointConfiguration: String
MutualTlsAuthentication: MutualTlsAuthentication
OwnershipVerificationCertificateArn: String
Route53: Route53Configuration
SecurityPolicy: String
```

Properties

BasePath

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

Type: List

Required: No

Default: /

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

NormalizeBasePath

Indicates whether non-alphanumeric characters are allowed in basepaths defined by the `BasePath` property. When set to `True`, non-alphanumeric characters are removed from basepaths.

Use `NormalizeBasePath` with the `BasePath` property.

Type: Boolean

Required: No

Default: True

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

CertificateArn

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

Type: String

Required: Yes

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

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

```
DomainName
```

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

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

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the [DomainName](#) property of an `AWS::ApiGateway::DomainName` resource.

```
EndpointConfiguration
```

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

Valid values: REGIONAL or EDGE

Type: String

Required: No

Default: REGIONAL

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

```
MutualTlsAuthentication
```

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

Type: [MutualTlsAuthentication](#)

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [MutualTlsAuthentication](#) property of an `AWS::ApiGateway::DomainName` resource.

OwnershipVerificationCertificateArn

The ARN of the public certificate issued by ACM to validate ownership of your custom domain. Required only when you configure mutual TLS and you specify an ACM imported or private CA certificate ARN for the `CertificateArn`.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [OwnershipVerificationCertificateArn](#) property of an `AWS::ApiGateway::DomainName` resource.

Route53

Defines an Amazon Route 53 configuration.

Type: [Route53Configuration](#)

Required: No

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

SecurityPolicy

The TLS version plus cipher suite for this domain name.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [SecurityPolicy](#) property of an `AWS::ApiGateway::DomainName` resource.

Examples

DomainName

DomainName example

YAML

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

```
DistributionDomainName: String
EvaluateTargetHealth: Boolean
HostedZoneId: String
HostedZoneName: String
IpV6: Boolean
Region: String
SetIdentifier: String
```

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.

Region

Latency-based resource record sets only: The Amazon EC2 Region where you created the resource that this resource record set refers to. The resource typically is an AWS resource, such as an EC2 instance or an ELB load balancer, and is referred to by an IP address or a DNS domain name, depending on the record type.

When Amazon Route 53 receives a DNS query for a domain name and type for which you have created latency resource record sets, Route 53 selects the latency resource record set that has the lowest latency between the end user and the associated Amazon EC2 Region. Route 53 then returns the value that is associated with the selected resource record set.

Note the following:

- You can only specify one `ResourceRecord` per latency resource record set.
- You can only create one latency resource record set for each Amazon EC2 Region.
- You aren't required to create latency resource record sets for all Amazon EC2 Regions. Route 53 will choose the region with the best latency from among the regions that you create latency resource record sets for.
- You can't create non-latency resource record sets that have the same values for the `Name` and `Type` elements as latency resource record sets.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [Region](#) property of
an `AWS::Route53::RecordSetGroup RecordSet` data type.

`SetIdentifier`

Resource record sets that have a routing policy other than simple: An identifier that differentiates
among multiple resource record sets that have the same combination of name and type, such
as multiple weighted resource record sets named acme.example.com that have a type of A. In a
group of resource record sets that have the same name and type, the value of `SetIdentifier`
must be unique for each resource record set.

For information about routing policies, see [Choosing a routing policy](#) in the *Amazon Route 53
Developer Guide*.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [SetIdentifier](#)
property of an `AWS::Route53::RecordSetGroup RecordSet` data type.

Examples

Basic example

In this example, we configure a custom domain and Route 53 record sets for our API.

YAML

```
Resources:
  MyApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      Domain:
        DomainName: www.example.com
        CertificateArn: arn:aws:acm:us-east-1:123456789012:certificate/
abcdef12-3456-7890-abcd-ef1234567890
        EndpointConfiguration: REGIONAL
        Route53:
```

```
            HostedZoneId: ABCDEFGHIJKLMNOP
```

EndpointConfiguration

The endpoint type of a REST API.

Syntax

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

YAML

```
Type: String
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

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

AWS::Serverless::Application

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

> ⓘ **Note**
>
> When you deploy to AWS CloudFormation, AWS SAM transforms your AWS SAM resources into AWS CloudFormation resources. For more information, see Generated AWS CloudFormation 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
  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 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

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 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::Connector

Configures permissions between two resources. For an introduction to connectors, see Managing resource permissions with AWS SAM connectors.

For more information on generated AWS CloudFormation resources, see AWS CloudFormation resources generated when you specify AWS::Serverless::Connector.

To provide feedback on connectors, submit a new issue at the *serverless-application-model AWS GitHub repository*.

> ⓘ **Note**
>
> When you deploy to AWS CloudFormation, AWS SAM transforms your AWS SAM resources into AWS CloudFormation resources. For more information, see Generated AWS CloudFormation resources.

Syntax

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

> ⓘ **Note**
>
> We recommend using the embedded connectors syntax for most use cases. Being embedded within the source resource makes it easier to read and maintain over time. When you need to reference a source resource that is not within the same AWS

SAM template, such as a resource in a nested stack or a shared resource, use the
`AWS::Serverless::Connector` syntax.

Embedded connectors

```
<source-resource-logical-id>:
  Connectors:
    <connector-logical-id:
      Properties:
        Destination: ResourceReference | List of ResourceReference
        Permissions: List
        SourceReference: SourceReference
```

AWS::Serverless::Connector

```
Type: AWS::Serverless::Connector
Properties:
  Destination: ResourceReference | List of ResourceReference
  Permissions: List
  Source: ResourceReference
```

Properties

`Destination`

The destination resource.

Type: ResourceReference | List of ResourceReference

Required: Yes

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

`Permissions`

The permission type that the source resource is allowed to perform on the destination resource.

Read includes AWS Identity and Access Management (IAM) actions that allow reading data from
the resource.

`Write` inclues IAM actions that allow initiating and writing data to a resource.

Valid values: `Read` or `Write`

Type: List

Required: Yes

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

`Source`

The source resource. Required when using the `AWS::Serverless::Connector` syntax.

Type: [ResourceReference](#)

Required: Conditional

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

`SourceReference`

The source resource.

> **ⓘ Note**
>
> Use with the embedded connectors syntax when defining additional properties for the source resource.

Type: [SourceReference](#)

Required: No

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

Examples

Embedded connectors

The following example uses embedded connectors to define a `Write` data connection between an AWS Lambda function and Amazon DynamoDB table:

```
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyTable:
    Type: AWS::Serverless::SimpleTable
  MyFunction:
    Type: AWS::Serverless::Function
    Connectors:
      MyConn:
        Properties:
          Destination:
            Id: MyTable
          Permissions:
            - Write
    ...
```

The following example uses embedded connectors to define `Read` and `Write` permissions:

```
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Connectors:
      MyConn:
        Properties:
          Destination:
            Id: MyTable
          Permissions:
            - Read
            - Write
  MyTable:
    Type: AWS::DynamoDB::Table
    ...
```

The following example uses embedded connectors to define a source resource with a property other than `Id`:

```
Transform: AWS::Serverless-2016-10-31
Transform: AWS::Serverless-2016-10-31
...
Resources:
```

```
  MyApi:
    Type: AWS::Serverless::Api
    Connectors:
      ApitoLambdaConn:
        Properties:
          SourceReference:
            Qualifier: Prod/GET/foobar
          Destination:
            Id: MyTable
          Permissions:
            - Read
            - Write
  MyTable:
    Type: AWS::DynamoDB::Table
    ...
```

AWS::Serverless::Connector

The following example uses the [AWS::Serverless::Connector](#) resource to have an AWS Lambda function read from, and write to an Amazon DynamoDB table:

```
MyConnector:
  Type: AWS::Serverless::Connector
  Properties:
    Source:
      Id: MyFunction
    Destination:
      Id: MyTable
    Permissions:
      - Read
      - Write
```

The following example uses the [AWS::Serverless::Connector](#) resource to have a Lambda function write to an Amazon SNS topic, with both resources in the same template:

```
MyConnector:
  Type: AWS::Serverless::Connector
  Properties:
    Source:
      Id: MyLambda
    Destination:
      Id: MySNSTopic
    Permissions:
```

```
      - Write
```

The following example uses the [AWS::Serverless::Connector](#) resource to have an Amazon SNS topic write to a Lambda function, which then writes to an Amazon DynamoDB table, with all resources in the same template:

```
Transform: AWS::Serverless-2016-10-31
Resources:
  Topic:
    Type: AWS::SNS::Topic
    Properties:
      Subscription:
        - Endpoint: !GetAtt Function.Arn
          Protocol: lambda

  Function:
    Type: AWS::Serverless::Function
    Properties:
      Runtime: nodejs16.x
      Handler: index.handler
      InlineCode: |
        const AWS = require('aws-sdk');
        exports.handler = async (event, context) => {
          const docClient = new AWS.DynamoDB.DocumentClient();
          await docClient.put({
            TableName: process.env.TABLE_NAME,
            Item: {
              id: context.awsRequestId,
              event: JSON.stringify(event)
            }
          }).promise();
        };
      Environment:
        Variables:
          TABLE_NAME: !Ref Table

  Table:
    Type: AWS::Serverless::SimpleTable

  TopicToFunctionConnector:
    Type: AWS::Serverless::Connector
    Properties:
      Source:
```

```
        Id: Topic
      Destination:
        Id: Function
      Permissions:
        - Write


  FunctionToTableConnector:
    Type: AWS::Serverless::Connector
    Properties:
      Source:
        Id: Function
      Destination:
        Id: Table
      Permissions:
        - Write
```

The following is the transformed AWS CloudFormation template from the example above:

```
"FunctionToTableConnectorPolicy": {
  "Type": "AWS::IAM::ManagedPolicy",
  "Metadata": {
    "aws:sam:connectors": {
      "FunctionToTableConnector": {
        "Source": {
          "Type": "AWS::Lambda::Function"
        },
        "Destination": {
          "Type": "AWS::DynamoDB::Table"
        }
      }
    }
  },
  "Properties": {
    "PolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": [
            "dynamodb:PutItem",
            "dynamodb:UpdateItem",
            "dynamodb:DeleteItem",
            "dynamodb:BatchWriteItem",
```

```
            "dynamodb:PartiQLDelete",
            "dynamodb:PartiQLInsert",
            "dynamodb:PartiQLUpdate"
          ],
          "Resource": [
            {
              "Fn::GetAtt": [
                "MyTable",
                "Arn"
              ]
            },
            {
              "Fn::Sub": [
                "${DestinationArn}/index/*",
                {
                  "DestinationArn": {
                    "Fn::GetAtt": [
                      "MyTable",
                      "Arn"
                    ]
                  }
                }
              ]
            }
          ]
        }
      ]
    },
    "Roles": [
      {
        "Ref": "MyFunctionRole"
      }
    ]
  }
}
```

ResourceReference

A reference to a resource that the [AWS::Serverless::Connector](#) resource type uses.

> ⓘ **Note**
>
> For resources in the same template, provide the `Id`. For resources not in the same template, use a combination of other properties. For more information, see [AWS SAM connector reference](#).

Syntax

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

YAML

```
Arn: String
Id: String
Name: String
Qualifier: String
QueueUrl: String
ResourceId: String
RoleName: String
Type: String
```

Properties

Arn

The ARN of a resource.

Type: String

Required: Conditional

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

Id

The [logical ID](#) of a resource in the same template.

> **ⓘ Note**
>
> When `Id` is specified, if the connector generates AWS Identity and Access Management (IAM) policies, the IAM role associated to those policies will be inferred from the resource `Id`. When `Id` is not specified, provide `RoleName` of the resource for connectors to attach generated IAM policies to an IAM role.

Type: String

Required: Conditional

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

Name

The name of a resource.

Type: String

Required: Conditional

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

Qualifier

A qualifier for a resource that narrows its scope. `Qualifier` replaces the * value at the end of a resource constraint ARN. For an example, see [API Gateway invoking a Lambda function](#).

> **ⓘ Note**
>
> Qualifier definition varies per resource type. For a list of supported source and destination resource types, see [AWS SAM connector reference](#).

Type: String

Required: Conditional

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

`QueueUrl`

The Amazon SQS queue URL. This property only applies to Amazon SQS resources.

Type: String

Required: Conditional

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

`ResourceId`

The ID of a resource. For example, the API Gateway API ID.

Type: String

Required: Conditional

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

`RoleName`

The role name associated with a resource.

> ⓘ **Note**
>
> When `Id` is specified, if the connector generates IAM policies, the IAM role associated to those policies will be inferred from the resource `Id`. When `Id` is not specified, provide `RoleName` of the resource for connectors to attach generated IAM policies to an IAM role.

Type: String

Required: Conditional

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

`Type`

The AWS CloudFormation type of a resource. For more information, go to AWS resource and property types reference.

Type: String

Required: Conditional

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

Examples

API Gateway invoking a Lambda function

The following example uses the [AWS::Serverless::Connector](#) resource to allow Amazon API Gateway to invoke an AWS Lambda function.

YAML

```yaml
Transform: AWS::Serverless-2016-10-31
Resources:
  MyRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Effect: Allow
            Action: sts:AssumeRole
            Principal:
              Service: lambda.amazonaws.com
      ManagedPolicyArns:
        - !Sub arn:${AWS::Partition}:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole

  MyFunction:
    Type: AWS::Lambda::Function
    Properties:
      Role: !GetAtt MyRole.Arn
      Runtime: nodejs16.x
      Handler: index.handler
      Code:
        ZipFile: |
          exports.handler = async (event) => {
            return {
              statusCode: 200,
              body: JSON.stringify({
```

```
                "message": "It works!"
            }),
          };
        };

  MyApi:
    Type: AWS::ApiGatewayV2::Api
    Properties:
      Name: MyApi
      ProtocolType: HTTP

  MyStage:
    Type: AWS::ApiGatewayV2::Stage
    Properties:
      ApiId: !Ref MyApi
      StageName: prod
      AutoDeploy: True

  MyIntegration:
    Type: AWS::ApiGatewayV2::Integration
    Properties:
      ApiId: !Ref MyApi
      IntegrationType: AWS_PROXY
      IntegrationUri: !Sub arn:aws:apigateway:${AWS::Region}:lambda:path/2015-03-31/
functions/${MyFunction.Arn}/invocations
      IntegrationMethod: POST
      PayloadFormatVersion: "2.0"

  MyRoute:
    Type: AWS::ApiGatewayV2::Route
    Properties:
      ApiId: !Ref MyApi
      RouteKey: GET /hello
      Target: !Sub integrations/${MyIntegration}

  MyConnector:
    Type: AWS::Serverless::Connector
    Properties:
      Source: # Use 'Id' when resource is in the same template
        Type: AWS::ApiGatewayV2::Api
        ResourceId: !Ref MyApi
        Qualifier: prod/GET/hello # Or "*" to allow all routes
      Destination: # Use 'Id' when resource is in the same template
        Type: AWS::Lambda::Function
```

```
      Arn: !GetAtt MyFunction.Arn
    Permissions:
      - Write

Outputs:
  Endpoint:
    Value: !Sub https://${MyApi}.execute-api.${AWS::Region}.${AWS::URLSuffix}/prod/
hello
```

SourceReference

A reference to a source resource that the AWS::Serverless::Connector resource type uses.

Syntax

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

YAML

```
Qualifier: String
```

Properties

Qualifier

A qualifier for a resource that narrows its scope. `Qualifier` replaces the * value at the end of a resource constraint ARN.

> ℹ️ **Note**
>
> Qualifier definition varies per resource type. For a list of supported source and destination resource types, see AWS SAM connector reference.

Type: String

Required: Conditional

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

Examples

The following example uses embedded connectors to define a source resource with a property other than Id:

```
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyApi:
    Type: AWS::Serverless::Api
    Connectors:
      ApitoLambdaConn:
        Properties:
          SourceReference:
            Qualifier: Prod/GET/foobar
          Destination:
            Id: MyTable
          Permissions:
            - Read
            - Write
  MyTable:
    Type: AWS::DynamoDB::Table
    ...
```

AWS::Serverless::Function

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

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

> **ⓘ Note**
>
> When you deploy to AWS CloudFormation, AWS SAM transforms your AWS SAM resources into AWS CloudFormation resources. For more information, see Generated AWS CloudFormation resources.

Syntax

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

YAML

```
Type: AWS::Serverless::Function
Properties:
  Architectures: List
  AssumeRolePolicyDocument: JSON
  AutoPublishAlias: String
  AutoPublishAliasAllProperties: Boolean
  AutoPublishCodeSha256: String
  CodeSigningConfigArn: String
  CodeUri: String | FunctionCode
  DeadLetterQueue: Map | DeadLetterQueue
  DeploymentPreference: DeploymentPreference
  Description: String
  Environment: Environment
  EphemeralStorage: EphemeralStorage
  EventInvokeConfig: EventInvokeConfiguration
  Events: EventSource
  FileSystemConfigs: List
  FunctionName: String
  FunctionUrlConfig: FunctionUrlConfig
  Handler: String
  ImageConfig: ImageConfig
  ImageUri: String
  InlineCode: String
  KmsKeyArn: String
  Layers: List
  LoggingConfig: LoggingConfig
  MemorySize: Integer
  PackageType: String
  PermissionsBoundary: String
  Policies: String | List | Map
  PropagateTags: Boolean
  ProvisionedConcurrencyConfig: ProvisionedConcurrencyConfig
  ReservedConcurrentExecutions: Integer
  Role: String
  RolePath: String
  Runtime: String
```

```
      RuntimeManagementConfig: RuntimeManagementConfig
      SnapStart: SnapStart
      Tags: Map
      Timeout: Integer
      Tracing: String
      VersionDescription: String
      VpcConfig: VpcConfig
```

Properties

Architectures

The instruction set architecture for the function.

For more information about this property, see [Lambda instruction set architectures](#) in the *AWS Lambda Developer Guide*.

Valid values: One of `x86_64` or `arm64`

Type: List

Required: No

Default: `x86_64`

AWS CloudFormation compatibility: This property is passed directly to the [Architectures](#) property of an `AWS::Lambda::Function` resource.

AssumeRolePolicyDocument

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

Type: JSON

Required: No

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

AutoPublishAlias

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

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. For general information about generated AWS CloudFormation resources, see Generated AWS CloudFormation resources.

Type: String

Required: No

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

AutoPublishAliasAllProperties

Specifies when a new AWS::Lambda::Version is created. When true, a new Lambda version is created when any property in the Lambda function is modified. When false, a new Lambda version is created only when any of the following properties are modified:

- Environment, MemorySize, or SnapStart.

- Any change that results in an update to the Code property, such as CodeDict, ImageUri, or InlineCode.

This property requires AutoPublishAlias to be defined.

If AutoPublishSha256 is also specified, its behavior takes precedence over AutoPublishAliasAllProperties: true.

Type: Boolean

Required: No

Default value: false

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

AutoPublishCodeSha256

When used, this string works with the `CodeUri` value to determine if a new Lambda version needs to be published. This property is often used to resolve the following deployment issue: A deployment package is stored in an Amazon S3 location and is replaced by a new deployment package with 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).

This problem is marked by an AWS SAM template having 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
- The `CodeUri` property is set and doesn't change between deployments.

In this scenario, updating `AutoPublishCodeSha256` results in a new Lambda version being created successfully. However, new function code deployed to Amazon S3 will not be recognized. To recognize new function code, consider using versioning in your Amazon S3 bucket. Specify the `Version` property for your Lambda function and configure your bucket to always use the latest deployment package.

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

Type: String

Required: No

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

CodeSigningConfigArn

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

Type: String

Required: No

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

CodeUri

The code for the function. Accepted values include:

- The function's Amazon S3 URI. For example, s3://bucket-123456789/sam-app/1234567890abcdefg.

- The local path to the function. For example, hello_world/.

- A FunctionCode object.

> **ⓘ Note**
>
> If you provide a function's Amazon S3 URI or FunctionCode object, you must reference a
> valid Lambda deployment package.
> If you provide a local file path, use the AWS SAM CLI to upload the local file at
> deployment. To learn more, see Using the AWS SAM CLI to upload local files at
> deployment.
> If you use intrinsic functions in CodeUri property, AWS SAM will not be able to
> correctly parse the values. Consider using AWS::LanguageExtensions transform instead.

Type: [String | FunctionCode]

Required: Conditional. When PackageType is set to Zip, one of CodeUri or InlineCode is
required.

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

DeadLetterQueue

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

> **ⓘ Note**
>
> If your Lambda function's event source is an Amazon SQS queue, configure a dead-letter
> queue for the source queue, not for the Lambda function. The dead-letter queue that
> you configure for a function is used for the function's asynchronous invocation queue,
> not for event source queues.

Type: Map | DeadLetterQueue

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

Required: No

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

See also: For more information about this property, see Deploying serverless applications
gradually.

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.

`EphemeralStorage`

An object that specifies the disk space, in MB, available to your Lambda function in `/tmp`.

For more information about this property, see [Lambda execution environment](#) in the *AWS Lambda Developer Guide*.

Type: [EphemeralStorage](#)

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [EphemeralStorage](#) property of an `AWS::Lambda::Function` resource.

`EventInvokeConfig`

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

Type: [EventInvokeConfiguration](#)

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

Required: No

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

FileSystemConfigs

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

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

Type: List

Required: No

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

FunctionName

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

Type: String

Required: No

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

FunctionUrlConfig

The object that describes a function URL. A function URL is an HTTPS endpoint that you can use to invoke your function.

For more information, see Function URLs in the *AWS Lambda Developer Guide*.

Type: FunctionUrlConfig

Required: No

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

Handler

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

Type: String

Required: Conditional

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

ImageConfig

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

Type: ImageConfig

Required: No

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

ImageUri

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

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

Building your application with necessary `Metadata` entries takes precedence over `ImageUri`, so if you specify both then `ImageUri` is ignored.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [ImageUri](#) property of the `AWS::Lambda::Function` Code data type.

InlineCode

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

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

Type: String

Required: Conditional

AWS CloudFormation compatibility: This property is passed directly to the [ZipFile](#) property of the `AWS::Lambda::Function` Code data type.

KmsKeyArn

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

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [KmsKeyArn](#) property of an `AWS::Lambda::Function` resource.

Layers

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

Type: List

Required: No

AWS CloudFormation compatibility: This property is passed directly to the <u>Layers</u> property of an `AWS::Lambda::Function` resource.

`LoggingConfig`

The function's Amazon CloudWatch Logs configuration settings.

Type: <u>LoggingConfig</u>

Required: No

AWS CloudFormation compatibility: This property is passed directly to the <u>LoggingConfig</u> property of an `AWS::Lambda::Function` resource.

`MemorySize`

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

Type: Integer

Required: No

AWS CloudFormation compatibility: This property is passed directly to the <u>MemorySize</u> property of an `AWS::Lambda::Function` resource.

`PackageType`

The deployment package type of the Lambda function. For more information, see <u>Lambda deployment packages</u> in the *AWS Lambda Developer Guide*.

Notes:

1. If this property is set to `Zip` (default), then either `CodeUri` or `InlineCode` applies, and `ImageUri` is ignored.

2. If this property is set to `Image`, then only `ImageUri` applies, and both `CodeUri` and `InlineCode` are ignored. The Amazon ECR repository required to store the function's container image can be auto created by the AWS SAM CLI. For more information, see <u>sam deploy</u>.

Valid values: `Zip` or `Image`

Type: String

Required: No

Default: `Zip`

AWS CloudFormation compatibility: This property is passed directly to the [PackageType](#) property of an `AWS::Lambda::Function` resource.

PermissionsBoundary

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

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [PermissionsBoundary](#) property of an `AWS::IAM::Role` resource.

Policies

Permission policies for this function. Policies will be appended to the function's default AWS Identity and Access Management (IAM) execution role.

This property accepts a single value or list of values. Allowed values include:

- [AWS SAM policy templates](#).
- The ARN of an [AWS managed policy](#) or [customer managed policy](#).
- The name of an AWS managed policy from the following [list](#).
- An [inline IAM policy](#) formatted in YAML as a map.

> **ⓘ Note**
>
> If you set the `Role` property, 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.

PropagateTags

Indicate whether or not to pass tags from the `Tags` property to your [AWS::Serverless::Function](#) generated resources. Specify `True` to propagate tags in your generated resources.

Type: Boolean

Required: No

Default: `False`

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

ProvisionedConcurrencyConfig

The provisioned concurrency configuration of a function's alias.

> ℹ️ **Note**
>
> `ProvisionedConcurrencyConfig` can be specified only if the `AutoPublishAlias` is set. Otherwise, an error results.

Type: [ProvisionedConcurrencyConfig](#)

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [ProvisionedConcurrencyConfig](#) property of an `AWS::Lambda::Alias` resource.

ReservedConcurrentExecutions

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

For more information about this property, see [Lambda Function Scaling](#) in the *AWS Lambda Developer Guide*.

Type: Integer

Required: No

AWS CloudFormation compatibility: This property is passed directly to the
[ReservedConcurrentExecutions](#) property of an `AWS::Lambda::Function` resource.

Role

The ARN of an IAM role to use as this function's execution role.

Type: String

Required: No

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

RolePath

The path to the function's IAM execution role.

Use this property when the role is generated for you. Do not use when the role is specified with
the `Role` property.

Type: String

Required: Conditional

AWS CloudFormation compatibility: This property is passed directly to the [Path](#) property of an
`AWS::IAM::Role` resource.

Runtime

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

> **ⓘ Note**
>
> If you specify the `provided` identifier for this property, you can use the `Metadata`
> resource attribute to instruct AWS SAM to build the custom runtime that this function

requires. For more information about building custom runtimes, see Building custom
runtimes.

Type: String

Required: Conditional

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

RuntimeManagementConfig

Configure runtime management options for your Lambda functions such as runtime
environment updates, rollback behavior, and selecting a specific runtime version. To learn more,
see Lambda runtime updates in the *AWS Lambda Developer Guide*.

Type: RuntimeManagementConfig

Required: No

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

SnapStart

Create a snapshot of any new Lambda function version. A snapshot is a cached state of your
initialized function, including all of its dependencies. The function is initialized just once and the
cached state is reused for all future invocations, improving application performance by reducing
the number of times your function must be initialized. To learn more, see Improving startup
performance with Lambda SnapStart in the *AWS Lambda Developer Guide*.

Type: SnapStart

Required: No

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

Tags

A map (string to string) that specifies the tags added to this function. For details about valid
keys and values for tags, see Tag Key and Value Requirements in the *AWS Lambda Developer
Guide*.

When the stack is created, AWS SAM automatically adds a `lambda:createdBy:SAM` tag to this Lambda function, and to the default roles that are generated for this function.

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 (whereas in AWS CloudFormation this property consists of a list of `Tag` objects). Also, AWS SAM automatically adds a `lambda:createdBy:SAM` tag to this Lambda function, and to the default roles that are generated for this function.

`Timeout`

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

Type: Integer

Required: No

Default: 3

AWS CloudFormation compatibility: This property is passed directly to the [Timeout](#) property of an `AWS::Lambda::Function` resource.

`Tracing`

The string that specifies the function's X-Ray tracing mode.

- `Active` – Activates X-Ray tracing for the function.
- `Disabled` – Deactivates X-Ray for the function.
- `PassThrough` – Activates X-Ray tracing for the function. Sampling decision is delegated to the downstream services.

If specified as `Active` or `PassThrough` and the `Role` property is not set, AWS SAM adds the `arn:aws:iam::aws:policy/AWSXrayWriteOnlyAccess` policy to the Lambda execution role that it creates for you.

For more information about X-Ray, see [Using AWS Lambda with AWS X-Ray](#) in the *AWS Lambda Developer Guide*.

Valid values: [`Active|Disabled|PassThrough`]

Type: String

Required: No

AWS CloudFormation compatibility: This property is similar to the <u>TracingConfig</u> property of an `AWS::Lambda::Function` resource.

`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 <u>Description</u> property of an `AWS::Lambda::Version` resource.

`VpcConfig`

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

Type: <u>VpcConfig</u>

Required: No

AWS CloudFormation compatibility: This property is passed directly to the <u>VpcConfig</u> 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 <u>Ref</u> in the *AWS CloudFormation User Guide*.

Fn::GetAtt

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

For more information about using `Fn::GetAtt`, see [`Fn::GetAtt`](#) in the *AWS CloudFormation User Guide*.

`Arn`

    The ARN of the underlying Lambda function.

Examples

Simple function

The following is a basic example of an [AWS::Serverless::Function](#) resource of package type `Zip` (default) and function code in an Amazon S3 bucket.

YAML

```
Type: AWS::Serverless::Function
Properties:
  Handler: index.handler
  Runtime: python3.9
  CodeUri: s3://bucket-name/key-name
```

Function properties example

The following is an example of an [AWS::Serverless::Function](#) of package type `Zip` (default) that uses `InlineCode`, `Layers`, `Tracing`, `Policies`, `Amazon 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: python3.9
  InlineCode: |
```

```
    def handler(event, context):
      print("Hello, world!")
  ReservedConcurrentExecutions: 30
  Layers:
    - Ref: MyLayer
  Tracing: Active
  Timeout: 120
  FileSystemConfigs:
    - Arn: !Ref MyEfsFileSystem
      LocalMountPath: /mnt/EFS
  Policies:
    - AWSLambdaExecute
    - Version: '2012-10-17'
      Statement:
        - Effect: Allow
          Action:
            - s3:GetObject
            - s3:GetObjectACL
          Resource: 'arn:aws:s3:::my-bucket/*'
  Events:
    ApiEvent:
      Type: Api
      Properties:
        Path: /path
        Method: get
```

ImageConfig example

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

YAML

```
HelloWorldFunction:
  Type: AWS::Serverless::Function
  Properties:
    PackageType: Image
    ImageUri: account-id.dkr.ecr.region.amazonaws.com/ecr-repo-name:image-name
    ImageConfig:
      Command:
        - "app.lambda_handler"
      EntryPoint:
        - "entrypoint1"
      WorkingDirectory: "workDir"
```

RuntimeManagementConfig examples

A Lambda function configured to update its runtime environment according to current behavior:

```
TestFunction
  Type: AWS::Serverless::Function
  Properties:
    ...
    Runtime: python3.9
    RuntimeManagementConfig:
      UpdateRuntimeOn: Auto
```

A Lambda function configured to update its runtime environment when the function is updated:

```
TestFunction
  Type: AWS::Serverless::Function
  Properties:
    ...
    Runtime: python3.9
    RuntimeManagementConfig:
      UpdateRuntimeOn: FunctionUpdate
```

A Lambda function configured to update its runtime environment manually:

```
TestFunction
  Type: AWS::Serverless::Function
  Properties:
    ...
    Runtime: python3.9
    RuntimeManagementConfig:
      RuntimeVersionArn: arn:aws:lambda:us-
east-1::runtime:4c459dd0104ee29ec65dcad056c0b3ddbe20d6db76b265ade7eda9a066859b1e
      UpdateRuntimeOn: Manual
```

SnapStart examples

Example of a Lambda function with SnapStart turned on for future versions:

```
TestFunc
  Type: AWS::Serverless::Function
  Properties:
    ...
```

```
    SnapStart:
      ApplyOn: PublishedVersions
```

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 Dead-letter queues in the *AWS Lambda Developer Guide*.

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.

Valid values: SNS, SQS

Type: String

Required: Yes

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

Examples

DeadLetterQueue

Dead Letter Queue example for an SNS topic.

YAML

```
DeadLetterQueue:
  Type: SNS
  TargetArn: arn:aws:sns:us-east-2:123456789012:my-topic
```

DeploymentPreference

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

> **ⓘ Note**
>
> You must specify an `AutoPublishAlias` in your AWS::Serverless::Function 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
    PassthroughCondition: Boolean
    Role: String
    TriggerConfigurations: List
    Type: String
```

Properties

Alarms

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

This property accepts the `Fn::If` intrinsic function. See the Examples section at the bottom of this topic for an example template that uses `Fn::If`.

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

Required: No

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

PassthroughCondition

If True, and if this deployment preference is enabled, the function's Condition will be passed through to the generated CodeDeploy resource. Generally, you should set this to True. Otherwise, the CodeDeploy resource would be created even if the function's Condition resolves to False.

Type: Boolean

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](#).

Type: String

Required: Yes

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

Examples

DeploymentPreference with pre- and post-traffic hooks.

Example deployment preference that contains pre- and post-traffic hooks.

YAML

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

DeploymentPreference with Fn::If intrinsic function

Example deployment preference that uses `Fn::If` for configuring alarms. In this example, `Alarm1` will be configured if `MyCondition` is `true`, and `Alarm2` and `Alarm5` will be configured if `MyCondition` is `false`.

YAML

```
DeploymentPreference:
  Enabled: true
  Type: Canary10Percent10Minutes
  Alarms:
    Fn::If:
      - MyCondition
      - - Alarm1
```

```
        - - Alarm2
          - Alarm5
```

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

Required: No

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

PreTraffic

Lambda function that is run before traffic shifting.

Type: String

Required: No

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

Examples

Hooks

Example hook functions

YAML

```
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
    MaximumEventAgeInSeconds: Integer
    MaximumRetryAttempts: Integer
```

Properties

`DestinationConfig`

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

Type: EventInvokeDestinationConfiguration

Required: No

AWS CloudFormation compatibility: This property is similar to the <u>DestinationConfig</u> 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 <u>MaximumEventAgeInSeconds</u> 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 <u>MaximumRetryAttempts</u> property of an `AWS::Lambda::EventInvokeConfig` resource.

Examples

MaximumEventAgeInSeconds

MaximumEventAgeInSeconds example

YAML

```
EventInvokeConfig:
  MaximumEventAgeInSeconds: 60
  MaximumRetryAttempts: 2
  DestinationConfig:
```

```
      OnSuccess:
        Type: SQS
        Destination: arn:aws:sqs:us-west-2:012345678901:my-queue
      OnFailure:
        Type: Lambda
        Destination: !GetAtt DestinationLambda.Arn
```

EventInvokeDestinationConfiguration

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

Syntax

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

YAML

```
    OnFailure: OnFailure
    OnSuccess: OnSuccess
```

Properties

OnFailure

   A destination for events that failed processing.

   Type: OnFailure

   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

   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
      Destination: arn:aws:sqs:us-west-2:012345678901:my-queue
    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 <u>OnFailure</u> 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
      Destination: !GetAtt DestinationLambda.Arn  # Arn of a Lambda function declared
  in the template file.
```

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 tempate 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
      Destination: !GetAtt DestinationLambda.Arn  # Arn of a Lambda function declared
  in the template file.
```

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 tempate 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: AlexaSkill | Api | CloudWatchEvent | CloudWatchLogs | Cognito
  | DocumentDB | DynamoDB | EventBridgeRule | HttpApi | IoTRule | Kinesis | MQ | MSK
  | S3 | Schedule | ScheduleV2 | SelfManagedKafka | SNS | SQS
  Type: String
```

Properties

Properties

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

Type: [AlexaSkill](#) | [Api](#) | [CloudWatchEvent](#) | [CloudWatchLogs](#) | [Cognito](#) | [DocumentDB](#) | [DynamoDB](#) | [EventBridgeRule](#) | [HttpApi](#) | [IoTRule](#) | [Kinesis](#) | [MQ](#) | [MSK](#) | [S3](#) | [Schedule](#) | [ScheduleV2](#) | [SelfManagedKafka](#) | [SNS](#) | [SQS](#)

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: AlexaSkill, Api, CloudWatchEvent, CloudWatchLogs, Cognito, DocumentDB, DynamoDB, EventBridgeRule, HttpApi, IoTRule, Kinesis, MQ, MSK, S3, Schedule, ScheduleV2, SelfManagedKafka, 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

APIEvent

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 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 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
Method: String
Path: String
RequestModel: RequestModel
RequestParameters: List of [ String | RequestParameter ]
RestApiId: String
TimeoutInMillis: Integer
```

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

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](#) resource.

Type: [RequestModel](#)

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`.

A list can contain both parameter name strings and [RequestParameter](#) objects. For strings, the `Required` and `Caching` properties will default to `false`.

Type: List of [String | [RequestParameter](#)]

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](#) resource defined in this template.

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

This cannot reference an [AWS::Serverless::Api](#) resource defined in another template.

Type: String

Required: No

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

TimeoutInMillis

Custom timeout between 50 and 29,000 milliseconds.

> ⓘ **Note**
>
> When you specify this property, AWS SAM modifies your OpenAPI definition. The OpenAPI definition must be specified inline using the DefinitionBody property.

Type: Integer

Required: No

Default: 29,000 milliseconds or 29 seconds

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

Examples

Basic example

YAML

```
Events:
  ApiEvent:
    Type: Api
```

```
    Properties:
      Path: /path
      Method: get
      RequestParameters:
        - method.request.header.Authorization
        - method.request.querystring.keyword:
            Required: true
            Caching: false
```

ApiFunctionAuth

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

Syntax

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

YAML

```
ApiKeyRequired: Boolean
AuthorizationScopes: List
Authorizer: String
InvokeRole: String
OverrideApiAuth: Boolean
ResourcePolicy: ResourcePolicyStatement
```

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 a global authorizer specified for your `AWS::Serverless::Api` resource, you can override the authorizer by setting `Authorizer` to `NONE`. For an example, see [Override a global authorizer for your Amazon API Gateway REST API](#).

> ⓘ **Note**
>
> If you use the `DefinitionBody` property of an `AWS::Serverless::Api` resource to describe your API, you must use `OverrideApiAuth` with `Authorizer` to override your global authorizer. See [OverrideApiAuth](#) for more information.

Valid values: `AWS_IAM`, `NONE`, or the logical ID for any authorizer defined in your AWS SAM template.

Type: String

Required: No

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

`InvokeRole`

Specifies the `InvokeRole` to use for `AWS_IAM` authorization.

Type: String

Required: No

Default: CALLER_CREDENTIALS

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

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

OverrideApiAuth

Specify as `true` to override the global authorizer configuration of your `AWS::Serverless::Api` resource. This property is only required if you specify a global authorizer and use the `DefinitionBody` property of an `AWS::Serverless::Api` resource to describe your API.

> ⓘ **Note**
>
> When you specify `OverrideApiAuth` as `true`, AWS SAM will override your global authorizer with any values provided for `ApiKeyRequired`, `Authorizer`, or `ResourcePolicy`. Therefore, at least one of these properties must also be specified when using `OverrideApiAuth`. For an example, see [Override a global authorizer when DefinitionBody for AWS::Serverless::Api is specified](#).

Type: Boolean

Required: No

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

ResourcePolicy

Configure Resource Policy for this path on an API.

Type: [ResourcePolicyStatement](#)

Required: No

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

Examples

Function-Auth

The following example specifies authorization at the function level.

YAML

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

Override a global authorizer for your Amazon API Gateway REST API

You can specify a global authorizer for your `AWS::Serverless::Api` resource. The following is an example that configures a global default authorizer:

```yaml
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyApiWithLambdaRequestAuth:
    Type: AWS::Serverless::Api
    Properties:
      ...
      Auth:
        Authorizers:
          MyLambdaRequestAuth:
            FunctionArn: !GetAtt MyAuthFn.Arn
        DefaultAuthorizer: MyLambdaRequestAuth
```

To override the default authorizer for your AWS Lambda function, you can specify `Authorizer` as `NONE`. The following is an example:

```yaml
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  ...
  MyFn:
    Type: AWS::Serverless::Function
```

```
      Properties:
        ...
      Events:
        LambdaRequest:
          Type: Api
          Properties:
            RestApiId: !Ref MyApiWithLambdaRequestAuth
            Method: GET
            Auth:
              Authorizer: NONE
```

Override a global authorizer when DefinitionBody for AWS::Serverless::Api is specified

When using the `DefinitionBody` property to describe your `AWS::Serverless::Api` resource, the previous override method does not work. The following is an example of using the `DefinitionBody` property for an `AWS::Serverless::Api` resource:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyApiWithLambdaRequestAuth:
    Type: AWS::Serverless::Api
    Properties:
      ...
      DefinitionBody:
        swagger: 2.0
        ...
        paths:
          /lambda-request:
            ...
      Auth:
        Authorizers:
          MyLambdaRequestAuth:
            FunctionArn: !GetAtt MyAuthFn.Arn
        DefaultAuthorizer: MyLambdaRequestAuth
```

To override the global authorizer, use the `OverrideApiAuth` property. The following is an example that uses `OverrideApiAuth` to override the global authorizer with the value provided for `Authorizer`:

```
AWSTemplateFormatVersion: '2010-09-09'
```

```
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyApiWithLambdaRequestAuth:
    Type: AWS::Serverless::Api
    Properties:
      ...
      DefinitionBody:
        swagger: 2-0
        ...
        paths:
          /lambda-request:
            ...
      Auth:
        Authorizers:
          MyLambdaRequestAuth:
            FunctionArn: !GetAtt MyAuthFn.Arn
        DefaultAuthorizer: MyLambdaRequestAuth

  MyAuthFn:
    Type: AWS::Serverless::Function
    ...

  MyFn:
    Type: AWS::Serverless::Function
      Properties:
        ...
        Events:
          LambdaRequest:
            Type: Api
            Properties:
              RestApiId: !Ref MyApiWithLambdaRequestAuth
              Method: GET
              Auth:
                Authorizer: NONE
                OverrideApiAuth: true
              Path: /lambda-token
```

ResourcePolicyStatement

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

Syntax

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

YAML

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

Properties

`AwsAccountBlacklist`

The AWS accounts to block.

Type: List of String

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 of String

Required: No

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

CustomStatements

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

Type: List

Required: No

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

IntrinsicVpcBlacklist

The list of virtual private clouds (VPCs) to block, where each VPC is specified as a reference such as a [dynamic reference](#) or the Ref [intrinsic function](#). For an example use of this property, see the Examples section at the bottom of this page.

Type: List

Required: No

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

IntrinsicVpcWhitelist

The list of VPCs to allow, where each VPC is specified as a reference such as a [dynamic reference](#) or the Ref [intrinsic function](#).

Type: List

Required: No

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

IntrinsicVpceBlacklist

The list of VPC endpoints to block, where each VPC endpoint is specified as a reference such as a [dynamic reference](#) or the Ref [intrinsic function](#).

Type: List

Required: No

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

`IntrinsicVpceWhitelist`

The list of VPC endpoints to allow, where each VPC endpoint is specified as a reference such as a [dynamic reference](#) or the `Ref` [intrinsic function](#). For an example use of this property, see the Examples section at the bottom of this page.

Type: List

Required: No

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

`IpRangeBlacklist`

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

Type: List

Required: No

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

`IpRangeWhitelist`

The IP addresses or address ranges to allow.

Type: List

Required: No

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

`SourceVpcBlacklist`

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

Type: List

Required: No

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

`SourceVpcWhitelist`

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

Type: List

Required: No

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

Examples

Resource Policy Example

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

YAML

```
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

Configures a Request Model for a specific Api+Path+Method.

Syntax

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

YAML

```
    Model: String
    Required: Boolean
    ValidateBody: Boolean
    ValidateParameters: Boolean
```

Properties

`Model`

Name of a model defined in the Models property of the AWS::Serverless::Api.

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 the OpenApi definition for the 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.

ValidateBody

Specifies whether API Gateway uses the `Model` to validate the request body. For more information, see [Enable request validation in API Gateway](#) 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.

ValidateParameters

Specifies whether API Gateway uses the `Model` to validate request path parameters, query strings, and headers. For more information, see [Enable request validation in API Gateway](#) 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.

Examples

Request Model

Request Model Example

YAML

```
RequestModel:
```

```
    Model: User
    Required: true
    ValidateBody: true
    ValidateParameters: 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 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

```
  Enabled: Boolean
  EventBusName: String
  Input: String
  InputPath: String
  Pattern: EventPattern
  State: String
```

Properties

`Enabled`

Indicates whether the rule is enabled.

To disable the rule, set this property to `false`.

> **ⓘ Note**
>
> Specify either the `Enabled` or `State` property, but not both.

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.

`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.

State

The state of the rule.

Accepted values: `DISABLED | ENABLED`

> ⓘ **Note**
>
>   Specify either the `Enabled` or `State` property, but not both.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the State 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:
    Enabled: false
    Input: '{"Key": "Value"}'
    Pattern:
      detail:
        state:
          - running
```

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

```
  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

```
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
```

DocumentDB

The object describing a `DocumentDB` event source type. For more information, see [Using AWS Lambda with Amazon DocumentDB](#) in the *AWS Lambda Developer Guide*.

Syntax

To declare this entity in your AWS SAM template, use the following syntax.

YAML

```yaml
BatchSize: Integer
Cluster: String
CollectionName: String
DatabaseName: String
Enabled: Boolean
FilterCriteria: FilterCriteria
FullDocument: String
MaximumBatchingWindowInSeconds: Integer
SecretsManagerKmsKeyId: String
SourceAccessConfigurations: List
StartingPosition: String
StartingPositionTimestamp: Double
```

Properties

`BatchSize`

The maximum number of items to retrieve in a single batch.

Type: Integer

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [`BatchSize`](#) property of an `AWS::Lambda::EventSourceMapping` resource.

`Cluster`

The Amazon Resource Name (ARN) of the Amazon DocumentDB cluster.

Type: String

Required: Yes

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

`CollectionName`

The name of the collection to consume within the database. If you do not specify a collection, Lambda consumes all collections.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the CollectionName property of an `AWS::Lambda::EventSourceMapping DocumentDBEventSourceConfig` data type.

`DatabaseName`

The name of the database to consume within the Amazon DocumentDB cluster.

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the DatabaseName property of an `AWS::Lambda::EventSourceMapping DocumentDBEventSourceConfig` data type.

`Enabled`

If `true`, the event source mapping is active. To pause polling and invocation, set to `false`.

Type: Boolean

Required: No

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

`FilterCriteria`

An object that defines the criteria that determines whether Lambda should process an event. For more information, see Lambda event filtering in the *AWS Lambda Developer Guide*.

Type: FilterCriteria

Required: No

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

`FullDocument`

Determines what Amazon DocumentDB sends to your event stream during document update operations. If set to `UpdateLookup`, Amazon DocumentDB sends a delta describing the changes, along with a copy of the entire document. Otherwise, Amazon DocumentDB sends only a partial document that contains the changes.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the `FullDocument` property of an `AWS::Lambda::EventSourceMapping` `DocumentDBEventSourceConfig` data type.

`MaximumBatchingWindowInSeconds`

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

Type: Integer

Required: No

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

`SecretsManagerKmsKeyId`

The AWS Key Management Service (AWS KMS) key ID of a customer managed key from AWS Secrets Manager. Required when you use a customer managed key from Secrets Manager with a Lambda execution role that doesn't include the `kms:Decrypt` permission.

The value of this property is a UUID. For example: `1abc23d4-567f-8ab9-cde0-1fab234c5d67`.

Type: String

Required: Conditional

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

SourceAccessConfigurations

An array of the authentication protocol or virtual host. Specify this using the [SourceAccessConfigurations](#) data type.

For the DocumentDB event source type, the only valid configuration type is BASIC_AUTH.

- BASIC_AUTH – The Secrets Manager secret that stores your broker credentials. For this type, the credential must be in the following format: {"username": "your-username", "password": "your-password"}. Only one object of type BASIC_AUTH is allowed.

Type: List

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the [SourceAccessConfigurations](#) property of an AWS::Lambda::EventSourceMapping resource.

StartingPosition

The position in a stream from which to start reading.

- AT_TIMESTAMP – Specify a time from which to start reading records.
- LATEST – Read only new records.
- TRIM_HORIZON – Process all available records.

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the [StartingPosition](#) property of an AWS::Lambda::EventSourceMapping resource.

StartingPositionTimestamp

The time from which to start reading, in Unix time seconds. Define StartingPositionTimestamp when StartingPosition is specified as AT_TIMESTAMP.

Type: Double

Required: No

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

Examples

Amazon DocumentDB event source

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
    ...
      Events:
        MyDDBEvent:
          Type: DocumentDB
          Properties:
            Cluster: "arn:aws:rds:us-west-2:123456789012:cluster:docdb-2023-01-01"
            BatchSize: 10
            MaximumBatchingWindowInSeconds: 5
            DatabaseName: "db1"
            CollectionName: "collection1"
            FullDocument: "UpdateLookup"
            SourceAccessConfigurations:
              - Type: BASIC_AUTH
                URI: "arn:aws:secretsmanager:us-west-2:123456789012:secret:doc-db"
```

DynamoDB

The object describing a DynamoDB event source type. For more information, see Using AWS
Lambda with Amazon DynamoDB in the *AWS Lambda Developer Guide*.

AWS SAM generates an AWS::Lambda::EventSourceMapping resource when this event type is set.

Syntax

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

YAML

```
    BatchSize: Integer
    BisectBatchOnFunctionError: Boolean
    DestinationConfig: DestinationConfig
    Enabled: Boolean
    FilterCriteria: FilterCriteria
    FunctionResponseTypes: List
    MaximumBatchingWindowInSeconds: Integer
    MaximumRecordAgeInSeconds: Integer
    MaximumRetryAttempts: Integer
    ParallelizationFactor: Integer
    StartingPosition: String
    StartingPositionTimestamp: Double
    Stream: String
    TumblingWindowInSeconds: Integer
```

Properties

`BatchSize`

> The maximum number of items to retrieve in a single batch.
>
> *Type*: Integer
>
> *Required*: No
>
> *Default*: 100
>
> *AWS CloudFormation compatibility*: This property is passed directly to the `BatchSize` property of an `AWS::Lambda::EventSourceMapping` resource.
>
> *Minimum*: 1
>
> *Maximum*: 1000

`BisectBatchOnFunctionError`

> If the function returns an error, split the batch in two and retry.
>
> *Type*: Boolean
>
> *Required*: No

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

DestinationConfig

An Amazon Simple Queue Service (Amazon SQS) queue or Amazon Simple Notification Service (Amazon SNS) topic destination for discarded records.

Type: DestinationConfig

Required: No

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

Enabled

Disables the event source mapping to pause polling and invocation.

Type: Boolean

Required: No

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

FilterCriteria

A object that defines the criteria to determine whether Lambda should process an event. For more information, see AWS Lambda event filtering in the *AWS Lambda Developer Guide*.

Type: FilterCriteria

Required: No

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

FunctionResponseTypes

A list of the response types currently applied to the event source mapping. For more information, see Reporting batch item failures in the *AWS Lambda Developer Guide*.

Valid values: `ReportBatchItemFailures`

Type: List

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [FunctionResponseTypes](#) property of an `AWS::Lambda::EventSourceMapping` resource.

`MaximumBatchingWindowInSeconds`

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

Type: Integer

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [MaximumBatchingWindowInSeconds](#) property of an `AWS::Lambda::EventSourceMapping` resource.

`MaximumRecordAgeInSeconds`

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

Type: Integer

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [MaximumRecordAgeInSeconds](#) property of an `AWS::Lambda::EventSourceMapping` resource.

`MaximumRetryAttempts`

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

Type: Integer

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [MaximumRetryAttempts](#) property of an `AWS::Lambda::EventSourceMapping` resource.

`ParallelizationFactor`

The number of batches to process from each shard concurrently.

Type: Integer

Required: No

AWS CloudFormation compatibility: This property is passed directly to the
[ParallelizationFactor](#) property of an AWS::Lambda::EventSourceMapping resource.

StartingPosition

The position in a stream from which to start reading.

- AT_TIMESTAMP – Specify a time from which to start reading records.

- LATEST – Read only new records.

- TRIM_HORIZON – Process all available records.

Valid values: AT_TIMESTAMP | LATEST | TRIM_HORIZON

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the [StartingPosition](#)
property of an AWS::Lambda::EventSourceMapping resource.

StartingPositionTimestamp

The time from which to start reading, in Unix time seconds. Define
StartingPositionTimestamp when StartingPosition is specified as AT_TIMESTAMP.

Type: Double

Required: No

AWS CloudFormation compatibility: This property is passed directly to the
[StartingPositionTimestamp](#) property of an AWS::Lambda::EventSourceMapping
resource.

Stream

The Amazon Resource Name (ARN) of the DynamoDB stream.

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the [EventSourceArn](#)
property of an AWS::Lambda::EventSourceMapping resource.

`TumblingWindowInSeconds`

> The duration, in seconds, of a processing window. The valid range is 1 to 900 (15 minutes).
>
> For more information, see [Tumbling windows](#) in the *AWS Lambda Developer Guide*.
>
> *Type*: Integer
>
> *Required*: No
>
> *AWS CloudFormation compatibility*: This property is passed directly to the [TumblingWindowInSeconds](#) property of an `AWS::Lambda::EventSourceMapping` resource.

Examples

DynamoDB event source for existing DynamoDB table

DynamoDB event source for a DynamoDB table that already exists in an AWS account.

YAML

```
Events:
  DDBEvent:
    Type: DynamoDB
    Properties:
      Stream: arn:aws:dynamodb:us-east-1:123456789012:table/TestTable/
stream/2016-08-11T21:21:33.291
      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

```
Events:
  DDBEvent:
    Type: DynamoDB
```

```
    Properties:
      Stream:
        !GetAtt MyDynamoDBTable.StreamArn    # This must be the name of a DynamoDB table
  declared in the same template file
        StartingPosition: TRIM_HORIZON
        BatchSize: 10
        Enabled: false
```

EventBridgeRule

The object describing an `EventBridgeRule` event source type, which sets your serverless function as the target of an Amazon EventBridge rule. For more information, see What Is Amazon EventBridge? in the *Amazon EventBridge User Guide*.

AWS SAM generates an AWS::Events::Rule resource when this event type is set.

Syntax

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

YAML

```
    DeadLetterConfig: DeadLetterConfig
    EventBusName: String
    Input: String
    InputPath: String
    InputTransformer: InputTransformer
    Pattern: EventPattern
    RetryPolicy: RetryPolicy
    RuleName: String
    State: String
    Target: Target
```

Properties

`DeadLetterConfig`

Configure the Amazon Simple Queue Service (Amazon SQS) queue where EventBridge sends events after a failed target invocation. Invocation can fail, for example, when sending an event to a Lambda function that doesn't exist, or when EventBridge has insufficient permissions to

invoke the Lambda function. For more information, see Event retry policy and using dead-letter queues in the *Amazon EventBridge User Guide*.

> ℹ️ **Note**
>
> The AWS::Serverless::Function resource type has a similar data type, DeadLetterQueue, which handles failures that occur after successful invocation of the target Lambda function. Examples of these types of failures include Lambda throttling, or errors returned by the Lambda target function. For more information about the function DeadLetterQueue property, see Dead-letter queues in the *AWS Lambda Developer Guide*.

Type: DeadLetterConfig

Required: No

AWS CloudFormation compatibility: This property is similar to the DeadLetterConfig property of the AWS::Events::Rule Target data type. The AWS SAM version of this property includes additional subproperties, in case you want AWS SAM to create the dead-letter queue for you.

EventBusName

The event bus to associate with this rule. If you omit this property, AWS SAM uses the default event bus.

Type: String

Required: No

Default: Default event bus

AWS CloudFormation compatibility: This property is passed directly to the EventBusName property of an AWS::Events::Rule resource.

Input

Valid JSON text passed to the target. If you use this property, nothing from the event text itself is passed to the target.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the <u>Input</u> 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 <u>InputPath</u> property of an `AWS::Events::Rule Target` resource.

InputTransformer

Settings to enable you to provide custom input to a target based on certain event data. You can extract one or more key-value pairs from the event and then use that data to send customized input to the target. For more information, see <u>Amazon EventBridge input transformation</u> in the *Amazon EventBridge User Guide*.

Type: <u>InputTransformer</u>

Required: No

AWS CloudFormation compatibility: This property is passed directly to the <u>InputTransformer</u> property of an `AWS::Events::Rule Target` data type.

Pattern

Describes which events are routed to the specified target. For more information, see <u>Amazon EventBridge events</u> and <u>EventBridge event patterns</u> in the *Amazon EventBridge User Guide*.

Type: <u>EventPattern</u>

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the <u>EventPattern</u> property of an `AWS::Events::Rule` resource.

RetryPolicy

A `RetryPolicy` object that includes information about the retry policy settings. For more information, see <u>Event retry policy and using dead-letter queues</u> in the *Amazon EventBridge User Guide*.

Type: [RetryPolicy](#)

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [RetryPolicy](#) property of the `AWS::Events::Rule Target` data type.

RuleName

The name of the rule.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [Name](#) property of an `AWS::Events::Rule` resource.

State

The state of the rule.

Accepted values: `DISABLED | ENABLED`

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [State](#) property of an `AWS::Events::Rule` resource.

Target

The AWS resource that EventBridge invokes when a rule is triggered. You can use this property to specify the logical ID of the target. If this property is not specified, then AWS SAM generates the logical ID of the target.

Type: [Target](#)

Required: No

AWS CloudFormation compatibility: This property is similar to the [Targets](#) property of an `AWS::Events::Rule` resource. The AWS SAM version of this property only allows you to specify the logical ID of a single target.

Examples

EventBridgeRule

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

YAML

```
EBRule:
  Type: EventBridgeRule
  Properties:
    Input: '{"Key": "Value"}'
    Pattern:
      detail:
        state:
          - terminated
    RetryPolicy:
      MaximumRetryAttempts: 5
      MaximumEventAgeInSeconds: 900
    DeadLetterConfig:
      Type: SQS
      QueueLogicalId: EBRuleDLQ
    Target:
      Id: MyTarget
```

DeadLetterConfig

The object used to specify the Amazon Simple Queue Service (Amazon SQS) queue where
EventBridge sends events after a failed target invocation. Invocation can fail, for example, when
sending an event to a Lambda function that doesn't exist, or insufficient permissions to invoke the
Lambda function. For more information, see Event retry policy and using dead-letter queues in the
Amazon EventBridge User Guide.

> ⓘ **Note**
>
> The AWS::Serverless::Function resource type has a similar data type, `DeadLetterQueue`
> which handles failures that occur after successful invocation of the target Lambda function.
> Examples of this type of failure include Lambda throttling, or errors returned by the
> Lambda target function. For more information about the function `DeadLetterQueue`
> property, see Dead-letter queues in the *AWS Lambda Developer Guide*.

Syntax

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

YAML

```
Arn: String
QueueLogicalId: String
Type: String
```

Properties

Arn

The Amazon Resource Name (ARN) of the Amazon SQS queue specified as the target for the dead-letter queue.

> ⓘ **Note**
>
> Specify either the `Type` property or `Arn` property, but not both.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the Arn property of the AWS::Events::Rule DeadLetterConfig data type.

QueueLogicalId

The custom name of the dead letter queue that AWS SAM creates if `Type` is specified.

> ⓘ **Note**
>
> If the `Type` property is not set, this property is ignored.

Type: String

Required: No

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

Type

The type of the queue. When this property is set, AWS SAM automatically creates a dead-letter queue and attaches necessary [resource-based policy](#) to grant permission to rule resource to send events to the queue.

> ⓘ **Note**
>
> Specify either the `Type` property or `Arn` property, but not both.

Valid values: SQS

Type: String

Required: No

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

Examples

DeadLetterConfig

DeadLetterConfig

YAML

```
DeadLetterConfig:
  Type: SQS
  QueueLogicalId: MyDLQ
```

Target

Configures the AWS resource that EventBridge invokes when a rule is triggered.

Syntax

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

YAML

```
Id: String
```

Properties

Id

The logical ID of the target.

The value of Id can include alphanumeric characters, periods (.), hyphens (-), and underscores (_).

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the Id property of the AWS::Events::Rule Target data type.

Examples

Target

YAML

```
EBRule:
  Type: EventBridgeRule
  Properties:
    Target:
      Id: MyTarget
```

HttpApi

The object describing an event source with type HttpApi.

If an OpenApi definition for the specified path and method exists on the API, SAM will add the Lambda integration and security section (if applicable) for you.

If no OpenApi definition for the specified path and method exists on the API, SAM will create this definition for you.

Syntax

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

YAML

```
ApiId: String
Auth: HttpApiFunctionAuth
Method: String
Path: String
PayloadFormatVersion: String
RouteSettings: RouteSettings
TimeoutInMillis: Integer
```

Properties

`ApiId`

Identifier of an AWS::Serverless::HttpApi resource defined in this template.

If not defined, a default AWS::Serverless::HttpApi 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 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](#)

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:
      Path: /authenticated
      Method: GET
      Auth:
        Authorizer: OpenIdAuth
        AuthorizationScopes:
          - scope1
```

```
        - scope2
```

HttpApiFunctionAuth

Configures authorization at the event level.

Configure Auth for a specific API + Path + Method

Syntax

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

YAML

```
AuthorizationScopes: List
Authorizer: String
```

Properties

AuthorizationScopes

The authorization scopes to apply to this API, path, and method.

Scopes listed here will override any scopes applied by the `DefaultAuthorizer` if one exists.

Type: List

Required: No

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

Authorizer

The `Authorizer` for a specific Function. To use IAM authorization, specify `AWS_IAM` and specify `true` for `EnableIamAuthorizer` in the `Globals` section of your template.

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

Specifing Authorization at Function level

YAML

```
Auth:
  Authorizer: OpenIdAuth
  AuthorizationScopes:
    - scope1
    - scope2
```

IAM authorization

Specifies IAM authorization at the event level. To use `AWS_IAM` authorization at the event level, you must also specify `true` for `EnableIamAuthorizer` in the `Globals` section of your template. For more information, see Globals section of the AWS SAM template.

YAML

```
Globals:
  HttpApi:
    Auth:
      EnableIamAuthorizer: true

Resources:
  HttpApiFunctionWithIamAuth:
    Type: AWS::Serverless::Function
    Properties:
      Events:
        ApiEvent:
          Type: HttpApi
          Properties:
            Path: /iam-auth
            Method: GET
            Auth:
              Authorizer: AWS_IAM
```

```
      Handler: index.handler
      InlineCode: |
        def handler(event, context):
          return {'body': 'HttpApiFunctionWithIamAuth', 'statusCode': 200}
      Runtime: python3.9
```

IoTRule

The object describing an `IoTRule` event source type.

Creates an [AWS::IoT::TopicRule](#) resource to declare an AWS IoT rule. For more information see [AWS CloudFormation documentation](#)

Syntax

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

YAML

```
  AwsIotSqlVersion: String
  Sql: String
```

Properties

`AwsIotSqlVersion`

The version of the SQL rules engine to use when evaluating the rule.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [AwsIotSqlVersion](#) property of an `AWS::IoT::TopicRule TopicRulePayload` resource.

`Sql`

The SQL statement used to query the topic. For more information, see [AWS IoT SQL Reference](#) 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

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

Kinesis

The object describing a `Kinesis` event source type. For more information, see [Using AWS Lambda with Amazon Kinesis](#) in the *AWS Lambda Developer Guide*.

AWS SAM generates an [AWS::Lambda::EventSourceMapping](#) resource when this event type is set.

Syntax

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

YAML

```
  BatchSize: Integer
  BisectBatchOnFunctionError: Boolean
  DestinationConfig: DestinationConfig
  Enabled: Boolean
  FilterCriteria: FilterCriteria
  FunctionResponseTypes: List
  MaximumBatchingWindowInSeconds: Integer
  MaximumRecordAgeInSeconds: Integer
  MaximumRetryAttempts: Integer
  ParallelizationFactor: Integer
  StartingPosition: String
  StartingPositionTimestamp: Double
```

```
    Stream: String
    TumblingWindowInSeconds: Integer
```

Properties

BatchSize

The maximum number of items to retrieve in a single batch.

Type: Integer

Required: No

Default: 100

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

Minimum: 1

Maximum: 10000

BisectBatchOnFunctionError

If the function returns an error, split the batch in two and retry.

Type: Boolean

Required: No

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

DestinationConfig

An Amazon Simple Queue Service (Amazon SQS) queue or Amazon Simple Notification Service
(Amazon SNS) topic destination for discarded records.

Type: DestinationConfig

Required: No

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

Enabled

Disables the event source mapping to pause polling and invocation.

Type: Boolean

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [Enabled](#) property of an `AWS::Lambda::EventSourceMapping` resource.

FilterCriteria

A object that defines the criteria to determine whether Lambda should process an event. For more information, see [AWS Lambda event filtering](#) in the *AWS Lambda Developer Guide*.

Type: [FilterCriteria](#)

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [FilterCriteria](#) property of an `AWS::Lambda::EventSourceMapping` resource.

FunctionResponseTypes

A list of the response types currently applied to the event source mapping. For more information, see [Reporting batch item failures](#) in the *AWS Lambda Developer Guide*.

Valid values: `ReportBatchItemFailures`

Type: List

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [FunctionResponseTypes](#) property of an `AWS::Lambda::EventSourceMapping` resource.

MaximumBatchingWindowInSeconds

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

Type: Integer

Required: No

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

MaximumRecordAgeInSeconds

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

Type: Integer

Required: No

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

MaximumRetryAttempts

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

Type: Integer

Required: No

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

ParallelizationFactor

The number of batches to process from each shard concurrently.

Type: Integer

Required: No

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

StartingPosition

The position in a stream from which to start reading.

- AT_TIMESTAMP – Specify a time from which to start reading records.

- LATEST – Read only new records.

- TRIM_HORIZON – Process all available records.

Valid values: AT_TIMESTAMP | LATEST | TRIM_HORIZON

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the [StartingPosition](#) property of an AWS::Lambda::EventSourceMapping resource.

StartingPositionTimestamp

The time from which to start reading, in Unix time seconds. Define StartingPositionTimestamp when StartingPosition is specified as AT_TIMESTAMP.

Type: Double

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [StartingPositionTimestamp](#) property of an AWS::Lambda::EventSourceMapping resource.

Stream

The Amazon Resource Name (ARN) of the data stream or a stream consumer.

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the [EventSourceArn](#) property of an AWS::Lambda::EventSourceMapping resource.

TumblingWindowInSeconds

The duration, in seconds, of a processing window. The valid range is 1 to 900 (15 minutes).

For more information, see [Tumbling windows](#) in the *AWS Lambda Developer Guide*.

Type: Integer

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [TumblingWindowInSeconds](#) property of an AWS::Lambda::EventSourceMapping resource.

Examples

Kinesis event source

The following is an example of a Kinesis event source.

YAML

```
Events:
  KinesisEvent:
    Type: Kinesis
    Properties:
      Stream: arn:aws:kinesis:us-east-1:123456789012:stream/my-stream
      StartingPosition: TRIM_HORIZON
      BatchSize: 10
      Enabled: false
      FilterCriteria:
        Filters:
          - Pattern: '{"key": ["val1", "val2"]}'
```

MQ

The object describing an MQ event source type. For more information, see Using Lambda with Amazon MQ in the *AWS Lambda Developer Guide*.

AWS Serverless Application Model (AWS SAM) generates an AWS::Lambda::EventSourceMapping resource when this event type is set.

> ⓘ **Note**
>
> To have an Amazon MQ queue in a virtual private cloud (VPC) that connects to a Lambda function in a public network, your function's execution role must include the following permissions:
>
> - `ec2:CreateNetworkInterface`
> - `ec2:DeleteNetworkInterface`
> - `ec2:DescribeNetworkInterfaces`
> - `ec2:DescribeSecurityGroups`
> - `ec2:DescribeSubnets`
> - `ec2:DescribeVpcs`

> For more information, see Execution role permissions in the *AWS Lambda Developer Guide.*

Syntax

To declare this entity in your AWS SAM template, use the following syntax.

YAML

```
BatchSize: Integer
Broker: String
DynamicPolicyName: Boolean
Enabled: Boolean
FilterCriteria: FilterCriteria
MaximumBatchingWindowInSeconds: Integer
Queues: List
SecretsManagerKmsKeyId: String
SourceAccessConfigurations: List
```

Properties

BatchSize

The maximum number of items to retrieve in a single batch.

Type: Integer

Required: No

Default: 100

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

Minimum: 1

Maximum: 10000

Broker

The Amazon Resource Name (ARN) of the Amazon MQ broker.

Type: String

Required: Yes

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

DynamicPolicyName

By default, the AWS Identity and Access Management (IAM) policy name is `SamAutoGeneratedAMQPolicy` for backward compatibility. Specify `true` to use an auto-generated name for your IAM policy. This name will include the Amazon MQ event source logical ID.

> **ⓘ Note**
>
> When using more than one Amazon MQ event source, specify `true` to avoid duplicate IAM policy names.

Type: Boolean

Required: No

Default: `false`

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

Enabled

If `true`, the event source mapping is active. To pause polling and invocation, set to `false`.

Type: Boolean

Required: No

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

FilterCriteria

A object that defines the criteria that determines whether Lambda should process an event. For more information, see AWS Lambda event filtering in the *AWS Lambda Developer Guide*.

Type: FilterCriteria

Required: No

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

MaximumBatchingWindowInSeconds

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

Type: Integer

Required: No

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

Queues

The name of the Amazon MQ broker destination queue to consume.

Type: List

Required: Yes

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

SecretsManagerKmsKeyId

The AWS Key Management Service (AWS KMS) key ID of a customer managed key from AWS
Secrets Manager. Required when you use a customer managed key from Secrets Manager with a
Lambda execution role that doesn't included the kms:Decrypt permission.

The value of this property is a UUID. For example: 1abc23d4-567f-8ab9-
cde0-1fab234c5d67.

Type: String

Required: Conditional

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

SourceAccessConfigurations

An array of the authentication protocol or vitual host. Specify this using the [SourceAccessConfigurations](#) data type.

For the MQ event source type, the only valid configuration types are BASIC_AUTH and VIRTUAL_HOST.

- **BASIC_AUTH** – The Secrets Manager secret that stores your broker credentials. For this type, the credential must be in the following format: `{"username": "your-username", "password": "your-password"}`. Only one object of type BASIC_AUTH is allowed.

- **VIRTUAL_HOST** – The name of the virtual host in your RabbitMQ broker. Lambda will use this Rabbit MQ's host as the event source. Only one object of type VIRTUAL_HOST is allowed.

Type: List

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the [SourceAccessConfigurations](#) property of an `AWS::Lambda::EventSourceMapping` resource.

Examples

Amazon MQ event source

The following is an example of an MQ event source type for an Amazon MQ broker.

YAML

```
Events:
  MQEvent:
    Type: MQ
    Properties:
      Broker: arn:aws:mq:us-
east-2:123456789012:broker:MyBroker:b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9
      Queues: List of queues
      SourceAccessConfigurations:
        - Type: BASIC_AUTH
          URI: arn:aws:secretsmanager:us-east-1:01234567890:secret:MyBrokerSecretName
      BatchSize: 200
```

```
      Enabled: true
```

MSK

The object describing an MSK event source type. For more information, see Using AWS Lambda with Amazon MSK in the *AWS Lambda Developer Guide*.

AWS Serverless Application Model (AWS SAM) generates an AWS::Lambda::EventSourceMapping resource when this event type is set.

Syntax

To declare this entity in your AWS SAM template, use the following syntax.

YAML

```
ConsumerGroupId: String
DestinationConfig: DestinationConfig
FilterCriteria: FilterCriteria
MaximumBatchingWindowInSeconds: Integer
SourceAccessConfigurations: SourceAccessConfigurations
StartingPosition: String
StartingPositionTimestamp: Double
Stream: String
Topics: List
```

Properties

ConsumerGroupId

A string that configures how events will be read from Kafka topics.

Type: String

Required: No

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

DestinationConfig

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

Use this property to specify the destination of failed invocations from the Amazon MSK event source.

Type: DestinationConfig

Required: No

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

`FilterCriteria`

A object that defines the criteria that determines whether Lambda should process an event. For more information, see AWS Lambda event filtering in the *AWS Lambda Developer Guide*.

Type: FilterCriteria

Required: No

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

`MaximumBatchingWindowInSeconds`

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

Type: Integer

Required: No

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

`SourceAccessConfigurations`

An array of the authentication protocol, VPC components, or virtual host to secure and define your event source.

Valid values: `CLIENT_CERTIFICATE_TLS_AUTH`

Type: List of SourceAccessConfiguration

Required: No

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

StartingPosition

The position in a stream from which to start reading.

- AT_TIMESTAMP – Specify a time from which to start reading records.

- LATEST – Read only new records.

- TRIM_HORIZON – Process all available records.

Valid values: AT_TIMESTAMP | LATEST | TRIM_HORIZON

Type: String

Required: No

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

StartingPositionTimestamp

The time from which to start reading, in Unix time seconds. Define
StartingPositionTimestamp when StartingPosition is specified as AT_TIMESTAMP.

Type: Double

Required: No

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

Stream

The Amazon Resource Name (ARN) of the data stream or a stream consumer.

Type: String

Required: Yes

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

`Topics`

>   The name of the Kafka topic.

>   *Type*: List

>   *Required*: Yes

>   *AWS CloudFormation compatibility*: This property is passed directly to the [Topics](#) property of
>   an `AWS::Lambda::EventSourceMapping` resource.

Examples

Amazon MSK Example for Existing Cluster

The following is an example of an MSK event source type for an Amazon MSK cluster that already
exists in an AWS account.

YAML

```
Events:
  MSKEvent:
    Type: MSK
    Properties:
      StartingPosition: LATEST
      Stream: arn:aws:kafka:us-east-1:012345678012:cluster/exampleClusterName/
abcdefab-1234-abcd-5678-cdef0123ab01-2
      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

```
Events:
  MSKEvent:
    Type: MSK
    Properties:
      StartingPosition: LATEST
```

```
      Stream:
        Ref: MyMskCluster    # This must be the name of an MSK cluster declared in the
  same template file
      Topics:
        - MyTopic
```

S3

The object describing an S3 event source type.

Syntax

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

YAML

```
Bucket: String
Events: String | List
Filter: NotificationFilter
```

Properties

Bucket

S3 bucket name. This bucket must exist in the same template.

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is similar to the BucketName property of an AWS::S3::Bucket resource. This is a required field in SAM. This field only accepts a reference to the S3 bucket created in this template

Events

The Amazon S3 bucket event for which to invoke the Lambda function. See Amazon S3 supported event types for a list of valid values.

Type: String | List

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the [Event](#) property of the `AWS::S3::Bucket LambdaConfiguration` data type.

`Filter`

The filtering rules that determine which Amazon S3 objects invoke the Lambda function. For information about Amazon S3 key name filtering, see [Configuring Amazon S3 Event Notifications](#) in the *Amazon Simple Storage Service User 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

```
Events:
  S3Event:
    Type: S3
    Properties:
      Bucket:
        Ref: ImagesBucket     # This must be the name of an S3 bucket declared in the
 same template file
      Events: s3:ObjectCreated:*
      Filter:
        S3Key:
          Rules:
          - Name: prefix      # or "suffix"
            Value: value       # The value to search for in the S3 object key names
```

Schedule

The object describing a `Schedule` event source type, which sets your serverless function as the target of an Amazon EventBridge rule that triggers on a schedule. For more information, see [What Is Amazon EventBridge?](#) in the *Amazon EventBridge User Guide*.

AWS Serverless Application Model (AWS SAM) generates an AWS::Events::Rule resource when this event type is set.

> ⓘ **Note**
>
> EventBridge now offers a new scheduling capability, Amazon EventBridge Scheduler. Amazon EventBridge Scheduler is a serverless scheduler that allows you to create, run, and manage tasks from one central, managed service. EventBridge Scheduler is highly customizable, and offers improved scalability over EventBridge scheduled rules, with a wider set of target API operations and AWS services.
>
> We recommend that you use EventBridge Scheduler to invoke targets on a schedule. To define this event source type in your AWS SAM templates, see ScheduleV2.

Syntax

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

YAML

```
DeadLetterConfig: DeadLetterConfig
Description: String
Enabled: Boolean
Input: String
Name: String
RetryPolicy: RetryPolicy
Schedule: String
State: String
```

Properties

DeadLetterConfig

Configure the Amazon Simple Queue Service (Amazon SQS) queue where EventBridge sends events after a failed target invocation. Invocation can fail, for example, when sending an event to a Lambda function that doesn't exist, or when EventBridge has insufficient permissions to invoke the Lambda function. For more information, see Event retry policy and using dead-letter queues in the *Amazon EventBridge User Guide*.

> **ⓘ Note**
>
> The [AWS::Serverless::Function](#) resource type has a similar data type,
> `DeadLetterQueue`, which handles failures that occur after successful invocation of the
> target Lambda function. Examples of these types of failures include Lambda throttling,
> or errors returned by the Lambda target function. For more information about the
> function `DeadLetterQueue` property, see [Dead-letter queues](#) in the *AWS Lambda*
> *Developer Guide*.

Type: [DeadLetterConfig](#)

Required: No

AWS CloudFormation compatibility: This property is similar to the [DeadLetterConfig](#) property
of the `AWS::Events::Rule Target` data type. The AWS SAM version of this property includes
additional subproperties, in case you want AWS SAM to create the dead-letter queue for you.

Description

A description of the rule.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [Description](#)
property of an `AWS::Events::Rule` resource.

Enabled

Indicates whether the rule is enabled.

To disable the rule, set this property to `false`.

> **ⓘ Note**
>
> Specify either the `Enabled` or `State` property, but not both.

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 Input property of an
AWS::Events::Rule Target resource.

Name

The name of the rule. If you don't specify a name, AWS CloudFormation generates a unique
physical ID and uses that ID for the rule name.

Type: String

Required: No

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

RetryPolicy

A RetryPolicy object that includes information about the retry policy settings. For more
information, see Event retry policy and using dead-letter queues in the *Amazon EventBridge
User Guide*.

Type: RetryPolicy

Required: No

AWS CloudFormation compatibility: This property is passed directly to the RetryPolicy
property of the AWS::Events::Rule Target data type.

```
Schedule
```

The scheduling expression that determines when and how often the rule runs. For more information, see Schedule Expressions for Rules.

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the `ScheduleExpression` property of an `AWS::Events::Rule` resource.

```
State
```

The state of the rule.

Accepted values: `DISABLED | ENABLED`

> ⓘ **Note**
>
> Specify either the `Enabled` or `State` property, but not both.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the `State` property of an `AWS::Events::Rule` resource.

Examples

CloudWatch Schedule Event

CloudWatch Schedule Event Example

YAML

```
CWSchedule:
  Type: Schedule
  Properties:
```

```
    Schedule: 'rate(1 minute)'
    Name: TestSchedule
    Description: test schedule
    Enabled: false
```

DeadLetterConfig

The object used to specify the Amazon Simple Queue Service (Amazon SQS) queue where EventBridge sends events after a failed target invocation. Invocation can fail, for example, when sending an event to a Lambda function that doesn't exist, or insufficient permissions to invoke the Lambda function. For more information, see Event retry policy and using dead-letter queues in the *Amazon EventBridge User Guide*.

> **ⓘ Note**
>
> The AWS::Serverless::Function resource type has a similar data type, `DeadLetterQueue` which handles failures that occur after successful invocation of the target Lambda function. Examples of this type of failure include Lambda throttling, or errors returned by the Lambda target function. For more information about the function `DeadLetterQueue` property, see Dead-letter queues in the *AWS Lambda Developer Guide*.

Syntax

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

YAML

```
Arn: String
QueueLogicalId: String
Type: String
```

Properties

Arn

The Amazon Resource Name (ARN) of the Amazon SQS queue specified as the target for the dead-letter queue.

> **ⓘ Note**
>
> Specify either the `Type` property or `Arn` property, but not both.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [Arn] property of the `AWS::Events::Rule DeadLetterConfig` data type.

QueueLogicalId

The custom name of the dead letter queue that AWS SAM creates if `Type` is specified.

> **ⓘ Note**
>
> If the `Type` property is not set, this property is ignored.

Type: String

Required: No

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

Type

The type of the queue. When this property is set, AWS SAM automatically creates a dead-letter queue and attaches necessary [resource-based policy] to grant permission to rule resource to send events to the queue.

> **ⓘ Note**
>
> Specify either the `Type` property or `Arn` property, but not both.

Valid values: SQS

Type: String

Required: No

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

Examples

DeadLetterConfig

DeadLetterConfig

YAML

```
DeadLetterConfig:
  Type: SQS
  QueueLogicalId: MyDLQ
```

ScheduleV2

The object describing a `ScheduleV2` event source type, which sets your serverless function as
the target of an Amazon EventBridge Scheduler event that triggers on a schedule. For more
information, see What is Amazon EventBridge Scheduler? in the *EventBridge Scheduler User Guide*.

AWS Serverless Application Model (AWS SAM) generates an AWS::Scheduler::Schedule 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

```
DeadLetterConfig: DeadLetterConfig
Description: String
EndDate: String
FlexibleTimeWindow: FlexibleTimeWindow
GroupName: String
Input: String
KmsKeyArn: String
Name: String
OmitName: Boolean
```

```
PermissionsBoundary: String
RetryPolicy: RetryPolicy
RoleArn: String
ScheduleExpression: String
ScheduleExpressionTimezone: String
StartDate: String
State: String
```

Properties

`DeadLetterConfig`

Configure the Amazon Simple Queue Service (Amazon SQS) queue where EventBridge sends events after a failed target invocation. Invocation can fail, for example, when sending an event to a Lambda function that doesn't exist, or when EventBridge has insufficient permissions to invoke the Lambda function. For more information, see Configuring a dead-letter queue for EventBridge Scheduler in the *EventBridge Scheduler User Guide*.

> **ⓘ Note**
>
> The AWS::Serverless::Function resource type has a similar data type, `DeadLetterQueue`, which handles failures that occur after successful invocation of the target Lambda function. Examples of these types of failures include Lambda throttling, or errors returned by the Lambda target function. For more information about the function `DeadLetterQueue` property, see Dead-letter queues in the *AWS Lambda Developer Guide*.

Type: DeadLetterConfig

Required: No

AWS CloudFormation compatibility: This property is similar to the DeadLetterConfig property of the `AWS::Scheduler::Schedule Target` data type. The AWS SAM version of this property includes additional subproperties, in case you want AWS SAM to create the dead-letter queue for you.

`Description`

A description of the schedule.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the <u>Description</u> property of an `AWS::Scheduler::Schedule` resource.

EndDate

The date, in UTC, before which the schedule can invoke its target. Depending on the schedule's recurrence expression, invocations might stop on, or before, the **EndDate** you specify.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the <u>EndDate</u> property of an `AWS::Scheduler::Schedule` resource.

FlexibleTimeWindow

Allows configuration of a window within which a schedule can be invoked.

Type: <u>FlexibleTimeWindow</u>

Required: No

AWS CloudFormation compatibility: This property is passed directly to the <u>FlexibleTimeWindow</u> property of an `AWS::Scheduler::Schedule` resource.

GroupName

The name of the schedule group to associate with this schedule. If not defined, the default group is used.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the <u>GroupName</u> property of an `AWS::Scheduler::Schedule` 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::Scheduler::Schedule Target` resource.

KmsKeyArn

The ARN for a KMS Key that will be used to encrypt customer data.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the KmsKeyArn property
of an `AWS::Scheduler::Schedule` resource.

Name

The name of the schedule. If you don't specify a name, AWS SAM generates a name in the
format *Function-Logical-IDEvent-Source-Name* and uses that ID for the schedule name.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the Name property of an
`AWS::Scheduler::Schedule` resource.

OmitName

By default, AWS SAM generates and uses a schedule name in the format of *<Function-
logical-ID><event-source-name>*. Set this property to `true` to have AWS
CloudFormation generate a unique physical ID and use that for the schedule name instead.

Type: Boolean

Required: No

Default: `false`

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

PermissionsBoundary

The ARN of the policy used to set the permissions boundary for the role.

> **ⓘ Note**
>
> If `PermissionsBoundary` is defined, AWS SAM will apply the same boundaries to the scheduler schedule's target IAM role.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [PermissionsBoundary](#) property of an `AWS::IAM::Role` resource.

RetryPolicy

A **RetryPolicy** object that includes information about the retry policy settings.

Type: [RetryPolicy](#)

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [RetryPolicy](#) property of the `AWS::Scheduler::Schedule Target` data type.

RoleArn

The ARN of the IAM role that EventBridge Scheduler will use for the target when the schedule is invoked.

Type: [RoleArn](#)

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [RoleArn](#) property of the `AWS::Scheduler::Schedule Target` data type.

ScheduleExpression

The scheduling expression that determines when and how often the scheduler schedule event runs.

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the
[ScheduleExpression](#) property of an `AWS::Scheduler::Schedule` resource.

`ScheduleExpressionTimezone`

The timezone in which the scheduling expression is evaluated.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the
[ScheduleExpressionTimezone](#) property of an `AWS::Scheduler::Schedule` resource.

`StartDate`

The date, in UTC, after which the schedule can begin invoking a target. Depending on the
schedule's recurrence expression, invocations might occur on, or after, the **StartDate** you
specify.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [StartDate](#) property
of an `AWS::Scheduler::Schedule` resource.

`State`

The state of the Scheduler schedule.

Accepted values: `DISABLED | ENABLED`

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [State](#) property of an
`AWS::Scheduler::Schedule` resource.

Examples

Basic example of defining a ScheduleV2 resource

```
Resources:
```

```
  Function:
    Properties:
      ...
      Events:
        ScheduleEvent:
          Type: ScheduleV2
          Properties:
            ScheduleExpression: "rate(1 minute)"
        ComplexScheduleEvent:
          Type: ScheduleV2
          Properties:
            ScheduleExpression: rate(1 minute)
            FlexibleTimeWindow:
              Mode: FLEXIBLE
              MaximumWindowInMinutes: 5
            StartDate: '2022-12-28T12:00:00.000Z'
            EndDate: '2023-01-28T12:00:00.000Z'
            ScheduleExpressionTimezone: UTC
            RetryPolicy:
              MaximumRetryAttempts: 5
              MaximumEventAgeInSeconds: 300
            DeadLetterConfig:
              Type: SQS
```

SelfManagedKafka

The object describing a `SelfManagedKafka` event source type. For more information, see [Using AWS Lambda with with self-managed Apache Kafka](#) in the *AWS Lambda Developer Guide*.

AWS Serverless Application Model (AWS SAM) generates an [AWS::Lambda::EventSourceMapping](#) resource when this event type is set.

Syntax

To declare this entity in your AWS SAM template, use the following syntax.

YAML

```
  BatchSize: Integer
  ConsumerGroupId: String
  DestinationConfig: DestinationConfig
  Enabled: Boolean
  FilterCriteria: FilterCriteria
```

```
    KafkaBootstrapServers: List
    SourceAccessConfigurations: SourceAccessConfigurations
    StartingPosition: String
    StartingPositionTimestamp: Double
    Topics: List
```

Properties

BatchSize

　　The maximum number of records in each batch that Lambda pulls from your stream and sends
　　to your function.

　　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

ConsumerGroupId

　　A string that configures how events will be read from Kafka topics.

　　Type: String

　　Required: No

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

DestinationConfig

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

　　Use this property to specify the destination of failed invocations from the self-managed Kafka
　　event source.

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.

FilterCriteria

A object that defines the criteria to determine whether Lambda should process an event. For
more information, see AWS Lambda event filtering in the *AWS Lambda Developer Guide*.

Type: FilterCriteria

Required: No

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

KafkaBootstrapServers

The list of bootstrap servers for your Kafka brokers. Include the port, for example
`broker.example.com:`*`xxxx`*

Type: List

Required: No

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

SourceAccessConfigurations

An array of the authentication protocol, VPC components, or virtual host to secure and define
your event source.

Valid values: BASIC_AUTH | CLIENT_CERTIFICATE_TLS_AUTH | SASL_SCRAM_256_AUTH | SASL_SCRAM_512_AUTH | SERVER_ROOT_CA_CERTIFICATE

Type: List of SourceAccessConfiguration

Required: Yes

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

StartingPosition

The position in a stream from which to start reading.

- AT_TIMESTAMP – Specify a time from which to start reading records.
- LATEST – Read only new records.
- TRIM_HORIZON – Process all available records.

Valid values: AT_TIMESTAMP | LATEST | TRIM_HORIZON

Type: String

Required: No

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

StartingPositionTimestamp

The time from which to start reading, in Unix time seconds. Define StartingPositionTimestamp when StartingPosition is specified as AT_TIMESTAMP.

Type: Double

Required: No

AWS CloudFormation compatibility: This property is passed directly to the StartingPositionTimestamp 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

Self-managed Kafka event source

The following is an example of a `SelfManagedKafka` event source type.

YAML

```yaml
Events:
  SelfManagedKafkaEvent:
    Type: SelfManagedKafka
    Properties:
      BatchSize: 1000
      Enabled: true
      KafkaBootstrapServers:
        - abc.xyz.com:xxxx
      SourceAccessConfigurations:
        -  Type: BASIC_AUTH
           URI: arn:aws:secretsmanager:us-west-2:123456789012:secret:my-path/my-secret-
name-1a2b3c
      Topics:
        - MyKafkaTopic
```

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

```yaml
  FilterPolicy: SnsFilterPolicy
```

```
    FilterPolicyScope: String
    RedrivePolicy: Json
    Region: String
    SqsSubscription: Boolean | SqsSubscriptionObject
    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.

`FilterPolicyScope`

> This attribute lets you choose the filtering scope by using one of the following string value
> types:
>
> - `MessageAttributes` – The filter is applied on the message attributes.
> - `MessageBody` – The filter is applied on the message body.
>
> *Type*: String
>
> *Required*: No
>
> *Default*: `MessageAttributes`
>
> *AWS CloudFormation compatibility*: This property is passed directly to the
> [`FilterPolicyScope`](#) property of an `AWS::SNS::Subscription` resource.

`RedrivePolicy`

> When specified, sends undeliverable messages to the specified Amazon SQS dead-letter
> queue. Messages that can't be delivered due to client errors (for example, when the subscribed
> endpoint is unreachable) or server errors (for example, when the service that powers the
> subscribed endpoint becomes unavailable) are held in the dead-letter queue for further analysis
> or reprocessing.

For more information about the redrive policy and dead-letter queues, see Amazon SQS dead-letter queues in the *Amazon Simple Queue Service Developer Guide*.

Type: Json

Required: No

AWS CloudFormation compatibility: This property is passed directly to the `RedrivePolicy` 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

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::Subscription` resource.

Examples

SNS Event Source Example

SNS Event Source Example

YAML

```
Events:
  SNSEvent:
    Type: SNS
    Properties:
      Topic: arn:aws:sns:us-east-1:123456789012:my_topic
      SqsSubscription: true
      FilterPolicy:
        store:
          - example_corp
        price_usd:
          - numeric:
              - ">="
              - 100
```

SqsSubscriptionObject

Specify an existing SQS queue option to SNS event

Syntax

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

YAML

```
  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 subscibing to an SNS topic.

YAML

```
QueuePolicyLogicalId: CustomQueuePolicyLogicalId
QueueArn:
  Fn::GetAtt: MyCustomQueue.Arn
QueueUrl:
  Ref: MyCustomQueue
BatchSize: 5
```

SQS

The object describing an SQS event source type. For more information, see [Using AWS Lambda with Amazon SQS](#) in the *AWS Lambda Developer Guide*.

SAM generates [AWS::Lambda::EventSourceMapping](#) resource when this event type is set

Syntax

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

YAML

```
  BatchSize: Integer
```

```
Enabled: Boolean
FilterCriteria: FilterCriteria
FunctionResponseTypes: List
MaximumBatchingWindowInSeconds: Integer
Queue: String
ScalingConfig: ScalingConfig
```

Properties

BatchSize

The maximum number of items to retrieve in a single batch.

Type: Integer

Required: No

Default: 10

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

Minimum: 1

Maximum: 10000

Enabled

Disables the event source mapping to pause polling and invocation.

Type: Boolean

Required: No

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

FilterCriteria

A object that defines the criteria to determine whether Lambda should process an event. For more information, see AWS Lambda event filtering in the *AWS Lambda Developer Guide*.

Type: FilterCriteria

Required: No

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

FunctionResponseTypes

A list of the response types currently applied to the event source mapping. For more information, see Reporting batch item failures in the *AWS Lambda Developer Guide*.

Valid values: ReportBatchItemFailures

Type: List

Required: No

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

MaximumBatchingWindowInSeconds

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

Type: Integer

Required: No

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

Queue

The ARN of the queue.

Type: String

Required: Yes

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

ScalingConfig

Scaling configuration of SQS pollers to control the invoke rate and set maximum concurrent invokes.

Type: [ScalingConfig](#)

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [ScalingConfig](#)
property of an `AWS::Lambda::EventSourceMapping` resource.

Examples

Basic SQS event

```
Events:
  SQSEvent:
    Type: SQS
    Properties:
      Queue: arn:aws:sqs:us-west-2:012345678901:my-queue
      BatchSize: 10
      Enabled: false
      FilterCriteria:
        Filters:
          - Pattern: '{"key": ["val1", "val2"]}'
```

Configure partial batch reporting for your SQS queue

```
Events:
  SQSEvent:
    Type: SQS
    Properties:
      Enabled: true
      FunctionResponseTypes:
        - ReportBatchItemFailures
      Queue: !GetAtt MySqsQueue.Arn
      BatchSize: 10
```

Lambda function with an SQS event that has scaling configured

```
MyFunction:
  Type: AWS::Serverless::Function
  Properties:
    ...
    Events:
```

```
    MySQSEvent:
      Type: SQS
      Properties:
        ...
        ScalingConfig:
          MaximumConcurrency: 10
```

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

CodeUri: Function Code example

YAML

```
CodeUri:
  Bucket: mybucket-name
  Key: mykey-name
  Version: 121212
```

FunctionUrlConfig

Creates an AWS Lambda function URL with the specified configuration parameters. A Lambda function URL is an HTTPS endpoint that you can use to invoke your function.

By default, the function URL that you create uses the $LATEST version of your Lambda function. If you specify an `AutoPublishAlias` for your Lambda function, the endpoint connects to the specified function alias.

For more information, see [Lambda function URLs](#) in the *AWS Lambda Developer Guide*.

Syntax

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

YAML

```
AuthType: String
Cors: Cors
InvokeMode: String
```

Properties

AuthType

The type of authorization for your function URL. To use AWS Identity and Access Management (IAM) to authorize requests, set to AWS_IAM. For open access, set to NONE.

Type: String

Required: Yes

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

Cors

The cross-origin resource sharing (CORS) settings for your function URL.

Type: Cors

Required: No

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

InvokeMode

The mode that your function URL will be invoked. To have your function return the response after invocation completes, set to BUFFERED. To have your function stream the response, set to RESPONSE_STREAM. The default value is BUFFERED.

Valid values: BUFFERED or RESPONSE_STREAM

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [InvokeMode](#) property of an `AWS::Lambda::Url` resource.

Examples

Function URL

The following example creates a Lambda function with a function URL. The function URL uses IAM authorization.

YAML

```
HelloWorldFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: hello_world/
    Handler: index.handler
    Runtime: nodejs20.x
    FunctionUrlConfig:
      AuthType: AWS_IAM
      InvokeMode: RESPONSE_STREAM

Outputs:
  MyFunctionUrlEndpoint:
      Description: "My Lambda Function URL Endpoint"
      Value:
        Fn::GetAtt: HelloWorldFunctionUrl.FunctionUrl
```

AWS::Serverless::GraphQLApi

Use the AWS Serverless Application Model (AWS SAM) `AWS::Serverless::GraphQLApi` resource type to create and configure an AWS AppSync GraphQL API for your serverless application.

To learn more about AWS AppSync, see [What is AWS AppSync?](#) in the *AWS AppSync Developer Guide*.

Syntax

YAML

```
LogicalId:
```

```
Type: AWS::Serverless::GraphQLApi
Properties:
  ApiKeys: ApiKeys
  Auth: Auth
  Cache: AWS::AppSync::ApiCache
  DataSources: DataSource
  DomainName: AWS::AppSync::DomainName
  Functions: Function
  Logging: LogConfig
  Name: String
  Resolvers: Resolver
  SchemaInline: String
  SchemaUri: String
  Tags:
  - Tag
  XrayEnabled: Boolean
```

Properties

ApiKeys

Create a unique key that can be used to perform GraphQL operations requiring an API key.

Type: ApiKeys

Required: No

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

Auth

Configure authentication for your GraphQL API.

Type: Auth

Required: Yes

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

Cache

The input of a `CreateApiCache` operation.

Type: [AWS::AppSync::ApiCache](#)

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [AWS::AppSync::ApiCache](#) resource.

DataSources

Create data sources for functions in AWS AppSync to connect to. AWS SAM supports Amazon DynamoDB and AWS Lambda data sources.

Type: [DataSource](#)

Required: Yes

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

DomainName

Custom domain name for your GraphQL API.

Type: [AWS::AppSync::DomainName](#)

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [AWS::AppSync::DomainName](#) resource. AWS SAM automatically generates the [AWS::AppSync::DomainNameApiAssociation](#) resource.

Functions

Configure functions in GraphQL APIs to perform certain operations.

Type: [Function](#)

Required: Yes

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

Logging

Configures Amazon CloudWatch logging for your GraphQL API.

If you don't specify this property, AWS SAM will generate `CloudWatchLogsRoleArn` and set the following values:

- `ExcludeVerboseContent: true`
- `FieldLogLevel: ALL`

To opt out of logging, specify the following:

```
Logging: false
```

Type: [LogConfig](#)

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [LogConfig](#) property of an `AWS::AppSync::GraphQLApi` resource.

LogicalId

The unique name of your GraphQL API.

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the [Name](#) property of an `AWS::AppSync::GraphQLApi` resource.

Name

The name of your GraphQL API. Specify this property to override the `LogicalId` value.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [Name](#) property of an `AWS::AppSync::GraphQLApi` resource.

Resolvers

Configure resolvers for the fields of your GraphQL API. AWS SAM supports [JavaScript pipeline resolvers](#).

  Type: [Resolver](#)

  Required: Yes

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

SchemaInline

  The text representation of a GraphQL schema in SDL format.

  Type: String

  Required: Conditional. You must specify SchemaInline or SchemaUri.

  AWS CloudFormation compatibility: This property is passed directly to the [Definition](#)
  property of an AWS::AppSync::GraphQLSchema resource.

SchemaUri

  The schema's Amazon Simple Storage Service (Amazon S3) bucket URI or path to a local folder.

  If you specify a path to a local folder, AWS CloudFormation requires that the file is first
  uploaded to Amazon S3 before deployment. You can use the AWS SAM CLI to facilitate this
  process. For more information, see [Using the AWS SAM CLI to upload local files at deployment](#).

  Type: String

  Required: Conditional. You must specify SchemaInline or SchemaUri.

  AWS CloudFormation compatibility: This property is passed directly to the
  [DefinitionS3Location](#) property of an AWS::AppSync::GraphQLSchema resource.

Tags

  Tags (key-value pairs) for this GraphQL API. Use tags to identify and categorize resources.

  Type: List of [Tag](#)

  Required: No

  AWS CloudFormation compatibility: This property is passed directly to the [Tag](#) property of an
  AWS::AppSync::GraphQLApi resource.

`XrayEnabled`

Indicate whether to use [AWS X-Ray tracing](#) for this resource.

Type: Boolean

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [XrayEnabled](#) property of an `AWS::AppSync::GraphQLApi` resource.

Examples

GraphQL API with DynamoDB data source

In this example, we create a GraphQL API that uses a DynamoDB table as a data source.

schema.graphql

```
schema {
  query: Query
  mutation: Mutation
}

type Query {
  getPost(id: String!): Post
}

type Mutation {
  addPost(author: String!, title: String!, content: String!): Post!
}

type Post {
  id: String!
  author: String
  title: String
  content: String
  ups: Int!
  downs: Int!
  version: Int!
}
```

template.yaml

```yaml
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  DynamoDBPostsTable:
    Type: AWS::Serverless::SimpleTable

  MyGraphQLAPI:
    Type: AWS::Serverless::GraphQLApi
    Properties:
      SchemaUri: ./sam_graphql_api/schema.graphql
      Auth:
        Type: AWS_IAM
      DataSources:
        DynamoDb:
          PostsDataSource:
            TableName: !Ref DynamoDBPostsTable
            TableArn: !GetAtt DynamoDBPostsTable.Arn
      Functions:
        preprocessPostItem:
          Runtime:
            Name: APPSYNC_JS
            Version: 1.0.0
          DataSource: NONE
          CodeUri: ./sam_graphql_api/preprocessPostItem.js
        createPostItem:
          Runtime:
            Name: APPSYNC_JS
            Version: "1.0.0"
          DataSource: PostsDataSource
          CodeUri: ./sam_graphql_api/createPostItem.js
        getPostFromTable:
          Runtime:
            Name: APPSYNC_JS
            Version: "1.0.0"
          DataSource: PostsDataSource
          CodeUri: ./sam_graphql_api/getPostFromTable.js
      Resolvers:
        Mutation:
          addPost:
            Runtime:
              Name: APPSYNC_JS
```

```
            Version: "1.0.0"
        Pipeline:
        - preprocessPostItem
        - createPostItem
    Query:
      getPost:
        CodeUri: ./sam_graphql_api/getPost.js
        Runtime:
          Name: APPSYNC_JS
          Version: "1.0.0"
        Pipeline:
        - getPostFromTable
```

createPostItem.js

```javascript
import { util } from "@aws-appsync/utils";

export function request(ctx) {
  const { key, values } = ctx.prev.result;
  return {
    operation: "PutItem",
    key: util.dynamodb.toMapValues(key),
    attributeValues: util.dynamodb.toMapValues(values),
  };
}

export function response(ctx) {
  return ctx.result;
}
```

getPostFromTable.js

```javascript
import { util } from "@aws-appsync/utils";

export function request(ctx) {
  return dynamoDBGetItemRequest({ id: ctx.args.id });
}

export function response(ctx) {
  return ctx.result;
}

/**
```

```
  * A helper function to get a DynamoDB item
  */
 function dynamoDBGetItemRequest(key) {
   return {
     operation: "GetItem",
     key: util.dynamodb.toMapValues(key),
   };
 }
```

preprocessPostItem.js

```
import { util } from "@aws-appsync/utils";

export function request(ctx) {
  const id = util.autoId();
  const { ...values } = ctx.args;
  values.ups = 1;
  values.downs = 0;
  values.version = 1;
  return { payload: { key: { id }, values: values } };
}


export function response(ctx) {
  return ctx.result;
}
```

Here is our resolver code:

getPost.js

```
export function request(ctx) {
  return {};
}

export function response(ctx) {
  return ctx.prev.result;
}
```

GraphQL API with a Lambda function as a data source

In this example, we create a GraphQL API that uses a Lambda function as a data source.

template.yaml

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyLambdaFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.handler
      Runtime: nodejs20.x
      CodeUri: ./lambda

  MyGraphQLAPI:
    Type: AWS::Serverless::GraphQLApi
    Properties:
      Name: MyApi
      SchemaUri: ./gql/schema.gql
      Auth:
        Type: API_KEY
      ApiKeys:
        MyApiKey:
          Description: my api key
      DataSources:
        Lambda:
          MyLambdaDataSource:
            FunctionArn: !GetAtt MyLambdaFunction.Arn
      Functions:
        lambdaInvoker:
          Runtime:
            Name: APPSYNC_JS
            Version: 1.0.0
          DataSource: MyLambdaDataSource
          CodeUri: ./gql/invoker.js
      Resolvers:
        Mutation:
          addPost:
            Runtime:
              Name: APPSYNC_JS
              Version: 1.0.0
            Pipeline:
            - lambdaInvoker
        Query:
          getPost:
            Runtime:
```

```
                Name: APPSYNC_JS
                Version: 1.0.0
            Pipeline:
            - lambdaInvoker

Outputs:
  MyGraphQLAPI:
    Description: AppSync API
    Value: !GetAtt MyGraphQLAPI.GraphQLUrl
  MyGraphQLAPIMyApiKey:
    Description: API Key for authentication
    Value: !GetAtt MyGraphQLAPIMyApiKey.ApiKey
```

schema.graphql

```
schema {
  query: Query
  mutation: Mutation
}
type Query {
  getPost(id: ID!): Post
}
type Mutation {
  addPost(id: ID!, author: String!, title: String, content: String): Post!
}
type Post {
  id: ID!
  author: String!
  title: String
  content: String
  ups: Int
  downs: Int
}
```

Here are our functions:

lambda/index.js

```
exports.handler = async (event) => {
  console.log("Received event {}", JSON.stringify(event, 3));

  const posts = {
    1: {
```

```
      id: "1",
      title: "First book",
      author: "Author1",
      content: "Book 1 has this content",
      ups: "100",
      downs: "10",
    },
  };

  console.log("Got an Invoke Request.");
  let result;
  switch (event.field) {
    case "getPost":
      return posts[event.arguments.id];
    case "addPost":
      // return the arguments back
      return event.arguments;
    default:
      throw new Error("Unknown field, unable to resolve " + event.field);
  }
};
```

invoker.js

```
import { util } from "@aws-appsync/utils";

export function request(ctx) {
  const { source, args } = ctx;
  return {
    operation: "Invoke",
    payload: { field: ctx.info.fieldName, arguments: args, source },
  };
}

export function response(ctx) {
  return ctx.result;
}
```

ApiKeys

Create a unique key that can be used to perform GraphQL operations requiring an API key.

Syntax

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

YAML

```
LogicalId:
  ApiKeyId: String
  Description: String
  ExpiresOn: Double
```

Properties

ApiKeyId

The unique name of your API key. Specify to override the `LogicalId` value.

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the [ApiKeyId](#) property of an `AWS::AppSync::ApiKey` resource.

Description

Description of your API key.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [Description](#) property of an `AWS::AppSync::ApiKey` resource.

ExpiresOn

The time after which the API key expires. The date is represented as seconds since the epoch, rounded down to the nearest hour.

Type: Double

Required: No

AWS CloudFormation compatibility: This property is passed directly to the <u>Expires</u> property of an `AWS::AppSync::ApiKey` resource.

LogicalId

The unique name of your API key.

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the <u>ApiKeyId</u> property of an `AWS::AppSync::ApiKey` resource.

Auth

Configure authorization for your GraphQL API.

Syntax

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

YAML

```
Additional:
- AuthProvider
LambdaAuthorizer: LambdaAuthorizerConfig
OpenIDConnect: OpenIDConnectConfig
Type: String
UserPool: UserPoolConfig
```

Properties

Additional

A list of additional authorization types for your GraphQL API.

Type: List of AuthProvider

Required: No

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

LambdaAuthorizer

Specify the optional authorization configuration for your Lambda function authorizer. You can
configure this optional property when `Type` is specified as `AWS_LAMBDA`.

Type: LambdaAuthorizerConfig

Required: No

AWS CloudFormation compatibility: This property is passed directly to the
`LambdaAuthorizerConfig` property of an `AWS::AppSync::GraphQLApi` resource.

OpenIDConnect

Specify the optional authorization configuration for your OpenID Connect compliant service.
You can configure this optional property when `Type` is specified as `OPENID_CONNECT`.

Type: OpenIDConnectConfig

Required: No

AWS CloudFormation compatibility: This property is passed directly to the
`OpenIDConnectConfig` property of an `AWS::AppSync::GraphQLApi` resource.

Type

The default authorization type between applications and your AWS AppSync GraphQL API.

For a list and description of allowed values, see Authorization and authentication in the *AWS
AppSync Developer Guide*.

When you specify a Lambda authorizer (`AWS_LAMBDA`), AWS SAM creates an AWS Identity and
Access Management (IAM) policy to provision permissions between your GraphQL API and
Lambda function.

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the
`AuthenticationType` property of an `AWS::AppSync::GraphQLApi` resource.

`UserPool`

> Specify the optional authorization configuration for using Amazon Cognito user pools. You can configure this optional property when `Type` is specified as `AMAZON_COGNITO_USER_POOLS`.
>
> *Type*: UserPoolConfig
>
> *Required*: No
>
> *AWS CloudFormation compatibility*: This property is passed directly to the `UserPoolConfig` property of an `AWS::AppSync::GraphQLApi` resource.

Examples

Configure a default and additional authorization type

In this example, we start by configuring a Lambda authorizer as the default authorization type for our GraphQL API.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyGraphQLAPI:
    Type: AWS::Serverless::GraphQLApi
    Properties:
      Auth:
        Type: AWS_LAMBDA
        LambdaAuthorizer:
          AuthorizerUri: !GetAtt Authorizer1.Arn
          AuthorizerResultTtlInSeconds: 10
          IdentityValidationExpression: hello
```

Next, we configure additional authorization types for our GraphQL API by adding the following to our AWS SAM template:

```
        Additional:
        - Type: AWS_IAM
        - Type: API_KEY
        - Type: OPENID_CONNECT
          OpenIDConnect:
```

```
            AuthTTL: 10
            ClientId: myId
            IatTTL: 10
            Issuer: prod
```

This results in the following AWS SAM template:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyGraphQLAPI:
    Type: AWS::Serverless::GraphQLApi
    Properties:
      Auth:
        Type: AWS_LAMBDA
        LambdaAuthorizer:
          AuthorizerUri: !GetAtt Authorizer1.Arn
          AuthorizerResultTtlInSeconds: 10
          IdentityValidationExpression: hello
        Additional:
        - Type: AWS_IAM
        - Type: API_KEY
        - Type: OPENID_CONNECT
          OpenIDConnect:
            AuthTTL: 10
            ClientId: myId
            IatTTL: 10
            Issuer: prod
```

AuthProvider

Optional authorization configuration for your additional GraphQL API authorization types.

Syntax

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

YAML

```
LambdaAuthorizer: LambdaAuthorizerConfig
OpenIDConnect: OpenIDConnectConfig
```

```
Type: String
UserPool: UserPoolConfig
```

Properties

LambdaAuthorizer

Specify the optional authorization configuration for your AWS Lambda function authorizer. You can configure this optional property when Type is specified as AWS_LAMBDA.

Type: LambdaAuthorizerConfig

Required: No

AWS CloudFormation compatibility: This property is passed directly to the LambdaAuthorizerConfig property of an AWS::AppSync::GraphQLApi AdditionalAuthenticationProvider object.

OpenIDConnect

Specify the optional authorization configuration for your OpenID Connect compliant service. You can configure this optional property when Type is specified as OPENID_CONNECT.

Type: OpenIDConnectConfig

Required: No

AWS CloudFormation compatibility: This property is passed directly to the OpenIDConnectConfig property of an AWS::AppSync::GraphQLApi AdditionalAuthenticationProvider object.

Type

The default authorization type between applications and your AWS AppSync GraphQL API.

For a list and description of allowed values, see Authorization and authentication in the *AWS AppSync Developer Guide*.

When you specify a Lambda authorizer (AWS_LAMBDA), AWS SAM creates an AWS Identity and Access Management (IAM) policy to provision permissions between your GraphQL API and Lambda function.

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the
<u>AuthenticationType</u> property of an AWS::AppSync::GraphQLApi
<u>AdditionalAuthenticationProvider</u> object.

UserPool

Specify the optional authorization configuration for using Amazon Cognito user pools. You can
configure this optional property when Type is specified as AMAZON_COGNITO_USER_POOLS.

Type: <u>UserPoolConfig</u>

Required: No

AWS CloudFormation compatibility: This property is passed directly to the <u>UserPoolConfig</u>
property of an AWS::AppSync::GraphQLApi <u>AdditionalAuthenticationProvider</u>
object.

DataSource

Configure a data source that your GraphQL API resolver can connect to. You can use AWS
Serverless Application Model (AWS SAM) templates to configure connections to the following data
sources:

- Amazon DynamoDB

- AWS Lambda

To learn more about data sources, see <u>Attaching a data source</u> in the *AWS AppSync Developer Guide*.

Syntax

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

YAML

```
DynamoDb: DynamoDb
Lambda: Lambda
```

Properties

DynamoDb

  Configure a DynamoDB table as a data source for your GraphQL API resolver.

  Type: DynamoDb

  Required: No

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

Lambda

  Configure a Lambda function as a data source for your GraphQL API resolver.

  Type: Lambda

  Required: No

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

DynamoDb

Configure an Amazon DynamoDB table as a data source for your GraphQL API resolver.

Syntax

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

YAML

```
LogicalId:
  DeltaSync: DeltaSyncConfig
  Description: String
  Name: String
  Permissions: List
  Region: String
  ServiceRoleArn: String
  TableArn: String
```

```
    TableName: String
    UseCallerCredentials: Boolean
    Versioned: Boolean
```

Properties

DeltaSync

> Describes a Delta Sync configuration.
>
> *Type*: DeltaSyncConfig
>
> *Required*: No
>
> *AWS CloudFormation compatibility*: This property is passed directly to the DeltaSyncConfig
> property of an AWS::AppSync::DataSource DynamoDBConfig object.

Description

> The description of your data source.
>
> *Type*: String
>
> *Required*: No
>
> *AWS CloudFormation compatibility*: This property is passed directly to the Description
> property of an AWS::AppSync::DataSource resource.

LogicalId

> The unique name of your data source.
>
> *Type*: String
>
> *Required*: Yes
>
> *AWS CloudFormation compatibility*: This property is passed directly to the Name property of an
> AWS::AppSync::DataSource resource.

Name

> The name of your data source. Specify this property to override the LogicalId value.
>
> *Type*: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [Name](#) property of an
`AWS::AppSync::DataSource` resource.

`Permissions`

Provision permissions to your data source using [AWS SAM connectors](#). You can provide any of
the following values in a list:

- `Read` – Allow your resolver to read your data source.

- `Write` – Allow your resolver to write to your data source.

AWS SAM uses an `AWS::Serverless::Connector` resource which is transformed at
deployment to provision your permissions. To learn about generated resources, see [AWS
CloudFormation resources generated when you specify AWS::Serverless::Connector](#).

> **ⓘ Note**
>
> You can specify `Permissions` or `ServiceRoleArn`, but not both. If neither are
> specified, AWS SAM will generate default values of `Read` and `Write`. To revoke access
> to your data source, remove the DynamoDB object from your AWS SAM template.

Type: List

Required: No

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have
an AWS CloudFormation equivalent. It is similar to the [Permissions](#) property of an
`AWS::Serverless::Connector` resource.

`Region`

The AWS Region of your DynamoDB table. If you don't specify it, AWS SAM uses [AWS::Region](#).

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [AwsRegion](#) property
of an `AWS::AppSync::DataSource DynamoDBConfig` object.

```
ServiceRoleArn
```

The AWS Identity and Access Management (IAM) service role ARN for the data source. The system assumes this role when accessing the data source.

You can specify `Permissions` or `ServiceRoleArn`, but not both.

Type: String

Required: No. If not specified, AWS SAM applies the default value for `Permissions`.

AWS CloudFormation compatibility: This property is passed directly to the [ServiceRoleArn](#) property of an `AWS::AppSync::DataSource` resource.

```
TableArn
```

The ARN for the DynamoDB table.

Type: String

Required: Conditional. If you don't specify `ServiceRoleArn`, `TableArn` is required.

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

```
TableName
```

The table name.

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the [TableName](#) property of an `AWS::AppSync::DataSource DynamoDBConfig` object.

```
UseCallerCredentials
```

Set to `true` to use IAM with this data source.

Type: Boolean

Required: No

AWS CloudFormation compatibility: This property is passed directly to the
[UseCallerCredentials](#) property of an `AWS::AppSync::DataSource DynamoDBConfig`
object.

`Versioned`

Set to `true` to use [Conflict Detection, Conflict Resolution, and Sync](#) with this data source.

Type: Boolean

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [Versioned](#) property
of an `AWS::AppSync::DataSource DynamoDBConfig` object.

Lambda

Configure an AWS Lambda function as a data source for your GraphQL API resolver.

Syntax

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

YAML

```
LogicalId:
  Description: String
  FunctionArn: String
  Name: String
  ServiceRoleArn: String
```

Properties

`Description`

The description of your data source.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the <u>Description</u> property of an `AWS::AppSync::DataSource` resource.

FunctionArn

The ARN for the Lambda function.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the <u>LambdaFunctionArn</u> property of an `AWS::AppSync::DataSource LambdaConfig` object.

LogicalId

The unique name of your data source.

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the <u>Name</u> property of an `AWS::AppSync::DataSource` resource.

Name

The name of your data source. Specify this property to override the `LogicalId` value.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the <u>Name</u> property of an `AWS::AppSync::DataSource` resource.

ServiceRoleArn

The AWS Identity and Access Management (IAM) service role ARN for the data source. The system assumes this role when accessing the data source.

> ⓘ **Note**
>
> To revoke access to your data source, remove the Lambda object from your AWS SAM template.

Type: String

Required: No. If not specified, AWS SAM will provision `Write` permissions using AWS SAM connectors.

AWS CloudFormation compatibility: This property is passed directly to the `ServiceRoleArn` property of an `AWS::AppSync::DataSource` resource.

Function

Configure functions in GraphQL APIs to perform certain operations.

Syntax

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

YAML

```yaml
LogicalId:
  CodeUri: String
  DataSource: String
  Description: String
  Id: String
  InlineCode: String
  MaxBatchSize: Integer
  Name: String
  Runtime: Runtime
  Sync: SyncConfig
```

Properties

CodeUri

The function code's Amazon Simple Storage Service (Amazon S3) URI or path to local folder.

If you specify a path to a local folder, AWS CloudFormation requires that the file is first uploaded to Amazon S3 before deployment. You can use the AWS SAM CLI to facilitate this process. For more information, see Using the AWS SAM CLI to upload local files at deployment.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [CodeS3Location](#) property of an `AWS::AppSync::FunctionConfiguration` resource.

`DataSource`

The name of the data source that this function will attach to.

- To reference a data source within the `AWS::Serverless::GraphQLApi` resource, specify its logical ID.

- To reference a data source outside of the `AWS::Serverless::GraphQLApi` resource, provide its `Name` attribute using the `Fn::GetAtt` intrinsic function. For example, `!GetAtt MyLambdaDataSource.Name`.

- To reference a data source from a different stack, use [Fn::ImportValue](#).

If a variation of [NONE | None | none] is specified, AWS SAM will generate a None value for the `AWS::AppSync::DataSource` [Type](#) object.

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the [DataSourceName](#) property of an `AWS::AppSync::FunctionConfiguration` resource.

`Description`

The description of your function.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [Description](#) property of an `AWS::AppSync::FunctionConfiguration` resource.

`Id`

The Function ID for a function located outside of the `AWS::Serverless::GraphQLApi` resource.

- To reference a function within the same AWS SAM template, use the `Fn::GetAtt` intrinsic function. For example `Id: !GetAtt createPostItemFunc.FunctionId`.

- To reference a function from a different stack, use [Fn::ImportValue](#).

When using `Id`, all other properties are not allowed. AWS SAM will automatically pass the Function ID of your referenced function.

Type: String

Required: No

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

InlineCode

The function code that contains the request and response functions.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [Code](#) property of an `AWS::AppSync::FunctionConfiguration` resource.

LogicalId

The unique name of your function.

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the [Name](#) property of an `AWS::AppSync::FunctionConfiguration` resource.

MaxBatchSize

The maximum number of resolver request inputs that will be sent to a single AWS Lambda function in a `BatchInvoke` operation.

Type: Integer

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [MaxBatchSize](#) property of an `AWS::AppSync::FunctionConfiguration` resource.

Name

The name of the function. Specify to override the `LogicalId` value.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [Name](#) property of an
`AWS::AppSync::FunctionConfiguration` resource.

Runtime

Describes a runtime used by an AWS AppSync pipeline resolver or AWS AppSync function.
Specifies the name and version of the runtime to use.

Type: [Runtime](#)

Required: Yes

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't
have an AWS CloudFormation equivalent. It is similar to the `Runtime` property of an
`AWS::AppSync::FunctionConfiguration` resource.

Sync

Describes a Sync configuration for a function.

Specifies which Conflict Detection strategy and Resolution strategy to use when the function is
invoked.

Type: [SyncConfig](#)

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [SyncConfig](#)
property of an `AWS::AppSync::FunctionConfiguration` resource.

Runtime

The runtime of your pipeline resolver or function. Specifies the name and version to use.

Syntax

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

YAML

```
Name: String
Version: String
```

Properties

Name

   The name of the runtime to use. Currently, the only allowed value is APPSYNC_JS.

   Type: String

   Required: Yes

   AWS CloudFormation compatibility: This property is passed directly to the [Name](#) property of an `AWS::AppSync::FunctionConfiguration AppSyncRuntime` object.

Version

   The version of the runtime to use. Currently, the only allowed version is 1.0.0.

   Type: String

   Required: Yes

   AWS CloudFormation compatibility: This property is passed directly to the [RuntimeVersion](#) property of an `AWS::AppSync::FunctionConfiguration AppSyncRuntime` object.

Resolver

Configure resolvers for the fields of your GraphQL API. AWS Serverless Application Model (AWS SAM) supports [JavaScript pipeline resolvers](#).

Syntax

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

YAML

```
OperationType:
```

```
    LogicalId:
      Caching: CachingConfig
      CodeUri: String
      FieldName: String
      InlineCode: String
      MaxBatchSize: Integer
      Pipeline: List
      Runtime: Runtime
      Sync: SyncConfig
```

Properties

Caching

The caching configuration for the resolver that has caching activated.

Type: CachingConfig

Required: No

AWS CloudFormation compatibility: This property is passed directly to the CachingConfig property of an AWS::AppSync::Resolver resource.

CodeUri

The resolver function code's Amazon Simple Storage Service (Amazon S3) URI or path to a local folder.

If you specify a path to a local folder, AWS CloudFormation requires that the file is first uploaded to Amazon S3 before deployment. You can use the AWS SAM CLI to facilitate this process. For more information, see Using the AWS SAM CLI to upload local files at deployment.

If neither CodeUri or InlineCode are provided, AWS SAM will generate InlineCode that redirects the request to the first pipeline function and receives the response from the last pipeline function.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the CodeS3Location property of an AWS::AppSync::Resolver resource.

`FieldName`

The name of your resolver. Specify this property to override the `LogicalId` value.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [FieldName](#) property of an `AWS::AppSync::Resolver` resource.

`InlineCode`

The resolver code that contains the request and response functions.

If neither `CodeUri` or `InlineCode` are provided, AWS SAM will generate `InlineCode` that redirects the request to the first pipeline function and receives the response from the last pipeline function.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [Code](#) property of an `AWS::AppSync::Resolver` resource.

`LogicalId`

The unique name for your resolver. In a GraphQL schema, your resolver name should match the field name that its used for. Use that same field name for `LogicalId`.

Type: String

Required: Yes

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

`MaxBatchSize`

The maximum number of resolver request inputs that will be sent to a single AWS Lambda function in a `BatchInvoke` operation.

Type: Integer

Required: No

AWS CloudFormation compatibility: This property is passed directly to the `MaxBatchSize` property of an `AWS::AppSync::Resolver` resource.

OperationType

The GraphQL operation type that is associated with your resolver. For example, `Query`, `Mutation`, or `Subscription`. You can nest multiple resolvers by `LogicalId` within a single `OperationType`.

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the `TypeName` property of an `AWS::AppSync::Resolver` resource.

Pipeline

Functions linked with the pipeline resolver. Specify functions by logical ID in a list.

Type: List

Required: Yes

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent. It is similar to the `PipelineConfig` property of an `AWS::AppSync::Resolver` resource.

Runtime

The runtime of your pipeline resolver or function. Specifies the name and version to use.

Type: Runtime

Required: Yes

AWS CloudFormation compatibility: This property is unique to AWS SAM and doesn't have an AWS CloudFormation equivalent. It is similar to the `Runtime` property of an `AWS::AppSync::Resolver` resource.

Sync

Describes a Sync configuration for a resolver.

Specifies which Conflict Detection strategy and Resolution strategy to use when the resolver is invoked.

Type: SyncConfig

Required: No

AWS CloudFormation compatibility: This property is passed directly to the SyncConfig property of an `AWS::AppSync::Resolver` resource.

Examples

Use the AWS SAM generated resolver function code and save fields as variables

Here is the GraphQL schema for our example:

```
schema {
  query: Query
  mutation: Mutation
}

type Query {
  getPost(id: ID!): Post
}

type Mutation {
  addPost(author: String!, title: String!, content: String!): Post!
}

type Post {
  id: ID!
  author: String
  title: String
  content: String
}
```

Here is a snippet of our AWS SAM template:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31

...
Resources:
```

```
  MyGraphQLApi:
    Type: AWS::Serverless::GraphQLApi
    Properties:
      ...
      Functions:
        preprocessPostItem:
          ...
        createPostItem:
          ...
      Resolvers:
        Mutation:
          addPost:
            Runtime:
              Name: APPSYNC_JS
              Version: 1.0.0
            Pipeline:
            - preprocessPostItem
            - createPostItem
```

In our AWS SAM template, we don't specify `CodeUri` or `InlineCode`. At deployment, AWS SAM automatically generates the following inline code for our resolver:

```
export function request(ctx) {
  return {};
}

export function response(ctx) {
  return ctx.prev.result;
}
```

This default resolver code redirects the request to the first pipeline function and receives the response from the last pipeline function.

In our first pipeline function, we can use the provided `args` field to parse the request object and create our variables. We can then use these variables within our function. Here is an example of our `preprocessPostItem` function:

```
import { util } from "@aws-appsync/utils";

export function request(ctx) {
  const author = ctx.args.author;
  const title = ctx.args.title;
```

```
    const content = ctx.args.content;

    // Use variables to process data

}

export function response(ctx) {
    return ctx.result;
}
```

Runtime

The runtime of your pipeline resolver or function. Specifies the name and version to use.

Syntax

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

YAML

```
Name: String
Version: String
```

Properties

Name

The name of the runtime to use. Currently, the only allowed value is APPSYNC_JS.

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the Name property of an AWS::AppSync::Resolver AppSyncRuntime object.

Version

The version of the runtime to use. Currently, the only allowed version is 1.0.0.

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the RuntimeVersion property of an `AWS::AppSync::Resolver` `AppSyncRuntime` object.

AWS::Serverless::HttpApi

Creates an Amazon API Gateway HTTP API, which enables you to create RESTful APIs with lower latency and lower costs than REST APIs. For more information, see Working with HTTP APIs in the *API Gateway Developer Guide*.

We recommend that you use AWS CloudFormation hooks or IAM policies to verify that API Gateway resources have authorizers attached to them to control access to them.

For more information about using AWS CloudFormation hooks, see Registering hooks in the *AWS CloudFormation CLI user guide* and the apigw-enforce-authorizer GitHub repository.

For more information about using IAM policies, see Require that API routes have authorization in the *API Gateway Developer Guide*.

> **ⓘ Note**
>
> When you deploy to AWS CloudFormation, AWS SAM transforms your AWS SAM resources into AWS CloudFormation resources. For more information, see Generated AWS CloudFormation resources.

Syntax

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

YAML

```
Type: AWS::Serverless::HttpApi
Properties:
  AccessLogSettings: AccessLogSettings
  Auth: HttpApiAuth
  CorsConfiguration: String | HttpApiCorsConfiguration
  DefaultRouteSettings: RouteSettings
  DefinitionBody: JSON
  DefinitionUri: String | HttpApiDefinition
```

```
      Description: String
      DisableExecuteApiEndpoint: Boolean
      Domain: HttpApiDomainConfiguration
      FailOnWarnings: Boolean
      Name: String
      PropagateTags: Boolean
      RouteSettings: RouteSettings
      StageName: String
      StageVariables: Json
      Tags: Map
```

Properties

`AccessLogSettings`

The settings for access logging in a stage.

Type: AccessLogSettings

Required: No

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

`Auth`

Configures authorization for controlling access to your API Gateway HTTP API.

For more information, see Controlling access to HTTP APIs with JWT authorizers in the *API Gateway Developer Guide*.

Type: HttpApiAuth

Required: No

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

`CorsConfiguration`

Manages cross-origin resource sharing (CORS) for all your API Gateway HTTP APIs. Specify the domain to allow as a string, or specify an `HttpApiCorsConfiguration` object. Note that CORS requires AWS SAM to modify your OpenAPI definition, so CORS works only if the `DefinitionBody` property is specified.

For more information, see [Configuring CORS for an HTTP API](#) in the *API Gateway Developer Guide*.

> **ⓘ Note**
>
> If `CorsConfiguration` is set both in an OpenAPI definition and at the property level, then AWS SAM merges both configuration sources with the properties taking precedence. If this property is set to `true`, then all origins are allowed.

Type: String | [HttpApiCorsConfiguration](#)

Required: No

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

DefaultRouteSettings

The default route settings for this HTTP API. These settings apply to all routes unless overridden by the `RouteSettings` property for certain routes.

Type: [RouteSettings](#)

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [`RouteSettings`](#) property of an `AWS::ApiGatewayV2::Stage` resource.

DefinitionBody

The OpenAPI definition that describes your HTTP API. If you don't specify a `DefinitionUri` or a `DefinitionBody`, AWS SAM generates a `DefinitionBody` for you based on your template configuration.

Type: JSON

Required: No

AWS CloudFormation compatibility: This property is similar to the [Body](#) property of an `AWS::ApiGatewayV2::Api` resource. If certain properties are provided, AWS SAM may insert

content into or modify the `DefinitionBody` before it is passed to AWS CloudFormation. Properties include `Auth` and an `EventSource` of type HttpApi for a corresponding `AWS::Serverless::Function` resource.

`DefinitionUri`

The Amazon Simple Storage Service (Amazon S3) URI, local file path, or location object of the the OpenAPI definition that defines the HTTP API. The Amazon S3 object that this property references must be a valid OpenAPI definition file. If you don't specify a `DefinitionUri` or a `DefinitionBody` are specified, AWS SAM generates a `DefinitionBody` for you based on your template configuration.

If you provide a local file path, the template must go through the workflow that includes the `sam deploy` or `sam package` command for the definition to be transformed properly.

Intrinsic functions are not supported in external OpenApi definition files that you reference with `DefinitionUri`. To import an OpenApi definition into the template, use the `DefinitionBody` property with the Include transform.

Type: String | HttpApiDefinition

Required: No

AWS CloudFormation compatibility: This property is similar to the BodyS3Location property of an `AWS::ApiGatewayV2::Api` resource. The nested Amazon S3 properties are named differently.

`Description`

The description of the HTTP API resource.

When you specify `Description`, AWS SAM will modify the HTTP API resource's OpenApi definition by setting the `description` field. The following scenarios will result in an error:

- The `DefinitionBody` property is specified with the `description` field set in the Open API definition – This results in a conflict of the `description` field that AWS SAM won't resolve.

- The `DefinitionUri` property is specified – AWS SAM won't modify an Open API definition that is retrieved from Amazon S3.

Type: String

Required: No

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

DisableExecuteApiEndpoint

Specifies whether clients can invoke your HTTP API by using the default `execute-api` endpoint `https://{api_id}.execute-api.{region}.amazonaws.com`. By default, clients can invoke your API with the default endpoint. To require that clients only use a custom domain name to invoke your API, disable the default endpoint.

To use this property, you must specify the `DefinitionBody` property instead of the `DefinitionUri` property or define `x-amazon-apigateway-endpoint-configuration` with `disableExecuteApiEndpoint` in your OpenAPI definition.

Type: Boolean

Required: No

AWS CloudFormation compatibility: This property is similar to the [DisableExecuteApiEndpoint](#) property of an `AWS::ApiGatewayV2::Api` resource. It is passed directly to the `disableExecuteApiEndpoint` property of an [x-amazon-apigateway-endpoint-configuration](#) extension, which gets added to the [Body](#) property of an `AWS::ApiGatewayV2::Api` resource.

Domain

Configures a custom domain for this API Gateway HTTP API.

Type: [HttpApiDomainConfiguration](#)

Required: No

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

FailOnWarnings

Specifies whether to roll back the HTTP API creation (`true`) or not (`false`) when a warning is encountered. The default value is `false`.

Type: Boolean

Required: No

AWS CloudFormation compatibility: This property is passed directly to the <u>FailOnWarnings</u>
property of an `AWS::ApiGatewayV2::Api` resource.

Name

The name of the HTTP API resource.

When you specify `Name`, AWS SAM will modify the HTTP API resource's OpenAPI definition by
setting the `title` field. The following scenarios will result in an error:

- The `DefinitionBody` property is specified with the `title` field set in the Open API
  definition – This results in a conflict of the `title` field that AWS SAM won't resolve.

- The `DefinitionUri` property is specified – AWS SAM won't modify an Open API definition
  that is retrieved from Amazon S3.

Type: String

Required: No

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

PropagateTags

Indicate whether or not to pass tags from the `Tags` property to your <u>AWS::Serverless::HttpApi</u>
generated resources. Specify `True` to propagate tags in your generated resources.

Type: Boolean

Required: No

Default: `False`

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

RouteSettings

The route settings, per route, for this HTTP API. For more information, see <u>Working with routes
for HTTP APIs</u> in the *API Gateway Developer Guide*.

Type: <u>RouteSettings</u>

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 no name is specified, AWS SAM uses the $default stage from API Gateway.

Type: String

Required: No

Default: $default

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

StageVariables

A map that defines the stage variables. Variable names can have alphanumeric and underscore characters. The values must match [A-Za-z0-9-._~:/?#&=,]+.

Type: Json

Required: No

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

Tags

A map (string to string) that specifies the tags to add to this API Gateway stage. Keys can be 1 to 128 Unicode characters in length and cannot include the prefix aws:. You can use any of the following characters: the set of Unicode letters, digits, whitespace, _, ., /, =, +, and -. Values can be 1 to 256 Unicode characters in length.

Type: Map

Required: No

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

Additional notes: The Tags property requires AWS SAM to modify your OpenAPI definition, so tags are added only if the DefinitionBody property is specified—no tags are

added if the `DefinitionUri` property is specified. AWS SAM automatically adds an `httpapi:createdBy:SAM` tag. Tags are also added to the `AWS::ApiGatewayV2::Stage` resource and the `AWS::ApiGatewayV2::DomainName` resource (if `DomainName` is specified).

Return Values

Ref

When you pass the logical ID of this resource to the intrinsic `Ref` function, `Ref` returns the API ID of the underlying `AWS::ApiGatewayV2::Api` resource, for example, a1bcdef2gh.

For more information about using the `Ref` function, see [Ref](#) in the *AWS CloudFormation User Guide*.

Examples

Simple HttpApi

The following example shows the minimum needed to set up an HTTP API endpoint backed by an Lambda function. This example uses the default HTTP API that AWS SAM creates.

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Description: AWS SAM template with a simple API definition
Resources:
  ApiFunction:
    Type: AWS::Serverless::Function
    Properties:
      Events:
        ApiEvent:
          Type: HttpApi
      Handler: index.handler
      InlineCode: |
        def handler(event, context):
            return {'body': 'Hello World!', 'statusCode': 200}
      Runtime: python3.7
Transform: AWS::Serverless-2016-10-31
```

HttpApi with Auth

The following example shows how to set up authorization on HTTP API endpoints.

YAML

```
Properties:
  FailOnWarnings: true
  Auth:
    DefaultAuthorizer: OAuth2
    Authorizers:
      OAuth2:
        AuthorizationScopes:
          - scope4
        JwtConfiguration:
          issuer: "https://www.example.com/v1/connect/oauth2"
          audience:
            - MyApi
        IdentitySource: "$request.querystring.param"
```

HttpApi with OpenAPI definition

The following example shows how to add an OpenAPI definition to the template.

Note that AWS SAM fills in any missing Lambda integrations for HttpApi events that reference this HTTP API. AWS SAM also also adds any missing paths that HttpApi events reference.

YAML

```
Properties:
  FailOnWarnings: true
  DefinitionBody:
    info:
      version: '1.0'
      title:
        Ref: AWS::StackName
    paths:
      "/":
        get:
          security:
            - OpenIdAuth:
              - scope1
              - scope2
          responses: {}
    openapi: 3.0.1
    securitySchemes:
      OpenIdAuth:
```

```
            type: openIdConnect
          x-amazon-apigateway-authorizer:
            identitySource: "$request.querystring.param"
            type: jwt
            jwtConfiguration:
              audience:
              - MyApi
              issuer: https://www.example.com/v1/connect/oidc
            openIdConnectUrl: https://www.example.com/v1/connect/oidc/.well-known/openid-
  configuration
```

HttpApi with configuration settings

The following example shows how to add HTTP API and stage configurations to the template.

YAML

```
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 Amazon API Gateway HTTP API.

For more information about configuring access to HTTP APIs, see [Controlling and managing access to an HTTP API in API Gateway](#) in the *API Gateway Developer Guide*.

Syntax

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

YAML

```
Authorizers: OAuth2Authorizer | LambdaAuthorizer
DefaultAuthorizer: String
EnableIamAuthorizer: Boolean
```

Properties

`Authorizers`

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

Type: [OAuth2Authorizer](#) | [LambdaAuthorizer](#)

Required: No

Default: None

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

Additional notes: AWS SAM adds the authorizers to the OpenAPI definition.

`DefaultAuthorizer`

Specify the default authorizer to use for authorizing API calls to your API Gateway API. You can specify `AWS_IAM` as a default authorizer if `EnableIamAuthorizer` is set to `true`. Otherwise, specify an authorizer that you've defined in `Authorizers`.

Type: String

Required: No

Default: None

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

```
EnableIamAuthorizer
```

Specify whether to use IAM authorization for the API route.

Type: Boolean

Required: No

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

Examples

OAuth 2.0 Authorizer

OAuth 2.0 authorizer example

YAML

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

IAM authorizer

IAM authorizer example

YAML

```
Auth:
  EnableIamAuthorizer: true
  DefaultAuthorizer: AWS_IAM
```

LambdaAuthorizer

Configure a Lambda authorizer to control access to your Amazon API Gateway HTTP API with an AWS Lambda function.

For more information and examples, see [Working with AWS Lambda authorizers for HTTP APIs](#) in the *API Gateway Developer Guide*.

Syntax

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

YAML

```
AuthorizerPayloadFormatVersion: String
EnableFunctionDefaultPermissions: Boolean
EnableSimpleResponses: Boolean
FunctionArn: String
FunctionInvokeRole: String
Identity: LambdaAuthorizationIdentity
```

Properties

`AuthorizerPayloadFormatVersion`

Specifies the format of the payload sent to an HTTP API Lambda authorizer. Required for HTTP API Lambda authorizers.

This is passed through to the `authorizerPayloadFormatVersion` section of an `x-amazon-apigateway-authorizer` in the `securitySchemes` section of an OpenAPI definition.

Valid values: `1.0` or `2.0`

Type: String

Required: Yes

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

EnableFunctionDefaultPermissions

By default, the HTTP API resource is not granted permission to invoke the Lambda authorizer. Specify this property as `true` to automatically create permissions between your HTTP API resource and your Lambda authorizer.

Type: Boolean

Required: No

Default value: `false`

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

EnableSimpleResponses

Specifies whether a Lambda authorizer returns a response in a simple format. By default, a Lambda authorizer must return an AWS Identity and Access Management (IAM) policy. If enabled, the Lambda authorizer can return a boolean value instead of an IAM policy.

This is passed through to the `enableSimpleResponses` section of an `x-amazon-apigateway-authorizer` in the `securitySchemes` section of an OpenAPI definition.

Type: Boolean

Required: No

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

FunctionArn

The Amazon Resource Name (ARN) of the Lambda function that provides authorization for the API.

This is passed through to the `authorizerUri` section of an `x-amazon-apigateway-authorizer` in the `securitySchemes` section of an OpenAPI definition.

Type: String

Required: Yes

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

FunctionInvokeRole

The ARN of the IAM role that has the credentials required for API Gateway to invoke the authorizer function. Specify this parameter if your function's resource-based policy doesn't grant API Gateway `lambda:InvokeFunction` permission.

This is passed through to the `authorizerCredentials` section of an `x-amazon-apigateway-authorizer` in the `securitySchemes` section of an OpenAPI definition.

For more information, see [Create a Lambda authorizer](#) in the *API Gateway Developer Guide*.

Type: String

Required: No

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

Identity

Specifies an `IdentitySource` in an incoming request for an authorizer.

This is passed through to the `identitySource` section of an `x-amazon-apigateway-authorizer` in the `securitySchemes` section of an OpenAPI definition.

Type: [LambdaAuthorizationIdentity](#)

Required: No

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

Examples

LambdaAuthorizer

LambdaAuthorizer example

YAML

```
Auth:
```

```
Authorizers:
  MyLambdaAuthorizer:
    AuthorizerPayloadFormatVersion: 2.0
    FunctionArn:
      Fn::GetAtt:
        - MyAuthFunction
        - Arn
    FunctionInvokeRole:
      Fn::GetAtt:
        - LambdaAuthInvokeRole
        - Arn
    Identity:
      Headers:
        - Authorization
```

LambdaAuthorizationIdentity

Use property can be used to specify an IdentitySource in an incoming request for a Lambda authorizer. For more information about identity sources, see [Identity sources](#) in the *API Gateway Developer Guide*.

Syntax

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

YAML

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

Properties

Context

Converts the given context strings to a list of mapping expressions in the format `$context.contextString`.

Type: List

Required: No

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

Headers

Converts the headers to a list of mapping expressions in the format `$request.header.name`.

Type: List

Required: No

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

QueryStrings

Converts the given query strings to a list of mapping expressions in the format `$request.querystring.queryString`.

Type: List

Required: No

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

ReauthorizeEvery

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

Type: Integer

Required: No

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

StageVariables

Converts the given stage variables to a list of mapping expressions in the format `$stageVariables.stageVariable`.

Type: List

Required: No

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

Examples

LambdaRequestIdentity

Lambda request identity example

YAML

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

OAuth2Authorizer

Definition for an OAuth 2.0 authorizer, also known to as a JSON Web Token (JWT) authorizer.

For more information, see [Controlling access to HTTP APIs with JWT authorizers](#) in the *API Gateway
Developer Guide*.

Syntax

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

YAML

```
  AuthorizationScopes: List
```

```
    IdentitySource: String
    JwtConfiguration: Map
```

Properties

AuthorizationScopes

List of authorization scopes for this authorizer.

Type: List

Required: No

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

IdentitySource

Identity source expression for this authorizer.

Type: String

Required: No

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

JwtConfiguration

JWT configuration for this authorizer.

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

> ⓘ **Note**
>
> Properties issuer and audience are case insensitive and can be used either lowercase as in OpenAPI or uppercase Issuer and Audience as in AWS::ApiGatewayV2::Authorizer.

Type: Map

Required: No

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

Examples

OAuth 2.0 authorizer

OAuth 2.0 authorizer Example

YAML

```
Auth:
  Authorizers:
    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 *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

```
CorsConfiguration:
  AllowOrigins:
```

```
      - "https://example.com"
  AllowHeaders:
    - x-apigateway-header
  AllowMethods:
    - GET
  MaxAge: 600
  AllowCredentials: true
```

HttpApiDefinition

An OpenAPI document defining the API.

Syntax

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

YAML

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

Properties

Bucket

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

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the Bucket property of the AWS::ApiGatewayV2::Api BodyS3Location data type.

Key

The Amazon S3 key of the OpenAPI file.

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the [Key](#) property of the `AWS::ApiGatewayV2::Api BodyS3Location` data type.

`Version`

For versioned objects, the version of the OpenAPI file.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [Version](#) property of the `AWS::ApiGatewayV2::Api BodyS3Location` data type.

Examples

Definition Uri example

API Definition example

YAML

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

HttpApiDomainConfiguration

Configures a custom domain for an API.

Syntax

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

YAML

```
  BasePath: List
  CertificateArn: String
  DomainName: String
  EndpointConfiguration: String
  MutualTlsAuthentication: MutualTlsAuthentication
```

```
      OwnershipVerificationCertificateArn: String
      Route53: Route53Configuration
      SecurityPolicy: String
```

Properties

BasePath

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

Type: List

Required: No

Default: /

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

CertificateArn

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

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the CertificateArn
property of an `AWS::ApiGateway2::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. For information
about generated AWS CloudFormation resources, see Generated AWS CloudFormation
resources.

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the [DomainName](#) property of an `AWS::ApiGateway2::DomainName` resource.

EndpointConfiguration

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

The only valid value for HTTP APIs is REGIONAL.

Type: String

Required: No

Default: REGIONAL

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

MutualTlsAuthentication

The mutual transport layer security (TLS) authentication configuration for a custom domain name.

Type: [MutualTlsAuthentication](#)

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [MutualTlsAuthentication](#) property of an `AWS::ApiGatewayV2::DomainName` resource.

OwnershipVerificationCertificateArn

The ARN of the public certificate issued by ACM to validate ownership of your custom domain. Required only when you configure mutual TLS and you specify an ACM imported or private CA certificate ARN for the `CertificateArn`.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [OwnershipVerificationCertificateArn](#) property of the `AWS::ApiGatewayV2::DomainName DomainNameConfiguration` data type.

```
Route53
```

Defines an Amazon Route 53 configuration.

Type: [Route53Configuration](#)

Required: No

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

```
SecurityPolicy
```

The TLS version of the security policy for this domain name.

The only valid value for HTTP APIs is TLS_1_2.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [SecurityPolicy](#) property of the AWS::ApiGatewayV2::DomainName DomainNameConfiguration data type.

Examples

DomainName

DomainName example

YAML

```
Domain:
  DomainName: www.example.com
  CertificateArn: arn-example
  EndpointConfiguration: REGIONAL
  Route53:
    HostedZoneId: Z1PA6795UKMFR9
  BasePath:
    - foo
    - bar
```

Route53Configuration

Configures the Route53 record sets for an API.

Syntax

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

YAML

```
DistributionDomainName: String
EvaluateTargetHealth: Boolean
HostedZoneId: String
HostedZoneName: String
IpV6: Boolean
Region: String
SetIdentifier: String
```

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. You must include a trailing dot
(for example, www.example.com.) as part of the HostedZoneName.

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.

Region

Latency-based resource record sets only: The Amazon EC2 Region where you created the resource that this resource record set refers to. The resource typically is an AWS resource, such as an EC2 instance or an ELB load balancer, and is referred to by an IP address or a DNS domain name, depending on the record type.

When Amazon Route 53 receives a DNS query for a domain name and type for which you have created latency resource record sets, Route 53 selects the latency resource record set that has the lowest latency between the end user and the associated Amazon EC2 Region. Route 53 then returns the value that is associated with the selected resource record set.

Note the following:

- You can only specify one `ResourceRecord` per latency resource record set.
- You can only create one latency resource record set for each Amazon EC2 Region.
- You aren't required to create latency resource record sets for all Amazon EC2 Regions. Route 53 will choose the region with the best latency from among the regions that you create latency resource record sets for.
- You can't create non-latency resource record sets that have the same values for the `Name` and `Type` elements as latency resource record sets.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [Region](#) property of an `AWS::Route53::RecordSetGroup RecordSet` data type.

SetIdentifier

Resource record sets that have a routing policy other than simple: An identifier that differentiates among multiple resource record sets that have the same combination of name and type, such as multiple weighted resource record sets named acme.example.com that have a type of A. In a group of resource record sets that have the same name and type, the value of `SetIdentifier` must be unique for each resource record set.

For information about routing policies, see [Choosing a routing policy](#) in the *Amazon Route 53 Developer Guide*.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the `SetIdentifier` property of an `AWS::Route53::RecordSetGroup` `RecordSet` data type.

Examples

Route 53 Configuration Example

This example shows how to configure Route 53.

YAML

```
Domain:
  DomainName: www.example.com
  CertificateArn: arn-example
  EndpointConfiguration: EDGE
  Route53:
    HostedZoneId: Z1PA6795UKMFR9
    EvaluateTargetHealth: true
    DistributionDomainName: xyz
```

AWS::Serverless::LayerVersion

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

The AWS::Serverless::LayerVersion 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.

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.

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.

> ⓘ **Note**
>
> When you deploy to AWS CloudFormation, AWS SAM transforms your AWS SAM
> resources into AWS CloudFormation resources. For more information, see Generated AWS
> CloudFormation resources.

Syntax

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

YAML

```
Type: AWS::Serverless::LayerVersion
Properties:
  CompatibleArchitectures: List
  CompatibleRuntimes: List
  ContentUri: String | LayerContent
  Description: String
  LayerName: String
  LicenseInfo: String
  RetentionPolicy: String
```

Properties

CompatibleArchitectures

Specifies the supported instruction set architectures for the layer version.

For more information about this property, see Lambda instruction set architectures in the *AWS
Lambda Developer Guide*.

Valid values: x86_64, arm64

Type: List

Required: No

Default: x86_64

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

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

Amazon S3 Uri, path to local folder, or LayerContent object of the layer code.

If an Amazon S3 Uri or LayerContent object is provided, The Amazon S3 object referenced must
be a valid ZIP archive that contains the contents of an Lambda layer.

If a path to a local folder is provided, for the content to be transformed properly the template
must go through the workflow that includes sam build followed by either sam deploy or
sam package. By default, relative paths are resolved with respect to the AWS SAM template's
location.

Type: String | LayerContent

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`

This property specifies whether old versions of your `LayerVersion` are retained or deleted when you delete a resource. If you need to retain old versions of your `LayerVersion` when updating or replacing a resource, you must have the `UpdateReplacePolicy` attribute enabled. For information on doing this, refer to [UpdateReplacePolicy attribute](#) in the *AWS CloudFormation User Guide*.

Valid values: `Retain` or `Delete`

Type: String

Required: No

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

Additional notes: When you specify `Retain`, AWS SAM adds a [Resource attributes](#) 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

```
Properties:
  LayerName: MyLayer
  Description: Layer description
  ContentUri: 's3://my-bucket/my-layer.zip'
  CompatibleRuntimes:
    - nodejs10.x
    - nodejs12.x
  LicenseInfo: 'Available under the MIT-0 license.'
  RetentionPolicy: Retain
```

LayerContent

A ZIP archive that contains the contents of an [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.

> ⓘ **Note**
>
> When you deploy to AWS CloudFormation, AWS SAM transforms your AWS SAM resources into AWS CloudFormation resources. For more information, see Generated AWS CloudFormation resources.

Syntax

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

YAML

```
Type: AWS::Serverless::SimpleTable
Properties:
  PointInTimeRecoverySpecification: PointInTimeRecoverySpecification
  PrimaryKey: PrimaryKeyObject
  ProvisionedThroughput: ProvisionedThroughput
  SSESpecification: SSESpecification
  TableName: String
  Tags: Map
```

Properties

PointInTimeRecoverySpecification

The settings used to enable point in time recovery.

Type: [PointInTimeRecoverySpecification](#)

Required: No

AWS CloudFormation compatibility: This property is passed directly to the
[PointInTimeRecoverySpecification](#) property of an `AWS::DynamoDB::Table` resource.

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](#)

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 <u>SSESpecification</u> 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 <u>TableName</u> property of an `AWS::DynamoDB::Table` resource.

Tags

A map (string to string) that specifies the tags to be added to this SimpleTable. For details about valid keys and values for tags, see <u>Resource tag</u> in the *AWS CloudFormation User Guide*.

Type: Map

Required: No

AWS CloudFormation compatibility: This property is similar to the <u>Tags</u> 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 <u>Ref</u> 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.

Valid values: `String, Number, Binary`

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the [`AttributeType`] property of the `AWS::DynamoDB::Table AttributeDefinition` data type.

Examples

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](#).

> ⓘ **Note**
>
> When you deploy to AWS CloudFormation, AWS SAM transforms your AWS SAM resources into AWS CloudFormation resources. For more information, see [Generated AWS CloudFormation resources](#).

Syntax

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

YAML

```
Type: AWS::Serverless::StateMachine
Properties:
  AutoPublishAlias: String
  Definition: Map
  DefinitionSubstitutions: Map
  DefinitionUri: String | S3Location
  DeploymentPreference: DeploymentPreference
  Events: EventSource
  Logging: LoggingConfiguration
  Name: String
  PermissionsBoundary: String
  Policies: String | List | Map
  PropagateTags: Boolean
  RolePath: String
  Role: String
  Tags: Map
  Tracing: TracingConfiguration
  Type: String
```

Properties

AutoPublishAlias

The name of the state machine alias. To learn more about using Step Functions state machine aliases, see Manage continuous deployments with versions and aliases in the *AWS Step Functions Developer Guide*.

Use `DeploymentPreference` to configure deployment preferences for your alias. If you don't specify `DeploymentPreference`, AWS SAM will configure traffic to shift to the newer state machine version all at once.

AWS SAM sets the version's `DeletionPolicy` and `UpdateReplacePolicy` to `Retain` by default. Previous versions will not be deleted automatically.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the   Name property of an `AWS::StepFunctions::StateMachineAlias` resource.

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](#).

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](#)

Required: Conditional

AWS CloudFormation compatibility: This property is passed directly to the DefinitionS3Location property of an AWS::StepFunctions::StateMachine resource.

DeploymentPreference

The settings that enable and configure gradual state machine deployments. To learn more about Step Functions gradual deployments, see Manage continuous deployments with versions and aliases in the *AWS Step Functions Developer Guide*.

Specify AutoPublishAlias before configuring this property. Your DeploymentPreference settings will be applied to the alias specified with AutoPublishAlias.

When you specify DeploymentPreference, AWS SAM generates the StateMachineVersionArn sub-property value automatically.

Type: DeploymentPreference

Required: No

AWS CloudFormation compatibility: AWS SAM generates and attaches the StateMachineVersionArn property value to DeploymentPreference and passes DeploymentPreference to the DeploymentPreference property of an AWS::StepFunctions::StateMachineAlias 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

Required: No

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

Logging

Defines which execution history events are logged and where they are logged.

Type: LoggingConfiguration

Required: No

AWS CloudFormation compatibility: This property is passed directly to the
`LoggingConfiguration` property of an `AWS::StepFunctions::StateMachine` resource.

Name

The name of the state machine.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the `StateMachineName`
property of an `AWS::StepFunctions::StateMachine` resource.

PermissionsBoundary

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

Type: String

Required: No

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

Policies

Permission policies for this state machine. Policies will be appended to the state machine's
default AWS Identity and Access Management (IAM) execution role.

This property accepts a single value or list of values. Allowed values include:

- AWS SAM policy templates.
- The ARN of an AWS managed policy or customer managed policy.
- The name of an AWS managed policy from the following list.
- An inline IAM policy formatted in YAML as a map.

> **ⓘ Note**
>
> If you set the `Role` property, this property is ignored.

Type: String | List | Map

Required: No

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

PropagateTags

Indicate whether or not to pass tags from the Tags property to your
[AWS::Serverless::StateMachine](#) generated resources. Specify True to propagate tags in your
generated resources.

Type: Boolean

Required: No

Default: False

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

Role

The ARN of an IAM role to use as this state machine's execution role.

Type: String

Required: Conditional

AWS CloudFormation compatibility: This property is passed directly to the [RoleArn](#) property
of an AWS::StepFunctions::StateMachine resource.

RolePath

The path to the state machine's IAM execution role.

Use this property when the role is generated for you. Do not use when the role is specified with
the Role property.

Type: String

Required: Conditional

AWS CloudFormation compatibility: This property is passed directly to the <u>Path</u> property of an
`AWS::IAM::Role` resource.

Tags

A string-to-string map that specifies the tags added to the state machine and the
corresponding execution role. For information about valid keys and values for tags, see the <u>Tags</u>
property of an <u>AWS::StepFunctions::StateMachine</u> resource.

Type: Map

Required: No

AWS CloudFormation compatibility: This property is similar to the <u>Tags</u> property of an
`AWS::StepFunctions::StateMachine` resource. AWS SAM automatically adds a
`stateMachine:createdBy:SAM` tag to this resource, and to the default role that is generated
for it.

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 <u>AWS X-Ray and Step Functions</u> in the *AWS Step Functions
Developer Guide*.

Type: <u>TracingConfiguration</u>

Required: No

AWS CloudFormation compatibility: This property is passed directly to the
<u>TracingConfiguration</u> 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 <u>StateMachineType</u>
property of an `AWS::StepFunctions::StateMachine` resource.

Return Values

Ref

When you provide the logical ID of this resource to the Ref intrinsic function, Ref returns the Amazon Resource Name (ARN) of the underlying `AWS::StepFunctions::StateMachine` resource.

For more information about using the `Ref` function, see [Ref](#) in the *AWS CloudFormation User Guide*.

Fn::GetAtt

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

For more information about using `Fn::GetAtt`, see [Fn::GetAtt](#) in the *AWS CloudFormation User Guide*.

Name

Returns the name of the state machine, such as `HelloWorld-StateMachine`.

Examples

State Machine Definition File

The following is an example of an inline state machine definition that allows a lambda function to invoke state machine. Note that this example expects the `Role` property to configure proper policy to allow invocation. The `my_state_machine.asl.json` file must be written in the [Amazon States Language](#).

In this example, the `DefinitionSubstitution` entries allow the state machine to include resources that are declared in the AWS SAM template file.

YAML

```
MySampleStateMachine:
  Type: AWS::Serverless::StateMachine
  Properties:
```

```
      DefinitionUri: statemachine/my_state_machine.asl.json
      Role: arn:aws:iam::123456123456:role/service-role/my-sample-role
      Tracing:
        Enabled: true
      DefinitionSubstitutions:
        MyFunctionArn: !GetAtt MyFunction.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
          Resource: arn:aws:lambda:us-east-1:123456123456:function:my-sample-lambda-app
          End: true
    Role: arn:aws:iam::123456123456:role/service-role/my-sample-role
    Tracing:
      Enabled: true
```

EventSource

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

Syntax

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

YAML

```
  Properties: Schedule | ScheduleV2 | CloudWatchEvent | EventBridgeRule | Api
  Type: String
```

Properties

Properties

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

Type: Schedule | ScheduleV2 | CloudWatchEvent | EventBridgeRule | Api

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, ScheduleV2, 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

```
ApiEvent:
  Type: Api
  Properties:
```

```
    Method: get
    Path: /group/{user}
    RestApiId:
      Ref: MyApi
```

Api

The object describing an `Api` event source type. If an [AWS::Serverless::Api](#) 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

```
Auth: ApiStateMachineAuth
Method: String
Path: String
RestApiId: String
UnescapeMappingTemplate: Boolean
```

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](#)

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](#) resource that is defined in this template.

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

`UnescapeMappingTemplate`

Unescapes single quotes, by replacing `\'` with `'`, on the input that is passed to the state machine. Use when your input contains single quotes.

> **ⓘ Note**
>
> If set to `False` and your input contains single quotes, an error will occur.

Type: Boolean

Required: No

Default: False

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

```
Events:
  ApiEvent:
    Type: Api
    Properties:
      Path: /path
      Method: get
```

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

```
ApiKeyRequired: Boolean
```

```
    AuthorizationScopes: List
    Authorizer: String
    ResourcePolicy: ResourcePolicyStatement
```

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

Required: No

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

Examples

StateMachine-Auth

The following example specifies authorization at the state machine level.

YAML

```
Auth:
  ApiKeyRequired: true
  Authorizer: NONE
```

ResourcePolicyStatement

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

Syntax

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

YAML

```
  AwsAccountBlacklist: List
  AwsAccountWhitelist: List
  CustomStatements: List
  IntrinsicVpcBlacklist: List
  IntrinsicVpcWhitelist: List
```

```
        IntrinsicVpceBlacklist: List
        IntrinsicVpceWhitelist: List
        IpRangeBlacklist: List
        IpRangeWhitelist: List
        SourceVpcBlacklist: List
        SourceVpcWhitelist: List
```

Properties

AwsAccountBlacklist

   The AWS accounts to block.

   Type: List of String

   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 of String

   Required: No

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

CustomStatements

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

   Type: List

   Required: No

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

`IntrinsicVpcBlacklist`

The list of virtual private clouds (VPCs) to block, where each VPC is specified as a reference such as a [dynamic reference](#) or the `Ref` [intrinsic function](#). For an example use of this property, see the Examples section at the bottom of this page.

Type: List

Required: No

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

`IntrinsicVpcWhitelist`

The list of VPCs to allow, where each VPC is specified as a reference such as a [dynamic reference](#) or the `Ref` [intrinsic function](#).

Type: List

Required: No

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

`IntrinsicVpceBlacklist`

The list of VPC endpoints to block, where each VPC endpoint is specified as a reference such as a [dynamic reference](#) or the `Ref` [intrinsic function](#).

Type: List

Required: No

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

`IntrinsicVpceWhitelist`

The list of VPC endpoints to allow, where each VPC endpoint is specified as a reference such as a [dynamic reference](#) or the `Ref` [intrinsic function](#). For an example use of this property, see the Examples section at the bottom of this page.

Type: List

Required: No

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

`IpRangeBlacklist`

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

Type: List

Required: No

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

`IpRangeWhitelist`

The IP addresses or address ranges to allow.

Type: List

Required: No

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

`SourceVpcBlacklist`

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

Type: List

Required: No

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

`SourceVpcWhitelist`

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

Type: List

Required: No

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

Examples

Resource Policy Example

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

YAML

```
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](#) 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:
          - running
```

EventBridgeRule

The object describing an `EventBridgeRule` event source type, which sets your state machine as the target for an Amazon EventBridge rule. For more information, see What Is Amazon EventBridge? in the *Amazon EventBridge User Guide*.

AWS SAM generates an AWS::Events::Rule resource when this event type is set.

Syntax

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

YAML

```
DeadLetterConfig: DeadLetterConfig
EventBusName: String
Input: String
InputPath: String
InputTransformer: InputTransformer
Pattern: EventPattern
RetryPolicy: RetryPolicy
RuleName: String
State: String
Target: Target
```

Properties

`DeadLetterConfig`

Configure the Amazon Simple Queue Service (Amazon SQS) queue where EventBridge sends events after a failed target invocation. Invocation can fail, for example, when sending an event to a Lambda function that doesn't exist, or when EventBridge has insufficient permissions to invoke the Lambda function. For more information, see Event retry policy and using dead-letter queues in the *Amazon EventBridge User Guide*.

Type: DeadLetterConfig

Required: No

AWS CloudFormation compatibility: This property is similar to the <u>DeadLetterConfig</u> property of the `AWS::Events::Rule Target` data type. The AWS SAM version of this property includes additional subproperties, in case you want AWS SAM to create the dead-letter queue for you.

EventBusName

The event bus to associate with this rule. If you omit this property, AWS SAM uses the default event bus.

Type: String

Required: No

Default: Default event bus

AWS CloudFormation compatibility: This property is passed directly to the <u>EventBusName</u> 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 <u>Input</u> 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 <u>InputPath</u> property of an `AWS::Events::Rule Target` resource.

InputTransformer

Settings to enable you to provide custom input to a target based on certain event data. You can extract one or more key-value pairs from the event and then use that data to send customized input to the target. For more information, see Amazon EventBridge input transformation in the *Amazon EventBridge User Guide*.

Type: InputTransformer

Required: No

AWS CloudFormation compatibility: This property is passed directly to the `InputTransformer` property of an `AWS::Events::Rule Target` data type.

Pattern

Describes which events are routed to the specified target. For more information, see Events and Event Patterns in EventBridge in the *Amazon EventBridge User Guide*.

Type: EventPattern

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the `EventPattern` property of an `AWS::Events::Rule` resource.

RetryPolicy

A `RetryPolicy` object that includes information about the retry policy settings. For more information, see Event retry policy and using dead-letter queues in the *Amazon EventBridge User Guide*.

Type: RetryPolicy

Required: No

AWS CloudFormation compatibility: This property is passed directly to the `RetryPolicy` property of the `AWS::Events::Rule Target` data type.

RuleName

The name of the rule.

Type: String

Required: No

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

State

The state of the rule.

Valid values: [DISABLED | ENABLED]

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the State property of an
AWS::Events::Rule resource.

Target

The AWS resource that EventBridge invokes when a rule is triggered. You can use this property
to specify the logical ID of the target. If this property is not specified, then AWS SAM generates
the logical ID of the target.

Type: Target

Required: No

AWS CloudFormation compatibility: This property is similar to the Targets property of an
AWS::Events::Rule resource. The AWS SAM version of this property only allows you to
specify the logical ID of a single target.

Examples

EventBridgeRule

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

YAML

```
EBRule:
  Type: EventBridgeRule
  Properties:
    Input: '{"Key": "Value"}'
```

```
    Pattern:
      detail:
        state:
          - terminated
```

DeadLetterConfig

The object used to specify the Amazon Simple Queue Service (Amazon SQS) queue where EventBridge sends events after a failed target invocation. Invocation can fail, for example, when sending an event to a state machine that doesn't exist, or insufficient permissions to invoke the state machine. For more information, see [Event retry policy and using dead-letter queues](#) in the *Amazon EventBridge User Guide*.

Syntax

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

YAML

```
Arn: String
QueueLogicalId: String
Type: String
```

Properties

Arn

The Amazon Resource Name (ARN) of the Amazon SQS queue specified as the target for the dead-letter queue.

> **ⓘ Note**
>
> Specify either the `Type` property or `Arn` property, but not both.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [Arn](#) property of the `AWS::Events::Rule DeadLetterConfig` data type.

`QueueLogicalId`

The custom name of the dead letter queue that AWS SAM creates if `Type` is specified.

> ⓘ **Note**
>
> If the `Type` property is not set, this property is ignored.

Type: String

Required: No

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

`Type`

The type of the queue. When this property is set, AWS SAM automatically creates a dead-letter queue and attaches necessary [resource-based policy](#) to grant permission to rule resource to send events to the queue.

> ⓘ **Note**
>
> Specify either the `Type` property or `Arn` property, but not both.

Valid values: SQS

Type: String

Required: No

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

Examples

DeadLetterConfig

DeadLetterConfig

YAML

```
DeadLetterConfig:
  Type: SQS
  QueueLogicalId: MyDLQ
```

Target

Configures the AWS resource that EventBridge invokes when a rule is triggered.

Syntax

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

YAML

```
  Id: String
```

Properties

`Id`

The logical ID of the target.

The value of `Id` can include alphanumeric characters, periods (`.`), hyphens (`-`), and underscores (`_`).

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the [Id] property of the `AWS::Events::Rule Target` data type.

Examples

Target

YAML

```
EBRule:
```

```
   Type: EventBridgeRule
   Properties:
     Target:
       Id: MyTarget
```

Schedule

The object describing a `Schedule` event source type, which sets your state machine as the target of an EventBridge rule that triggers on a schedule. For more information, see What Is Amazon EventBridge? in the *Amazon EventBridge User Guide*.

AWS Serverless Application Model (AWS SAM) generates an AWS::Events::Rule resource when this event type is set.

Syntax

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

YAML

```
   DeadLetterConfig: DeadLetterConfig
   Description: String
   Enabled: Boolean
   Input: String
   Name: String
   RetryPolicy: RetryPolicy
   RoleArn: String
   Schedule: String
   State: String
   Target: Target
```

Properties

`DeadLetterConfig`

Configure the Amazon Simple Queue Service (Amazon SQS) queue where EventBridge sends events after a failed target invocation. Invocation can fail, for example, when sending an event to a Lambda function that doesn't exist, or when EventBridge has insufficient permissions to invoke the Lambda function. For more information, see Event retry policy and using dead-letter queues in the *Amazon EventBridge User Guide*.

Type: DeadLetterConfig

Required: No

AWS CloudFormation compatibility: This property is similar to the DeadLetterConfig property of the `AWS::Events::Rule Target` data type. The AWS SAM version of this property includes additional subproperties, in case you want AWS SAM to create the dead-letter queue for you.

`Description`

A description of the rule.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the Description property of an `AWS::Events::Rule` resource.

`Enabled`

Indicates whether the rule is enabled.

To disable the rule, set this property to `false`.

> **ⓘ Note**
>
> Specify either the `Enabled` or `State` property, but not both.

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 Input property of an
AWS::Events::Rule Target resource.

Name

The name of the rule. If you don't specify a name, AWS CloudFormation generates a unique
physical ID and uses that ID for the rule name.

Type: String

Required: No

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

RetryPolicy

A RetryPolicy object that includes information about the retry policy settings. For more
information, see Event retry policy and using dead-letter queues in the *Amazon EventBridge
User Guide*.

Type: RetryPolicy

Required: No

AWS CloudFormation compatibility: This property is passed directly to the RetryPolicy
property of the AWS::Events::Rule Target data type.

RoleArn

The ARN of the IAM role that EventBridge Scheduler will use for the target when the schedule is
invoked.

Type: RoleArn

Required: No. If not provided, a new role will be created and used.

AWS CloudFormation compatibility: This property is passed directly to the RoleArn property of
the AWS::Scheduler::Schedule Target data type.

```
Schedule
```

The scheduling expression that determines when and how often the rule runs. For more information, see Schedule Expressions for Rules.

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the ScheduleExpression property of an `AWS::Events::Rule` resource.

```
State
```

The state of the rule.

Accepted values: `DISABLED | ENABLED`

> **ⓘ Note**
>
> Specify either the `Enabled` or `State` property, but not both.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the State property of an `AWS::Events::Rule` resource.

```
Target
```

The AWS resource that EventBridge invokes when a rule is triggered. You can use this property to specify the logical ID of the target. If this property is not specified, then AWS SAM generates the logical ID of the target.

Type: Target

Required: No

AWS CloudFormation compatibility: This property is similar to the Targets property of an `AWS::Events::Rule` resource. The AWS SAM version of this property only allows you to specify the logical ID of a single target.

Examples

CloudWatch Schedule Event

CloudWatch Schedule Event Example

YAML

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

DeadLetterConfig

The object used to specify the Amazon Simple Queue Service (Amazon SQS) queue where EventBridge sends events after a failed target invocation. Invocation can fail, for example, when sending an event to a state machine that doesn't exist, or insufficient permissions to invoke the state machine. For more information, see [Event retry policy and using dead-letter queues](#) in the *Amazon EventBridge User Guide*.

Syntax

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

YAML

```
  Arn: String
  QueueLogicalId: String
  Type: String
```

Properties

Arn

The Amazon Resource Name (ARN) of the Amazon SQS queue specified as the target for the dead-letter queue.

> **ⓘ Note**
>
> Specify either the `Type` property or `Arn` property, but not both.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [Arn] property of the `AWS::Events::Rule DeadLetterConfig` data type.

`QueueLogicalId`

The custom name of the dead letter queue that AWS SAM creates if `Type` is specified.

> **ⓘ Note**
>
> If the `Type` property is not set, this property is ignored.

Type: String

Required: No

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

`Type`

The type of the queue. When this property is set, AWS SAM automatically creates a dead-letter queue and attaches necessary [resource-based policy] to grant permission to rule resource to send events to the queue.

> **ⓘ Note**
>
> Specify either the `Type` property or `Arn` property, but not both.

Valid values: SQS

Type: String

Required: No

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

Examples

DeadLetterConfig

DeadLetterConfig

YAML

```
DeadLetterConfig:
  Type: SQS
  QueueLogicalId: MyDLQ
```

Target

Configures the AWS resource that EventBridge invokes when a rule is triggered.

Syntax

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

YAML

```
  Id: String
```

Properties

Id

The logical ID of the target.

The value of `Id` can include alphanumeric characters, periods (`.`), hyphens (`-`), and underscores (`_`).

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the <u>Id</u> property of the
`AWS::Events::Rule` `Target` data type.

Examples

Target

YAML

```
EBRule:
  Type: Schedule
  Properties:
    Target:
      Id: MyTarget
```

ScheduleV2

The object describing a `ScheduleV2` event source type, which sets your state machine as
the target of an Amazon EventBridge Scheduler event that triggers on a schedule. For more
information, see <u>What is Amazon EventBridge Scheduler?</u> in the *EventBridge Scheduler User Guide.*

AWS Serverless Application Model (AWS SAM) generates an <u>AWS::Scheduler::Schedule</u> 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

```
DeadLetterConfig: DeadLetterConfig
Description: String
EndDate: String
FlexibleTimeWindow: FlexibleTimeWindow
GroupName: String
Input: String
KmsKeyArn: String
Name: String
```

```
OmitName: Boolean
PermissionsBoundary: String
RetryPolicy: RetryPolicy
RoleArn: String
ScheduleExpression: String
ScheduleExpressionTimezone: String
StartDate: String
State: String
```

Properties

DeadLetterConfig

Configure the Amazon Simple Queue Service (Amazon SQS) queue where EventBridge sends events after a failed target invocation. Invocation can fail, for example, when sending an event to a Lambda function that doesn't exist, or when EventBridge has insufficient permissions to invoke the Lambda function. For more information, see Configuring a dead-letter queue for EventBridge Scheduler in the *EventBridge Scheduler User Guide*.

Type: DeadLetterConfig

Required: No

AWS CloudFormation compatibility: This property is similar to the DeadLetterConfig property of the AWS::Scheduler::Schedule Target data type. The AWS SAM version of this property includes additional subproperties, in case you want AWS SAM to create the dead-letter queue for you.

Description

A description of the schedule.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the Description property of an AWS::Scheduler::Schedule resource.

EndDate

The date, in UTC, before which the schedule can invoke its target. Depending on the schedule's recurrence expression, invocations might stop on, or before, the **EndDate** you specify.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the EndDate property of an `AWS::Scheduler::Schedule` resource.

FlexibleTimeWindow

Allows configuration of a window within which a schedule can be invoked.

Type: FlexibleTimeWindow

Required: No

AWS CloudFormation compatibility: This property is passed directly to the FlexibleTimeWindow property of an `AWS::Scheduler::Schedule` resource.

GroupName

The name of the schedule group to associate with this schedule. If not defined, the default group is used.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the GroupName property of an `AWS::Scheduler::Schedule` 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::Scheduler::Schedule Target` resource.

KmsKeyArn

The ARN for a KMS Key that will be used to encrypt customer data.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [KmsKeyArn](#) property of an `AWS::Scheduler::Schedule` resource.

Name

The name of the schedule. If you don't specify a name, AWS SAM generates a name in the format *StateMachine-Logical-IDEvent-Source-Name* and uses that ID for the schedule name.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [Name](#) property of an `AWS::Scheduler::Schedule` resource.

OmitName

By default, AWS SAM generates and uses a schedule name in the format of *<State-machine-logical-ID><event-source-name>*. Set this property to `true` to have AWS CloudFormation generate a unique physical ID and use that for the schedule name instead.

Type: Boolean

Required: No

Default: `false`

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

PermissionsBoundary

The ARN of the policy used to set the permissions boundary for the role.

> **ⓘ Note**
>
> If `PermissionsBoundary` is defined, AWS SAM will apply the same boundaries to the scheduler schedule's target IAM role.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the
[PermissionsBoundary](#) property of an AWS::IAM::Role resource.

RetryPolicy

A RetryPolicy object that includes information about the retry policy settings.

Type: [RetryPolicy](#)

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [RetryPolicy](#)
property of the AWS::Scheduler::Schedule Target data type.

RoleArn

The ARN of the IAM role that EventBridge Scheduler will use for the target when the schedule is
invoked.

Type: [RoleArn](#)

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [RoleArn](#) property of
the AWS::Scheduler::Schedule Target data type.

ScheduleExpression

The scheduling expression that determines when and how often the schedule runs.

Type: String

Required: Yes

AWS CloudFormation compatibility: This property is passed directly to the
[ScheduleExpression](#) property of an AWS::Scheduler::Schedule resource.

ScheduleExpressionTimezone

The timezone in which the scheduling expression is evaluated.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the
[ScheduleExpressionTimezone](#) property of an `AWS::Scheduler::Schedule` resource.

StartDate

The date, in UTC, after which the schedule can begin invoking a target. Depending on the schedule's recurrence expression, invocations might occur on, or after, the **StartDate** you specify.

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [StartDate](#) property of an `AWS::Scheduler::Schedule` resource.

State

The state of the schedule.

Accepted values: `DISABLED | ENABLED`

Type: String

Required: No

AWS CloudFormation compatibility: This property is passed directly to the [State](#) property of an `AWS::Scheduler::Schedule` resource.

Examples

Basic example of defining a ScheduleV2 resource

```
StateMachine:
  Type: AWS::Serverless::StateMachine
  Properties:
    Name: MyStateMachine
```

```
    Events:
      ScheduleEvent:
        Type: ScheduleV2
        Properties:
          ScheduleExpression: "rate(1 minute)"
      ComplexScheduleEvent:
        Type: ScheduleV2
        Properties:
          ScheduleExpression: rate(1 minute)
          FlexibleTimeWindow:
            Mode: FLEXIBLE
            MaximumWindowInMinutes: 5
          StartDate: '2022-12-28T12:00:00.000Z'
          EndDate: '2023-01-28T12:00:00.000Z'
          ScheduleExpressionTimezone: UTC
          RetryPolicy:
            MaximumRetryAttempts: 5
            MaximumEventAgeInSeconds: 300
          DeadLetterConfig:
            Type: SQS
  DefinitionUri:
    Bucket: sam-demo-bucket
    Key: my-state-machine.asl.json
    Version: 3
  Policies:
    - LambdaInvokePolicy:
        FunctionName: !Ref MyFunction
```

For reference information for all the AWS resource and property types AWS CloudFormation and AWS SAM support, see [AWS resource and property types reference](#) in the *AWS CloudFormation User Guide*.

Resource attributes

Resource attributes are attributes that you can add to AWS SAM and AWS CloudFormation resources to control additional behaviors and relationships. For more information about resource attributes, see [Resource Attribute Reference](#) in the *AWS CloudFormation User Guide*.

AWS SAM support a subset of resource attributes that are defined by AWS CloudFormation. Of the supported resource attributes, some are copied to only the base generated AWS CloudFormation resource of the corresponding AWS SAM resource, and some are copied to all generated AWS

CloudFormation resources resulting from the corresponding AWS SAM resource. For more information about AWS CloudFormation resources generated from corresponding AWS SAM resources, see Generated AWS CloudFormation resources.

The following table summarizes resource attribute support by AWS SAM, subject to the Exceptions listed below.

Resource attributes	Destination generated resource(s)
DependsOn Metadata [1, 2]	Base AWS CloudFormation generated resource only. For information about the mapping between AWS SAM resources and base AWS CloudFormation resources, see Generated AWS CloudFormation resource scenarios.
Condition DeletionPolicy UpdateReplacePolicy	All generated AWS CloudFormation resources from the corresponding AWS SAM resource. For information about scenarios for generated AWS CloudFormation resources, see Generated AWS CloudFormation resource scenarios.

Notes:

1. For more information about using the `Metadata` resource attribute with the `AWS::Serverless::Function` resource type, see Building custom runtimes.

2. For more information about using the `Metadata` resource attribute with the `AWS::Serverless::LayerVersion` resource type, see Building layers.

Exceptions

There are a number of exceptions to the resource attribute rules described previously:

- For `AWS::Lambda::LayerVersion`, the AWS SAM-only custom field `RetentionPolicy` sets the `DeletionPolicy` for the generated AWS CloudFormation resources. This has a higher precedence than `DeletionPolicy` itself. If neither is set, then by default `DeletionPolicy` is set to `Retain`.

- For `AWS::Lambda::Version`, if `DeletionPolicy` is not specified, the default is `Retain`.

- For the scenario where `DeploymentPreferences` is specified for a serverless function, resource attributes are not copied to the following generated AWS CloudFormation resources:

  - `AWS::CodeDeploy::Application`

  - `AWS::CodeDeploy::DeploymentGroup`

  - The `AWS::IAM::Role` named `CodeDeployServiceRole` that is created for this scenario

- If your AWS SAM template contains multiple functions with API event sources that are implicitly created, then the functions will share the generated `AWS::ApiGateway::RestApi` resource. In this scenario, if the functions have different resource attributes, then for the generated `AWS::ApiGateway::RestApi` resource, AWS SAM copies the resource attributes according to the following prioritized lists:

  - `UpdateReplacePolicy`:

    1. `Retain`

    2. `Snapshot`

    3. `Delete`

  - `DeletionPolicy`:

    1. `Retain`

    2. `Delete`

Intrinsic functions

Intrinsic functions are built-in functions that enable you to assign values to properties that are only available at runtime. AWS SAM has limited support for certain intrinsic function properties, so it is unable to resolve some intrinsic functions. Consequently, we recommend adding the `AWS::LanguageExtensions` transform to resolve this:

```
Transform:
  - AWS::LanguageExtensions
  - AWS::Serverless-2016-10-31
```

> **ⓘ Note**
>
> Note: If you use intrinsic functions in CodeUri property, AWS SAM will not be able to correctly parse the values. Consider using `AWS::LanguageExtensions` transform instead.

For more information, refer to [Properties section of AWS::Serverless::Function](#).

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 one or more 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.

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 scenarios

The following table summarizes the AWS SAM resources and properties that make up the scenarios that generate AWS CloudFormation resources. The topics in the **Scenarios** column provide details about the additional AWS CloudFormation resources that AWS SAM generates for that scenario.

AWS SAM resource	Base AWS CloudFormation resource	Scenarios
[`AWS::Serverless::Api`](#)	[`AWS::ApiGateway::RestApi`](#)	• [DomainName property is specified](#) • [UsagePlan property is specified](#)

AWS SAM resource	Base AWS CloudFormation resource	Scenarios
[AWS::Serverless::Application](#)	[AWS::CloudFormation::Stack](#)	• Other than generating the base AWS CloudFormation resource, there are no additional scenarios for this serverless resource.
[AWS::Serverless::Function](#)	[AWS::Lambda::Function](#)	• [AutoPublishAlias property is specified](#) • [Role property is not specified](#) • [DeploymentPreference property is specified](#) • [An Api event source is specified](#) • [An HttpApi event source is specified](#) • [A streaming event source is specified](#) • [An event bridge (or event bus) event source is specified](#) • [An IotRule event source is specified](#) • [OnSuccess (or OnFailure) property is specified for Amazon SNS events](#) • [OnSuccess (or OnFailure) property is specified for Amazon SQS events](#)
[AWS::Serverless::HttpApi](#)	[AWS::ApiGatewayV2::Api](#)	• [StageName property is specified](#) • [StageName property is *not* specified](#) • [DomainName property is specified](#)

AWS SAM resource	Base AWS CloudFormation resource	Scenarios
AWS::Serverless::LayerVersion	AWS::Lambda::LayerVersion	• Other than generating the base AWS CloudFormation resource, there are no additional scenarios for this serverless resource.
AWS::Serverless::SimpleTable	AWS::DynamoDB::Table	• Other than generating the base AWS CloudFormation resource, there are no additional scenarios for this serverless resource.
AWS::Serverless::StateMachine	AWS::StepFunctions::StateMachine	• Role property is not specified • An Api event source is specified • An event bridge (or event bus) event source is specified

Topics

- [AWS CloudFormation resources generated when AWS::Serverless::Api is specified](#)
- [AWS CloudFormation resources generated when AWS::Serverless::Application is specified](#)
- [AWS CloudFormation resources generated when you specify AWS::Serverless::Connector](#)
- [AWS CloudFormation resources generated when AWS::Serverless::Function is specified](#)
- [AWS CloudFormation resources generated when AWS::Serverless::GraphQLApi is specified](#)
- [AWS CloudFormation resources generated when AWS::Serverless::HttpApi is specified](#)
- [AWS CloudFormation resources generated when AWS::Serverless::LayerVersion is specified](#)
- [AWS CloudFormation resources generated when AWS::Serverless::SimpleTable is specified](#)
- [AWS CloudFormation resources generated when AWS::Serverless::StateMachine is specified](#)

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 an `AWS::ApiGateway::RestApi` base AWS CloudFormation resource. In addition, it also always generates an `AWS::ApiGateway::Stage` and an `AWS::ApiGateway::Deployment` resource.

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 *`MyRestApiGamma`*`Stage`.

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, *`MyRestApi`*`Deployment`*`926eeb5ff1`*.

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](#)
- [UsagePlan property is specified](#)

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: `ApiGatewayDomainName`*926eeb5ff1*.

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::Application is specified

When an `AWS::Serverless::Application` is specified, AWS Serverless Application Model (AWS SAM) generates an `AWS::CloudFormation::Stack` base AWS CloudFormation resource.

`AWS::CloudFormation::Stack`

LogicalId: `<application-LogicalId>`

Referenceable property: N/A (you must use the `LogicalId` to reference this AWS CloudFormation resource)

AWS CloudFormation resources generated when you specify AWS::Serverless::Connector

> **ⓘ Note**
>
> When you define connectors through the embedded `Connectors` property, it is first transformed into an `AWS::Serverless::Connector` resource before generating these resources.

When you specify an `AWS::Serverless::Connector` resource in an AWS SAM template, AWS SAM generates the following AWS CloudFormation resources as needed.

`AWS::IAM::ManagedPolicy`

LogicalId:`<connector-LogicalId>``Policy`

Referenceable property: N/A (To reference this AWS CloudFormation resource, you must use the `LogicalId`.)

`AWS::SNS::TopicPolicy`

LogicalId:`<connector-LogicalId>``TopicPolicy`

Referenceable property: N/A (To reference this AWS CloudFormation resource, you must use the `LogicalId`.)

AWS::SQS::QueuePolicy

*LogicalId:**<connector-LogicalId>*QueuePolicy

Referenceable property: N/A (To reference this AWS CloudFormation resource, you must use the LogicalId.)

AWS::Lambda::Permission

*LogicalId:**<connector-LogicalId><permission>*LambdaPermission

<permission> is a permission specified by the `Permissions` property. For example, `Write`.

Referenceable property: N/A (To reference this AWS CloudFormation resource, you must use the LogicalId.)

AWS CloudFormation resources generated when AWS::Serverless::Function is specified

When an `AWS::Serverless::Function` is specified, AWS Serverless Application Model (AWS SAM) always creates an `AWS::Lambda::Function` base 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](#)
- [Role property is not specified](#)
- [DeploymentPreference property is specified](#)
- [An Api event source is specified](#)
- [An HttpApi event source is specified](#)
- [A streaming event source is specified](#)

- [An event bridge (or event bus) event source is specified](#)
- [An IotRule event source is specified](#)
- [OnSuccess (or OnFailure) property is specified for Amazon SNS events](#)
- [OnSuccess (or OnFailure) property is specified for Amazon SQS events](#)

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: *MyFunction*Alias*live*.

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, *MyFunction*Version*926eeb5ff1*.

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 an `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)

DeploymentPreference property is specified

When the `DeploymentPreference` property of an `AWS::Serverless::Function` is specified, AWS SAM generates the following resources AWS CloudFormation resources: `AWS::CodeDeploy::Application` and `AWS::CodeDeploy::DeploymentGroup`. In addition, if the `Role` property of the `DeploymentPreference` object is *not* specified, AWS SAM also generates an `AWS::IAM::Role` AWS CloudFormation resource.

AWS::CodeDeploy::Application

LogicalId: `ServerlessDeploymentApplication`

Referenceable property: N/A (you must use the `LogicalId` to reference this AWS CloudFormation resource)

AWS::CodeDeploy::DeploymentGroup

LogicalId: *`<function-LogicalId>`*`DeploymentGroup`

Referenceable property: N/A (you must use the `LogicalId` to reference this AWS CloudFormation resource)

AWS::IAM::Role

LogicalId: `CodeDeployServiceRole`

Referenceable property: N/A (you must use the `LogicalId` to reference this AWS CloudFormation resource)

An Api event source is specified

When the `Event` property of an `AWS::Serverless::Function` is set to `Api`, but the `RestApiId` property is *not* specified, AWS SAM generates the `AWS::ApiGateway::RestApi` AWS CloudFormation resource.

AWS::ApiGateway::RestApi

LogicalId: `ServerlessRestApi`

Referenceable property: N/A (you must use the `LogicalId` to reference this AWS CloudFormation resource)

An HttpApi event source is specified

When the `Event` property of an `AWS::Serverless::Function` is set to `HttpApi`, but the `ApiId` property is *not* specified, AWS SAM generates the `AWS::ApiGatewayV2::Api` AWS CloudFormation resource.

`AWS::ApiGatewayV2::Api`

LogicalId: `ServerlessHttpApi`

Referenceable property: N/A (you must use the `LogicalId` to reference this AWS CloudFormation resource)

A streaming event source is specified

When the `Event` property of an `AWS::Serverless::Function` is set to one of the streaming types, AWS SAM generates the `AWS::Lambda::EventSourceMapping` AWS CloudFormation resource. This applies to the following types: `DynamoDB`, `Kinesis`, `MQ`, `MSK`, and `SQS`.

`AWS::Lambda::EventSourceMapping`

LogicalId: *`<function-LogicalId><event-LogicalId>`*

Referenceable property: N/A (you must use the `LogicalId` to reference this AWS CloudFormation resource)

An event bridge (or event bus) event source is specified

When the `Event` property of an `AWS::Serverless::Function` is set to one of the event bridge (or event bus) types, AWS SAM generates the `AWS::Events::Rule` AWS CloudFormation resource. This applies to the following types: `EventBridgeRule`, `Schedule`, and `CloudWatchEvents`.

`AWS::Events::Rule`

LogicalId: *`<function-LogicalId><event-LogicalId>`*

Referenceable property: N/A (you must use the `LogicalId` to reference this AWS CloudFormation resource)

An IotRule event source is specified

When the `Event` property of an `AWS::Serverless::Function` is set to IoTRule, AWS SAM generates the `AWS::IoT::TopicRule` AWS CloudFormation resource.

AWS::IoT::TopicRule

LogicalId: <function-LogicalId><event-LogicalId>

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::GraphQLApi is specified

When you specify an `AWS::Serverless::GraphQLApi` resource in an AWS Serverless Application Model (AWS SAM) template, AWS SAM always creates the following base AWS CloudFormation resources.

`AWS::AppSync::DataSource`

LogicalId: *`<graphqlapi-LogicalId><datasource-RelativeId><datasource-Type>`*`DataSource`

Referenceable property: N/A (you must use the `LogicalId` to reference this AWS CloudFormation resource)

`AWS::AppSync::FunctionConfiguration`

LogicalId: *`<graphqlapi-LogicalId><function-RelativeId>`*

Referenceable property: N/A (you must use the `LogicalId` to reference this AWS CloudFormation resource)

AWS::AppSync::GraphQLApi

LogicalId: <graphqlapi-LogicalId>

Referenceable property: N/A (you must use the `LogicalId` to reference this AWS CloudFormation resource)

AWS::AppSync::GraphQLSchema

*LogicalId: <graphqlapi-LogicalId>*Schema

Referenceable property: N/A (you must use the `LogicalId` to reference this AWS CloudFormation resource)

AWS::AppSync::Resolver

LogicalId: <graphqlapi-LogicalId><OperationType><resolver-RelativeId>

Referenceable property: N/A (you must use the `LogicalId` to reference this AWS CloudFormation resource)

In addition to these AWS CloudFormation resources, when `AWS::Serverless::GraphQLApi` is specified, AWS SAM may also generate the following AWS CloudFormation resources.

`AWS::AppSync::ApiCache`

*LogicalId: <graphqlapi-LogicalId>*ApiCache

Referenceable property: N/A (you must use the `LogicalId` to reference this AWS CloudFormation resource)

`AWS::AppSync::ApiKey`

LogicalId: <graphqlapi-LogicalId><apikey-RelativeId>

Referenceable property: N/A (you must use the `LogicalId` to reference this AWS CloudFormation resource)

`AWS::AppSync::DomainName`

*LogicalId: <graphqlapi-LogicalId>*DomainName

Referenceable property: N/A (you must use the `LogicalId` to reference this AWS CloudFormation resource)

AWS::AppSync::DomainNameApiAssociation

*LogicalId: <graphqlapi-LogicalId>*DomainNameApiAssociation

Referenceable property: N/A (you must use the LogicalId to reference this AWS CloudFormation resource)

AWS SAM may also use the AWS::Serverless::Connector resource to provision permissions. For more information, see [AWS CloudFormation resources generated when you specify AWS::Serverless::Connector](#).

AWS CloudFormation resources generated when AWS::Serverless::HttpApi is specified

When an AWS::Serverless::HttpApi is specified, AWS Serverless Application Model (AWS SAM) generates an AWS::ApiGatewayV2::Api base 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](#)
- [StageName property is not specified](#)
- [DomainName property is specified](#)

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: *MyHttpApiGamma*Stage.

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, ApiGatewayDomainNameV2*926eeb5ff1*.

Referenceable property: *<httpapi-LogicalId>*.DomainName

AWS CloudFormation resources generated when AWS::Serverless::LayerVersion is specified

When an AWS::Serverless::LayerVersion is specified, AWS Serverless Application Model (AWS SAM) generates an AWS::Lambda::LayerVersion base AWS CloudFormation resource.

AWS::Lambda::LayerVersion

LogicalId: *<layerversion-LogicalId>*

Referenceable property: N/A (you must use the `LogicalId` to reference this AWS CloudFormation resource)

AWS CloudFormation resources generated when AWS::Serverless::SimpleTable is specified

When an `AWS::Serverless::SimpleTable` is specified, AWS Serverless Application Model (AWS SAM) generates an `AWS::DynamoDB::Table` base AWS CloudFormation resource.

AWS::DynamoDB::Table

LogicalId: *<simpletable-LogicalId>*

Referenceable property: N/A (you must use the `LogicalId` to reference this AWS CloudFormation resource)

AWS CloudFormation resources generated when AWS::Serverless::StateMachine is specified

When an `AWS::Serverless::StateMachine` is specified, AWS Serverless Application Model (AWS SAM) generates an `AWS::StepFunctions::StateMachine` base AWS CloudFormation resource.

AWS::StepFunctions::StateMachine

LogicalId: *<statemachine-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::StateMachine` is specified, AWS SAM also generates AWS CloudFormation resources for the following scenarios:

Scenarios

- [Role property is not specified](#)
- [An Api event source is specified](#)
- [An event bridge (or event bus) event source is specified](#)

Role property is not specified

When the `Role` property of an `AWS::Serverless::StateMachine` is *not* specified, AWS SAM generates an `AWS::IAM::Role` AWS CloudFormation resource.

AWS::IAM::Role

LogicalId: *<statemachine-LogicalId>*Role

Referenceable property: N/A (you must use the `LogicalId` to reference this AWS CloudFormation resource)

An Api event source is specified

When the `Event` property of an `AWS::Serverless::StateMachine` is set to `Api`, but the `RestApiId` property is *not* specified, AWS SAM generates the `AWS::ApiGateway::RestApi` AWS CloudFormation resource.

AWS::ApiGateway::RestApi

LogicalId: `ServerlessRestApi`

Referenceable property: N/A (you must use the `LogicalId` to reference this AWS CloudFormation resource)

An event bridge (or event bus) event source is specified

When the `Event` property of an `AWS::Serverless::StateMachine` is set to one of the event bridge (or event bus) types, AWS SAM generates the `AWS::Events::Rule` AWS CloudFormation resource. This applies to the following types: `EventBridgeRule`, `Schedule`, and `CloudWatchEvents`.

AWS::Events::Rule

LogicalId: *<statemachine-LogicalId><event-LogicalId>*

Referenceable property: N/A (you must use the `LogicalId` to reference this AWS CloudFormation resource)

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.

API Gateway Extension	Supported by AWS SAM
x-amazon-apigateway-any-method Object	Yes
x-amazon-apigateway-api-key-source Property	No
x-amazon-apigateway-auth Object	Yes
x-amazon-apigateway-authorizer Object	Yes
x-amazon-apigateway-authtype Property	Yes
x-amazon-apigateway-binary-media-types Property	Yes
x-amazon-apigateway-documentation Object	No
x-amazon-apigateway-endpoint-configuration Object	No
x-amazon-apigateway-gateway-responses Object	Yes
x-amazon-apigateway-gateway-responses.gatewayResponse Object	Yes
x-amazon-apigateway-gateway-responses.responseParameters Object	Yes
x-amazon-apigateway-gateway-responses.responseTemplates Object	Yes
x-amazon-apigateway-integration Object	Yes

x-amazon-apigateway-integration.requestTemplates Object	Yes
x-amazon-apigateway-integration.requestParameters Object	No
x-amazon-apigateway-integration.responses Object	Yes
x-amazon-apigateway-integration.response Object	Yes
x-amazon-apigateway-integration.responseTemplates Object	Yes
x-amazon-apigateway-integration.responseParameters Object	Yes
x-amazon-apigateway-request-validator Property	No
x-amazon-apigateway-request-validators Object	No
x-amazon-apigateway-request-validators.requestValidator Object	No

Authoring serverless applications

When you author a serverless application using AWS SAM, you construct an AWS SAM template to declare and configure the components of your application.

This section contains topics about validating your AWS SAM template and building your application with dependencies. It also contains topics about using AWS SAM for certain use cases such as working with Lambda layers, using nested applications, controlling access to API Gateway APIs, orchestrating AWS resources with Step Functions, and code signing your applications.

Topics

- [Managing resource access and permissions](#)
- [Scheduling events with EventBridge Scheduler](#)
- [Validating AWS SAM template files](#)
- [Working with layers](#)
- [Using nested applications](#)
- [Controlling access to API Gateway APIs](#)
- [Orchestrating AWS resources with AWS Step Functions](#)
- [Configuring code signing for AWS SAM applications](#)

Managing resource access and permissions

For your AWS resources to interact with one another, the proper access and permissions must be configured between your resources, requiring the configuration of AWS Identity and Access Management (IAM) users, roles, and policies to accomplish your interaction in a secure manner. To learn more, see [Controlling access with AWS Identity and Access Management](#) in the *AWS CloudFormation User Guide.*

The AWS Serverless Application Model (AWS SAM) provides two options that simplify management of access and permissions for your serverless applcations.

1. AWS SAM connectors
2. AWS SAM policy templates

AWS SAM connectors

Connectors are a way of provisioning permissions between two resources. You do this by describing how they should interact with each other in your AWS SAM template. They can be defined using either the `Connectors` resource attribute or `AWS::Serverless::Connector` resource type. Connectors support the provisioning of `Read` and `Write` access of data and events between a combination of AWS resources. To learn more about AWS SAM connectors, see [Managing resource permissions with AWS SAM connectors](#).

AWS SAM policy templates

AWS SAM policy templates are pre-defined sets of permissions that you can add to your AWS SAM templates to manage access and permissions between your AWS Lambda functions, AWS Step Functions state machines and the resources they interact with. To learn more about AWS SAM policy templates, see [AWS SAM policy templates](#).

AWS CloudFormation mechanisms

AWS CloudFormation mechanisms include the configuring of IAM users, roles, and policies to manage permissions between your AWS resources. To learn more, see [Managing permissions with AWS CloudFormation mechanisms](#).

Best practices

Throughout your serverless applications, you can use multiple methods to configure permissions between your resources. Therefore, you can select the best option for each scenario and use multiple options together throughout your applications. Here are a few things to consider when choosing the best option for you:

- AWS SAM connectors and policy templates both reduce the IAM expertise required to facilitate secure interactions between your AWS resources. Use connectors and policy templates when supported.

- AWS SAM connectors provide a simple and intuitive short-hand syntax to define permissions in your AWS SAM templates and require the least amount of IAM expertise. When both AWS SAM connectors and policy templates are supported, use AWS SAM connectors.

- AWS SAM connectors can provision `Read` and `Write` access of data and events between supported AWS SAM source and destination resources. For a list of supported resources, see [AWS SAM connector reference](#). When supported, use AWS SAM connectors.

- While AWS SAM policy templates are limited to permissions between your Lambda functions, Step Functions state machines and the AWS resources they interact with, policy templates do support all CRUD operations. When supported, and when an AWS SAM policy template for your scenario is available, use AWS SAM policy templates. For a list of available policy templates, see AWS SAM policy templates.

- For all other scenarios, or when granularity is required, use AWS CloudFormation mechanisms.

Managing resource permissions with AWS SAM connectors

Topics

- What are AWS SAM connectors?

- Example of connectors

- Supported connections between source and destination resources

- Using connectors

- How connectors work

- Benefits of AWS SAM connectors

- Learn more

- Provide feedback

What are AWS SAM connectors?

Connectors are an AWS Serverless Application Model (AWS SAM) abstract resource type, identified as `AWS::Serverless::Connector`, that provides simple and well-scoped permissions between your serverless application resources. Use the `Connectors` resource attribute by embedding it within a **source** resource. Then, define your **destination** resource and describe how data or events should flow between those resources. AWS SAM then composes the access policies necessary to facilitate the required interactions.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  <source-resource-logical-id>:
    Type: <resource-type>
    ...
```

```
  Connectors:
    <connector-name>:
      Properties:
        Destination:
          <properties-that-identify-destination-resource>
        Permissions:
          <permission-types-to-provision>
  ...
```

Example of connectors

In this example, we use connectors to write data from an AWS Lambda function to an Amazon DynamoDB table.
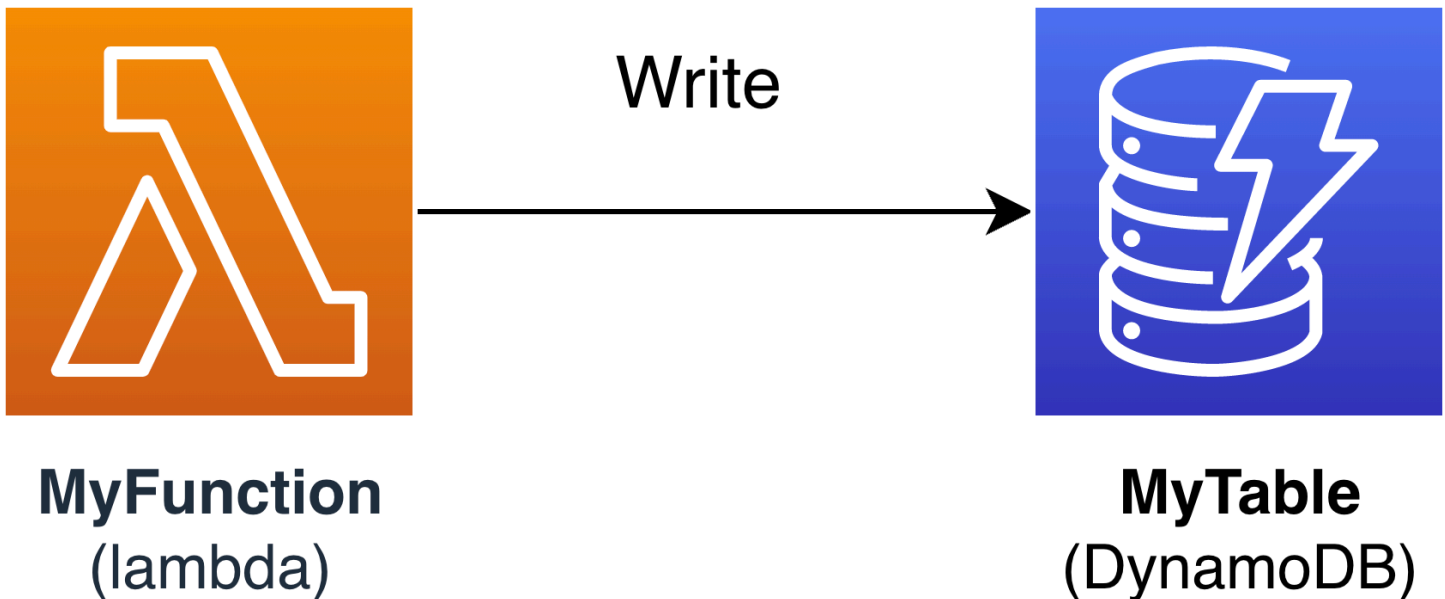
MyFunction
(lambda)

Write →

MyTable
(DynamoDB)

```
Transform: AWS::Serverless-2016-10-31
Resources:
  MyTable:
    Type: AWS::Serverless::SimpleTable
  MyFunction:
    Type: AWS::Serverless::Function
    Connectors:
      MyConn:
        Properties:
          Destination:
            Id: MyTable
          Permissions:
            - Write
    Properties:
```

```
      Runtime: nodejs16.x
      Handler: index.handler
      InlineCode: |
        const AWS = require("aws-sdk");
        const docClient = new AWS.DynamoDB.DocumentClient();
        exports.handler = async (event, context) => {
          await docClient.put({
            TableName: process.env.TABLE_NAME,
            Item: {
              id: context.awsRequestId,
              event: JSON.stringify(event)
            }
          }).promise();
        }
      Environment:
        Variables:
          TABLE_NAME: !Ref MyTable
```

The `Connectors` resource attribute is embedded within the Lambda function source resource. The DynamoDB table is defined as the destination resource using the `Id` property. Connectors will provision `Write` permissions between these two resources.

When you deploy your AWS SAM template to AWS CloudFormation, AWS SAM will automatically compose the necessary access policies required for this connection to work.

Supported connections between source and destination resources

Connectors support `Read` and `Write` data and event permission types between a select combination of source and destination resource connections. For example, connectors support a `Write` connection between an `AWS::ApiGateway::RestApi` source resource and an `AWS::Lambda::Function` destination resource.

Source and destination resources can be defined by using a combination of supported properties. Property requirements will depend on the connection you are making and where the resources are defined.

> ⓘ **Note**
>
> Connectors can provision permissions between supported serverless and non-serverless resource types.

For a list of supported resource connections and their property requirements, see Supported source and destination resource types for connectors.

Using connectors

Define Read and Write permissions

Read and `Write` permissions can be provisioned within a single connector:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Lambda::Function
    Connectors:
      MyTableConn:
        Properties:
          Destination:
            Id: MyTable
          Permissions:
            - Read
            - Write
  MyTable:
    Type: AWS::DynamoDB::Table
```

Define resources by using other supported properties

For both source and destination resources, when defined within the same template, use the `Id` property. Optionally, a `Qualifier` can be added to narrow the scope of your defined resource. When the resource is not within the same template, use a combination of supported properties.

- For a list of supported property combinations for source and destination resources, see Supported source and destination resource types for connectors.
- For a description of properties that you can use with connectors, see AWS::Serverless::Connector.

When you define a source resource with a property other than `Id`, use the `SourceReference` property.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
```

```
...
Resources:
  <source-resource-logical-id>:
    Type: <resource-type>
    ...
    Connectors:
      <connector-name>:
        Properties:
          SourceReference:
            Qualifier: <optional-qualifier>
            <other-supported-properties>
          Destination:
            <properties-that-identify-destination-resource>
          Permissions:
            <permission-types-to-provision>
```

Here's an example, using a `Qualifier` to narrow the scope of an Amazon API Gateway resource:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyApi:
    Type: AWS::Serverless::Api
    Connectors:
      ApiToLambdaConn:
        Properties:
          SourceReference:
            Qualifier: Prod/GET/foobar
          Destination:
            Id: MyFunction
          Permissions:
            - Write
  ...
```

Here's an example, using a supported combination of `Arn` and `Type` to define a destination resource from another template:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
```

```
      Type: AWS::Serverless::Function
    Connectors:
      TableConn:
        Properties:
          Destination:
            Type: AWS::DynamoDB::Table
            Arn: !GetAtt MyTable.Arn
  ...
```

Create multiple connectors from a single source

Within a source resource, you can define multiple connectors, each with a different destination resource.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Connectors:
      BucketConn:
        Properties:
          Destination:
            Id: MyBucket
          Permissions:
            - Read
            - Write
      SQSConn:
        Properties:
          Destination:
            Id: MyQueue
          Permissions:
            - Read
            - Write
      TableConn:
        Properties:
          Destination:
            Id: MyTable
          Permissions:
            - Read
            - Write
      TableConnWithTableArn:
```

```
      Properties:
        Destination:
          Type: AWS::DynamoDB::Table
          Arn: !GetAtt MyTable.Arn
        Permissions:
          - Read
          - Write
...
```

Create multi-destination connectors

Within a source resource, you can define a single connector with multiple destination resources. Here's an example of a Lambda function source resource connecting to an Amazon Simple Storage Service (Amazon S3) bucket and DynamoDB table:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Connectors:
      WriteAccessConn:
        Properties:
          Destination:
            - Id: OutputBucket
            - Id: CredentialTable
          Permissions:
            - Write
  ...
  OutputBucket:
    Type: AWS::S3::Bucket
  CredentialTable:
    Type: AWS::DynamoDB::Table
```

Define resource attributes with connectors

Resource attributes can be defined for resources to specify additional behaviors and relationships. To learn more about resource attributes, see Resource attribute reference in the *AWS CloudFormation User Guide*.

You can add resource attributes to your embedded connector by defining them on the same level as your connector properties. When your AWS SAM template is transformed at deployment, attributes will pass through to the generated resources.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Connectors:
      MyConn:
        DeletionPolicy: Retain
        DependsOn: AnotherFunction
        Properties:
          ...
```

How connectors work

> **ⓘ Note**
>
> This section explains how connectors provision the necessary resources behind the scenes. This happens for you automatically when using connectors.

First, the embedded `Connectors` resource attribute is transformed into an `AWS::Serverless::Connector` resource type. Its logical ID is automatically created as *<source-resource-logical-id><embedded-connector-logical-id>*.

For example, here is an embedded connector:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Lambda::Function
    Connectors:
      MyConn:
        Properties:
          Destination:
```

```
            Id: MyTable
        Permissions:
          - Read
          - Write
    MyTable:
      Type: AWS::DynamoDB::Table
```

This will generate the following `AWS::Serverless::Connector` resource:

```
Transform: AWS::Serverless-2016-10-31
Resources:
  ...
  MyFunctionMyConn:
    Type: AWS::Serverless::Connector
    Properties:
      Source:
        Id: MyFunction
      Destination:
        Id: MyTable
      Permissions:
        - Read
        - Write
```

> ⓘ **Note**
>
> You can also define connectors in your AWS SAM template by using this syntax. This is
> recommended when your source resource is defined on a separate template from your
> connector.

Next, the necessary access policies for this connection are automatically composed. For more
information about the resources generated by connectors, see AWS CloudFormation resources
generated when you specify AWS::Serverless::Connector.

Benefits of AWS SAM connectors

By automatically composing the appropriate access policies between resources, connectors provide
you the ability to author your serverless applications and focus on your application architecture
without needing expertise in AWS authorization capabilities, policy language, and service-specific
security settings. Therefore, connectors are a great benefit to developers who may be new to
serverless development, or seasoned developers looking to increase their development velocity.

Learn more

For more information about using AWS SAM connectors, see [AWS::Serverless::Connector](#).

Provide feedback

To provide feedback on connectors, [submit a new issue](#) at the *serverless-application-model AWS GitHub repository*.

AWS SAM policy templates

The AWS Serverless Application Model (AWS SAM) allows you to choose from a list of policy templates to scope the permissions of your Lambda functions and AWS Step Functions state machines to the resources that are used by your application.

AWS SAM applications in the AWS Serverless Application Repository that use policy templates don't require any special customer acknowledgments to deploy the application from the AWS Serverless Application Repository.

If you want to request a new policy template to be added, do the following:

1. Submit a pull request against the policy_templates.json source file in the `develop` branch of the AWS SAM GitHub project. You can find the source file in [policy_templates.json](#) on the GitHub website.

2. Submit an issue in the AWS SAM GitHub project that includes the reasons for your pull request and a link to the request. Use this link to submit a new issue: [AWS Serverless Application Model: Issues](#).

Syntax

For every policy template you specify in your AWS SAM template file, you must always specify an object containing the policy template's placeholder values. If a policy template does not require any placeholder values, you must specify an empty object.

YAML

```
MyFunction:
  Type: AWS::Serverless::Function
  Properties:
```

```
    Policies:
      - PolicyTemplateName1:        # Policy template with placeholder value
          Key1: Value1
      - PolicyTemplateName2: {}     # Policy template with no placeholder value
```

Examples

Example 1: Policy template with placeholder values

The following example shows that the SQSPollerPolicy 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.

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

Example 2: Policy template with no placeholder values

The following example contains the CloudWatchPutMetricPolicy 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.

```
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.

Policy Template	Description		
AcmGetCertificatePolicy	Gives a permission to read a certificate from AWS Certificate Manager.		
AMIDescribePolicy	Gives permission to describe Amazon Machine Images (AMIs).		
AthenaQueryPolicy	Gives permissions to execute Athena queries.		
AWSSecretsManagerGetSecretValuePolicy	Gives permission to get the secret value for the specified AWS Secrets Manager secret.		
AWSSecretsManagerRotationPolicy	Gives permission to rotate a secret in AWS Secrets Manager.		
CloudFormationDescribeStacksPolicy	Gives permission to describe AWS CloudFormation stacks.		
CloudWatchDashboardPolicy	Gives permissions to put metrics to operate on CloudWatch dashboards.		

Policy Template	Description		
CloudWatchDescribeAlarmHistoryPolicy	Gives permission to describe CloudWatch alarm history.		
CloudWatchPutMetricPolicy	Gives permission to send metrics to CloudWatch.		
CodeCommitCrudPolicy	Gives permissions to create/read/ update/delete objects within a specific CodeCommit repository.		
CodeCommitReadPolicy	Gives permissions to read objects within a specific CodeCommit repository.		
CodePipelineLambdaExecutionPolicy	Gives permission for a Lambda function invoked by CodePipeline to report the status of the job.		
CodePipelineReadOnlyPolicy	Gives read permission to get details about a CodePipeline pipeline.		
ComprehendBasicAccessPolicy	Gives permission for detecting entities, key phrases, languages, and sentiments.		
CostExplorerReadOnlyPolicy	Gives read-only permission to the read-only Cost Explorer APIs for billing history.		
DynamoDBBackupFullAccessPolicy	Gives read and write permission to DynamoDB on-demand backups for a table.		

Policy Template	Description		
DynamoDBCrudPolicy	Gives create, read, update, and delete permissions to an Amazon DynamoDB table.		
DynamoDBReadPolicy	Gives read-only permission to a DynamoDB table.		
DynamoDBReconfigurePolicy	Gives permission to reconfigure a DynamoDB table.		
DynamoDBRestoreFromBackupPolicy	Gives permission to restore a DynamoDB table from backup.		
DynamoDBStreamReadPolicy	Gives permission to describe and read DynamoDB streams and records.		
DynamoDBWritePolicy	Gives write-only permission to a DynamoDB table.		
EC2CopyImagePolicy	Gives permission to copy Amazon EC2 images.		
EC2DescribePolicy	Gives permission to describe Amazon Elastic Compute Cloud (Amazon EC2) instances.		
EcsRunTaskPolicy	Gives permission to start a new task for a task definition.		
EFSWriteAccessPolicy	Gives permission to mount an Amazon EFS file system with write access.		

Policy Template	Description		
EKSDescribePolicy	Gives permission to describe or list Amazon EKS clusters.		
ElasticMapReduceAddJobFlowStepsPolicy	Gives permission to add new steps to a running cluster.		
ElasticMapReduceCancelStepsPolicy	Gives permission to cancel a pending step or steps in a running cluster.		
ElasticMapReduceModifyInstanceFleetPolicy	Gives permission to list details and modify capacities for instance fleets within a cluster.		
ElasticMapReduceModifyInstanceGroupsPolicy	Gives permission to list details and modify settings for instance groups within a cluster.		
ElasticMapReduceSetTerminationProtectionPolicy	Gives permission to set termination protection for a cluster.		
ElasticMapReduceTerminateJobFlowsPolicy	Gives permission to shut down a cluster.		

Policy Template	Description		
ElasticsearchHttpPostPolicy	Gives POST permission to Amazon OpenSearch Service.		
EventBridgePutEventsPolicy	Gives permissions to send events to EventBridge.		
FilterLogEventsPolicy	Gives permission to filter CloudWatch Logs events from a specified log group.		
FirehoseCrudPolicy	Gives permission to create, write, update, and delete a Firehose delivery stream.		
FirehoseWritePolicy	Gives permission to write to a Firehose delivery stream.		
KinesisCrudPolicy	Gives permission to create, publish, and delete an Amazon Kinesis stream.		
KinesisStreamReadPolicy	Gives permission to list and read an Amazon Kinesis stream.		
KMSDecryptPolicy	Gives permission to decrypt with an AWS Key Management Service (AWS KMS) key.		
KMSEncryptPolicy	Gives permission to encrypt with an AWS Key Management Service (AWS KMS) key.		
LambdaInvokePolicy	Gives permission to invoke an AWS Lambda function, alias, or version.		

Policy Template	Description		
MobileAnalyticsWriteOnlyAccessPolicy	Gives write-only permission to put event data for all application resources.		
OrganizationsListAccountsPolicy	Gives read-only permission to list child account names and IDs.		
PinpointEndpointAccessPolicy	Gives permission to get and update endpoints for an Amazon Pinpoint application.		
PollyFullAccessPolicy	Gives full access permission to Amazon Polly lexicon resources.		
RekognitionDetectOnlyPolicy	Gives permission to detect faces, labels, and text.		
RekognitionFacesManagementPolicy	Gives permission to add, delete, and search faces in an Amazon Rekognition collection.		
RekognitionFacesPolicy	Gives permission to compare and detect faces and labels.		
RekognitionLabelsPolicy	Gives permission to detect object and moderation labels.		
RekognitionNoDataAccessPolicy	Gives permission to compare and detect faces and labels.		

Policy Template	Description		
RekognitionReadPolicy	Gives permission to list and search faces.		
RekognitionWriteOnlyAccessPolicy	Gives permission to create collection and index faces.		
Route53ChangeResourceRecordSetsPolicy	Gives permission to change resource record sets in Route 53.		
S3CrudPolicy	Gives create, read, update, and delete permission to act on the objects in an Amazon S3 bucket.		
S3FullAccessPolicy	Gives full access permission to act on the objects in an Amazon S3 bucket.		
S3ReadPolicy	Gives read-only permission to read objects in an Amazon Simple Storage Service (Amazon S3) bucket.		
S3WritePolicy	Gives write permission to write objects into an Amazon S3 bucket.		
SageMakerCreateEndpointConfigPolicy	Gives permission to create an endpoint configuration in SageMaker.		
SageMakerCreateEndpointPolicy	Gives permission to create an endpoint in SageMaker.		

Policy Template	Description		
ServerlessRepoReadWriteAccessPolicy	Gives permission to create and list applications in the AWS Serverless Application Repository service.		
SESBulkTemplatedCrudPolicy	Gives permission to send email, templated email, templated bulk emails and verify identity.		
SESBulkTemplatedCrudPolicy_v2	Gives permission to send Amazon SES email, templated email, and templated bulk emails and to verify identity.		
SESCrudPolicy	Gives permission to send email and verify identity.		
SESEmailTemplateCrudPolicy	Gives permission to create, get, list, update and delete Amazon SES email templates.		
SESSendBouncePolicy	Gives SendBounce permission to an Amazon Simple Email Service (Amazon SES) identity.		
SNSCrudPolicy	Gives permission to create, publish, and subscribe to Amazon SNS topics.		
SNSPublishMessagePolicy	Gives permission to publish a message to an Amazon Simple Notification Service (Amazon SNS) topic.		
SQSPollerPolicy	Gives permission to poll an Amazon Simple Queue Service (Amazon SQS) queue.		

Policy Template	Description		
SQSSendMessagePolicy	Gives permission to send message to an Amazon SQS queue.		
SSMParameterReadPolicy	Gives permission to access a parameter from an Amazon EC2 Systems Manager (SSM) parameter store to load secrets in this account. Use when parameter name doesn't have slash prefix.		
SSMParameterWithSlashPrefixReadPolicy	Gives permission to access a parameter from an Amazon EC2 Systems Manager (SSM) parameter store to load secrets in this account. Use when parameter name has slash prefix.		
StepFunctionsExecutionPolicy	Gives permission to start a Step Functions state machine execution.		
TextractDetectAnalyzePolicy	Gives access to detect and analyze documents with Amazon Textract.		
TextractGetResultPolicy	Gives access to get detected and analyzed documents from Amazon Textract.		
TextractPolicy	Gives full access to Amazon Textract.		
VPCAccessPolicy	Gives access to create, delete, describe, and detach elastic network interfaces.		

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.

```
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.

Topics

- AcmGetCertificatePolicy
- AMIDescribePolicy
- AthenaQueryPolicy
- AWSSecretsManagerGetSecretValuePolicy
- AWSSecretsManagerRotationPolicy
- CloudFormationDescribeStacksPolicy
- CloudWatchDashboardPolicy
- CloudWatchDescribeAlarmHistoryPolicy
- CloudWatchPutMetricPolicy
- CodePipelineLambdaExecutionPolicy

- [CodePipelineReadOnlyPolicy](#)
- [CodeCommitCrudPolicy](#)
- [CodeCommitReadPolicy](#)
- [ComprehendBasicAccessPolicy](#)
- [CostExplorerReadOnlyPolicy](#)
- [DynamoDBBackupFullAccessPolicy](#)
- [DynamoDBCrudPolicy](#)
- [DynamoDBReadPolicy](#)
- [DynamoDBReconfigurePolicy](#)
- [DynamoDBRestoreFromBackupPolicy](#)
- [DynamoDBStreamReadPolicy](#)
- [DynamoDBWritePolicy](#)
- [EC2CopyImagePolicy](#)
- [EC2DescribePolicy](#)
- [EcsRunTaskPolicy](#)
- [EFSWriteAccessPolicy](#)
- [EKSDescribePolicy](#)
- [ElasticMapReduceAddJobFlowStepsPolicy](#)
- [ElasticMapReduceCancelStepsPolicy](#)
- [ElasticMapReduceModifyInstanceFleetPolicy](#)
- [ElasticMapReduceModifyInstanceGroupsPolicy](#)
- [ElasticMapReduceSetTerminationProtectionPolicy](#)
- [ElasticMapReduceTerminateJobFlowsPolicy](#)
- [ElasticsearchHttpPostPolicy](#)
- [EventBridgePutEventsPolicy](#)
- [FilterLogEventsPolicy](#)
- [FirehoseCrudPolicy](#)
- [FirehoseWritePolicy](#)
- [KinesisCrudPolicy](#)
- [KinesisStreamReadPolicy](#)

- KMSDecryptPolicy

- KMSEncryptPolicy

- LambdaInvokePolicy

- MobileAnalyticsWriteOnlyAccessPolicy

- OrganizationsListAccountsPolicy

- PinpointEndpointAccessPolicy

- PollyFullAccessPolicy

- RekognitionDetectOnlyPolicy

- RekognitionFacesManagementPolicy

- RekognitionFacesPolicy

- RekognitionLabelsPolicy

- RekognitionNoDataAccessPolicy

- RekognitionReadPolicy

- RekognitionWriteOnlyAccessPolicy

- Route53ChangeResourceRecordSetsPolicy

- S3CrudPolicy

- S3FullAccessPolicy

- S3ReadPolicy

- S3WritePolicy

- SageMakerCreateEndpointConfigPolicy

- SageMakerCreateEndpointPolicy

- ServerlessRepoReadWriteAccessPolicy

- SESBulkTemplatedCrudPolicy

- SESBulkTemplatedCrudPolicy_v2

- SESCrudPolicy

- SESEmailTemplateCrudPolicy

- SESSendBouncePolicy

- SNSCrudPolicy

- SNSPublishMessagePolicy

- SQSPollerPolicy

- [SQSSendMessagePolicy](#)

- [SSMParameterReadPolicy](#)

- [SSMParameterWithSlashPrefixReadPolicy](#)

- [StepFunctionsExecutionPolicy](#)

- [TextractDetectAnalyzePolicy](#)

- [TextractGetResultPolicy](#)

- [TextractPolicy](#)

- [VPCAccessPolicy](#)

AcmGetCertificatePolicy

Gives a permission to read a certificate from AWS Certificate Manager.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "acm:GetCertificate"
    ],
    "Resource": {
      "Fn::Sub": [
        "${certificateArn}",
        {
          "certificateArn": {
            "Ref": "CertificateArn"
          }
        }
      ]
    }
  }
]
```

AMIDescribePolicy

Gives permission to describe Amazon Machine Images (AMIs).

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

```
      "ec2:DescribeImages"
    ],
    "Resource": "*"
  }
]
```

AthenaQueryPolicy

Gives permissions to execute Athena queries.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "athena:ListWorkGroups",
      "athena:GetExecutionEngine",
      "athena:GetExecutionEngines",
      "athena:GetNamespace",
      "athena:GetCatalogs",
      "athena:GetNamespaces",
      "athena:GetTables",
      "athena:GetTable"
    ],
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "athena:StartQueryExecution",
      "athena:GetQueryResults",
      "athena:DeleteNamedQuery",
      "athena:GetNamedQuery",
      "athena:ListQueryExecutions",
      "athena:StopQueryExecution",
      "athena:GetQueryResultsStream",
      "athena:ListNamedQueries",
      "athena:CreateNamedQuery",
      "athena:GetQueryExecution",
      "athena:BatchGetNamedQuery",
      "athena:BatchGetQueryExecution",
      "athena:GetWorkGroup"
    ],
    "Resource": {
      "Fn::Sub": [
```

```
          "arn:${AWS::Partition}:athena:${AWS::Region}:${AWS::AccountId}:workgroup/
${workgroupName}",
        {
          "workgroupName": {
            "Ref": "WorkGroupName"
          }
        }
      ]
    }
  }
]
```

AWSSecretsManagerGetSecretValuePolicy

Gives permission to get the secret value for the specified AWS Secrets Manager secret.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "secretsmanager:GetSecretValue"
    ],
    "Resource": {
      "Fn::Sub": [
        "${secretArn}",
        {
          "secretArn": {
            "Ref": "SecretArn"
          }
        }
      ]
    }
  }
]
```

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": "*"
  }
]
```

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/*"
    }
  }
]
```

CloudWatchDashboardPolicy

Gives permissions to put metrics to operate on CloudWatch dashboards.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "cloudwatch:GetDashboard",
      "cloudwatch:ListDashboards",
      "cloudwatch:PutDashboard",
      "cloudwatch:ListMetrics"
    ],
    "Resource": "*"
  }
]
```

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": "*"
    }
  ]
```

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": "*"
  }
]
```

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"
          }
```

```
            }
        ]
    }
  }
]
```

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:PostCommentForComparedCommit",
      "codecommit:PostCommentReply",
      "codecommit:UpdateComment",
      "codecommit:BatchGetCommits",
      "codecommit:CreateCommit",
      "codecommit:GetCommit",
      "codecommit:GetCommitHistory",
      "codecommit:GetDifferences",
      "codecommit:GetObjectIdentifier",
      "codecommit:GetReferences",
      "codecommit:GetTree",
      "codecommit:GetRepository",
      "codecommit:UpdateRepositoryDescription",
      "codecommit:ListTagsForResource",
      "codecommit:TagResource",
      "codecommit:UntagResource",
      "codecommit:GetRepositoryTriggers",
      "codecommit:PutRepositoryTriggers",
      "codecommit:TestRepositoryTriggers",
      "codecommit:GetBranch",
      "codecommit:GetCommit",
      "codecommit:UploadArchive",
      "codecommit:GetUploadArchiveStatus",
      "codecommit:CancelUploadArchive"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:codecommit:${AWS::Region}:${AWS::AccountId}:
${repositoryName}",
        {
          "repositoryName": {
            "Ref": "RepositoryName"
          }
        }
```

```
      ]
    }
  }
]
```

CodeCommitReadPolicy

Gives permissions to read objects within a specific CodeCommit repository.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "codecommit:GitPull",
      "codecommit:GetBranch",
      "codecommit:ListBranches",
      "codecommit:BatchDescribeMergeConflicts",
      "codecommit:DescribeMergeConflicts",
      "codecommit:GetMergeCommit",
      "codecommit:GetMergeOptions",
      "codecommit:BatchGetPullRequests",
      "codecommit:DescribePullRequestEvents",
      "codecommit:GetCommentsForPullRequest",
      "codecommit:GetCommitsFromMergeBase",
      "codecommit:GetMergeConflicts",
      "codecommit:GetPullRequest",
      "codecommit:ListPullRequests",
      "codecommit:GetBlob",
      "codecommit:GetFile",
      "codecommit:GetFolder",
      "codecommit:GetComment",
      "codecommit:GetCommentsForComparedCommit",
      "codecommit:BatchGetCommits",
      "codecommit:GetCommit",
      "codecommit:GetCommitHistory",
      "codecommit:GetDifferences",
      "codecommit:GetObjectIdentifier",
      "codecommit:GetReferences",
      "codecommit:GetTree",
      "codecommit:GetRepository",
      "codecommit:ListTagsForResource",
      "codecommit:GetRepositoryTriggers",
      "codecommit:TestRepositoryTriggers",
      "codecommit:GetBranch",
```

```
        "codecommit:GetCommit",
        "codecommit:GetUploadArchiveStatus"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:codecommit:${AWS::Region}:${AWS::AccountId}:
${repositoryName}",
        {
          "repositoryName": {
            "Ref": "RepositoryName"
          }
        }
      ]
    }
  }
]
```

ComprehendBasicAccessPolicy

Gives permission for detecting entities, key phrases, languages, and sentiments.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "comprehend:BatchDetectKeyPhrases",
      "comprehend:DetectDominantLanguage",
      "comprehend:DetectEntities",
      "comprehend:BatchDetectEntities",
      "comprehend:DetectKeyPhrases",
      "comprehend:DetectSentiment",
      "comprehend:BatchDetectDominantLanguage",
      "comprehend:BatchDetectSentiment"
    ],
    "Resource": "*"
  }
]
```

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": "*"
  }
]
```

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"
          }
        }
      ]
    }
  }
]
```

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: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"
            }
          }
        ]
      }
    ]
  }
]
```

DynamoDBReconfigurePolicy

Gives permission to reconfigure a DynamoDB table.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "dynamodb:UpdateTable"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
${tableName}",
        {
          "tableName": {
            "Ref": "TableName"
          }
        }
      ]
    }
  }
]
```

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/
${tableName}/backup/*",
        {
          "tableName": {
            "Ref": "TableName"
          }
        }
      ]
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "dynamodb:PutItem",
      "dynamodb:UpdateItem",
      "dynamodb:DeleteItem",
      "dynamodb:GetItem",
      "dynamodb:Query",
      "dynamodb:Scan",
      "dynamodb:BatchWriteItem"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
${tableName}",
        {
          "tableName": {
            "Ref": "TableName"
          }
        }
      ]
    }
  }
]
```

DynamoDBStreamReadPolicy

Gives permission to describe and read DynamoDB streams and records.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "dynamodb:DescribeStream",
      "dynamodb:GetRecords",
      "dynamodb:GetShardIterator"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
${tableName}/stream/${streamName}",
        {
          "tableName": {
            "Ref": "TableName"
          },
          "streamName": {
            "Ref": "StreamName"
          }
        }
      ]
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "dynamodb:ListStreams"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
${tableName}/stream/*",
        {
          "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"
            }
          }
        ]
      }
    ]
  }
]
```

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": {
            "Ref": "ImageId"
          }
        }
      ]
    }
  }
]
```

EC2DescribePolicy

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

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

EcsRunTaskPolicy

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

```
"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.

```
"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"
          }
        }
      ]
    },
    "Condition": {
      "StringEquals": {
        "elasticfilesystem:AccessPointArn": {
```

```
            "Fn::Sub": [
              "arn:${AWS::Partition}:elasticfilesystem:${AWS::Region}:
${AWS::AccountId}:access-point/${AccessPoint}",
              {
                "AccessPoint": {
                  "Ref": "AccessPoint"
                }
              }
            ]
          }
        }
      }
    }
  ]
```

EKSDescribePolicy

Gives permission to describe or list Amazon Elastic Kubernetes Service (Amazon EKS) clusters.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "eks:DescribeCluster",
      "eks:ListClusters"
    ],
    "Resource": "*"
  }
]
```

ElasticMapReduceAddJobFlowStepsPolicy

Gives permission to add new steps to a running cluster.

```
"Statement": [
  {
    "Action": "elasticmapreduce:AddJobFlowSteps",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:
${AWS::AccountId}:cluster/${clusterId}",
        {
```

```
        "clusterId": {
          "Ref": "ClusterId"
        }
      }
    ]
  },
  "Effect": "Allow"
}
]
```

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"
  }
]
```

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"
    }
  ]
```

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"
  }
]
```

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"
  }
]
```

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"
  }
]
```

ElasticsearchHttpPostPolicy

Gives POST and PUT permission to Amazon OpenSearch Service.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "es:ESHttpPost",
      "es:ESHttpPut"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:es:${AWS::Region}:${AWS::AccountId}:domain/
${domainName}/*",
        {
          "domainName": {
            "Ref": "DomainName"
          }
        }
      ]
    }
  }
]
```

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"
          }
        }
      ]
    }
  }
]
```

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"
          }
        }
      ]
    }
  }
]
```

FirehoseCrudPolicy

Gives permission to create, write, update, and delete a 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"
      }
    }
  ]
  }
 }
]
```

FirehoseWritePolicy

Gives permission to write to a Firehose delivery stream.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "firehose:PutRecord",
      "firehose:PutRecordBatch"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:firehose:${AWS::Region}:
${AWS::AccountId}:deliverystream/${deliveryStreamName}",
        {
          "deliveryStreamName": {
            "Ref": "DeliveryStreamName"
          }
        }
      ]
    }
  }
]
```

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"
            }
          }
        ]
      }
    }
  ]
```

KinesisStreamReadPolicy

Gives permission to list and read an Amazon Kinesis stream.

```
"Statement": [
  {
    "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}",
          {
            "streamName": {
              "Ref": "StreamName"
            }
          }
        ]
      }
    }
  ]
```

KMSDecryptPolicy

Gives permission to decrypt with an AWS Key Management Service (AWS KMS) key. Note that
keyId must be an AWS KMS key ID, and not a key alias.

```
"Statement": [
  {
    "Action": "kms:Decrypt",
    "Effect": "Allow",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:kms:${AWS::Region}:${AWS::AccountId}:key/${keyId}",
        {
          "keyId": {
            "Ref": "KeyId"
          }
        }
      ]
    }
  }
```

```
  ]
```

KMSEncryptPolicy

Gives permission to encrypt with an AWS KMS key. Note that keyId must be an AWS KMS key ID, and not a key alias.

```
"Statement": [
  {
    "Action": "kms:Encrypt",
    "Effect": "Allow",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:kms:${AWS::Region}:${AWS::AccountId}:key/${keyId}",
        {
          "keyId": {
            "Ref": "KeyId"
          }
        }
      ]
    }
  }
]
```

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"
          }
        }
```

```
      ]
    }
  }
 ]
```

MobileAnalyticsWriteOnlyAccessPolicy

Gives write-only permission to put event data for all application resources.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "mobileanalytics:PutEvents"
    ],
    "Resource": "*"
  }
]
```

OrganizationsListAccountsPolicy

Gives read-only permission to list child account names and IDs.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "organizations:ListAccounts"
    ],
    "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"
          }
        }
      ]
    }
  }
]
```

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/*"
        }
      ]
    }
  ]
```

RekognitionDetectOnlyPolicy

Gives permission to detect faces, labels, and text.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "rekognition:DetectFaces",
      "rekognition:DetectLabels",
      "rekognition:DetectModerationLabels",
      "rekognition:DetectText"
    ],
    "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": "*"
    }
  ]
```

RekognitionNoDataAccessPolicy

Gives permission to compare and detect faces and labels.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "rekognition:CompareFaces",
      "rekognition:DetectFaces",
      "rekognition:DetectLabels",
      "rekognition:DetectModerationLabels"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:rekognition:${AWS::Region}:${AWS::AccountId}:collection/
${collectionId}",
        {
          "collectionId": {
            "Ref": "CollectionId"
          }
        }
      ]
    }
  }
]
```

RekognitionReadPolicy

Gives permission to list and search faces.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "rekognition:ListCollections",
      "rekognition:ListFaces",
      "rekognition:SearchFaces",
      "rekognition:SearchFacesByImage"
```

```
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:rekognition:${AWS::Region}:${AWS::AccountId}:collection/
${collectionId}",
        {
          "collectionId": {
            "Ref": "CollectionId"
          }
        }
      ]
    }
  }
]
```

RekognitionWriteOnlyAccessPolicy

Gives permission to create collection and index faces.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "rekognition:CreateCollection",
      "rekognition:IndexFaces"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:rekognition:${AWS::Region}:${AWS::AccountId}:collection/
${collectionId}",
        {
          "collectionId": {
            "Ref": "CollectionId"
          }
        }
      ]
    }
  }
]
```

Route53ChangeResourceRecordSetsPolicy

Gives permission to change resource record sets in Route 53.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "route53:ChangeResourceRecordSets"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:route53:::hostedzone/${HostedZoneId}",
        {
          "HostedZoneId": {
            "Ref": "HostedZoneId"
          }
        }
      ]
    }
  }
]
```

S3CrudPolicy

Gives create, read, update, and delete permission to act on the objects in an Amazon S3 bucket.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
      "s3:ListBucket",
      "s3:GetBucketLocation",
      "s3:GetObjectVersion",
      "s3:PutObject",
      "s3:PutObjectAcl",
      "s3:GetLifecycleConfiguration",
      "s3:PutLifecycleConfiguration",
      "s3:DeleteObject"
    ],
    "Resource": [
      {
        "Fn::Sub": [
          "arn:${AWS::Partition}:s3:::${bucketName}",
          {
            "bucketName": {
```

```
          "Ref": "BucketName"
        }
      }
    ]
  },
  {
    "Fn::Sub": [
      "arn:${AWS::Partition}:s3:::${bucketName}/*",
      {
        "bucketName": {
          "Ref": "BucketName"
        }
      }
    ]
  }
]
}
]
```

S3FullAccessPolicy

Gives full access permission to act on the objects in an Amazon S3 bucket.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
      "s3:GetObjectAcl",
      "s3:GetObjectVersion",
      "s3:PutObject",
      "s3:PutObjectAcl",
      "s3:DeleteObject",
      "s3:DeleteObjectTagging",
      "s3:DeleteObjectVersionTagging",
      "s3:GetObjectTagging",
      "s3:GetObjectVersionTagging",
      "s3: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}",
            {
              "bucketName": {
                "Ref": "BucketName"
              }
            }
          ]
        }
      ]
    }
  ]
]
```

S3ReadPolicy

Gives read-only permission to read objects in an Amazon Simple Storage Service (Amazon S3) bucket.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
      "s3:ListBucket",
```

```
          "s3:GetBucketLocation",
          "s3:GetObjectVersion",
          "s3:GetLifecycleConfiguration"
      ],
      "Resource": [
        {
          "Fn::Sub": [
            "arn:${AWS::Partition}:s3:::${bucketName}",
            {
              "bucketName": {
                "Ref": "BucketName"
              }
            }
          ]
        },
        {
          "Fn::Sub": [
            "arn:${AWS::Partition}:s3:::${bucketName}/*",
            {
              "bucketName": {
                "Ref": "BucketName"
              }
            }
          ]
        }
      ]
    }
  ]
```

S3WritePolicy

Gives write permission to write objects into an Amazon S3 bucket.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "s3:PutObject",
      "s3:PutObjectAcl",
      "s3:PutLifecycleConfiguration"
    ],
    "Resource": [
      {
        "Fn::Sub": [
```

```
            "arn:${AWS::Partition}:s3:::${bucketName}",
            {
              "bucketName": {
                "Ref": "BucketName"
              }
            }
          ]
        },
        {
          "Fn::Sub": [
            "arn:${AWS::Partition}:s3:::${bucketName}/*",
            {
              "bucketName": {
                "Ref": "BucketName"
              }
            }
          ]
        }
      ]
    }
  ]
```

SageMakerCreateEndpointConfigPolicy

Gives permission to create an endpoint configuration in SageMaker.

```
"Statement": [
  {
    "Action": [
      "sagemaker:CreateEndpointConfig"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:sagemaker:${AWS::Region}:${AWS::AccountId}:endpoint-
config/${endpointConfigName}",
        {
          "endpointConfigName": {
            "Ref": "EndpointConfigName"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
```

```
    ]
```

SageMakerCreateEndpointPolicy

Gives permission to create an endpoint in SageMaker.

```
"Statement": [
  {
    "Action": [
      "sagemaker:CreateEndpoint"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:sagemaker:${AWS::Region}:${AWS::AccountId}:endpoint/
${endpointName}",
        {
          "endpointName": {
            "Ref": "EndpointName"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

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/*"
      }
    ]
  }
]
```

SESBulkTemplatedCrudPolicy

Gives permission to send Amazon SES email, templated email, and templated bulk emails and to verify identity.

> ### ⓘ Note
>
> The `ses:SendTemplatedEmail` action requires a template ARN. Use
> `SESBulkTemplatedCrudPolicy_v2` instead.

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

```
  ]
```

SESBulkTemplatedCrudPolicy_v2

Gives permission to send Amazon SES email, templated email, and templated bulk emails and to verify identity.

```
"Statement": [
  {
    "Action": [
      "ses:SendEmail",
      "ses:SendRawEmail",
      "ses:SendTemplatedEmail",
      "ses:SendBulkTemplatedEmail"
    ],
    "Effect": "Allow",
    "Resource": [
      {
        "Fn::Sub": [
          "arn:${AWS::Partition}:ses:${AWS::Region}:${AWS::AccountId}:identity/
${identityName}",
          {
            "identityName": {
              "Ref": "IdentityName"
            }
          }
        ]
      },
      {
        "Fn::Sub": [
          "arn:${AWS::Partition}:ses:${AWS::Region}:${AWS::AccountId}:template/
${templateName}",
          {
            "templateName": {
              "Ref": "TemplateName"
            }
          }
        ]
      }
    ]
  },
  {
    "Action": [
```

```
        "ses:GetIdentityVerificationAttributes",
        "ses:VerifyEmailIdentity"
    ],
    "Effect": "Allow",
    "Resource": "*"
  }
]
```

SESCrudPolicy

Gives permission to send email and verify identity.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ses:GetIdentityVerificationAttributes",
      "ses:SendEmail",
      "ses:SendRawEmail",
      "ses:VerifyEmailIdentity"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:ses:${AWS::Region}:${AWS::AccountId}:identity/
${identityName}",
        {
          "identityName": {
            "Ref": "IdentityName"
          }
        }
      ]
    }
  }
]
```

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": "*"
    }
  ]
```

SESSendBouncePolicy

Gives SendBounce permission to an Amazon Simple Email Service (Amazon SES) identity.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ses:SendBounce"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:ses:${AWS::Region}:${AWS::AccountId}:identity/
${identityName}",
        {
          "identityName": {
            "Ref": "IdentityName"
          }
        }
      ]
    }
  }
]
```

SNSCrudPolicy

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

```
"Statement": [
  {
    "Effect": "Allow",
```

```
      "Action": [
        "sns:ListSubscriptionsByTopic",
        "sns:CreateTopic",
        "sns:SetTopicAttributes",
        "sns:Subscribe",
        "sns:Publish"
      ],
      "Resource": {
        "Fn::Sub": [
          "arn:${AWS::Partition}:sns:${AWS::Region}:${AWS::AccountId}:${topicName}*",
          {
            "topicName": {
              "Ref": "TopicName"
            }
          }
        ]
      }
    }
  ]
```

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"
          }
        }
      ]
    }
  }
]
```

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"
          }
        }
      ]
    }
  }
]
```

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"
        }
      }
    ]
  }
 }
]
```

SSMParameterReadPolicy

Gives permission to access a parameter from an Amazon EC2 Systems Manager (SSM) parameter store to load secrets in this account. Use when parameter name doesn't have slash prefix.

> **ⓘ 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"
          }
        }
```

```
      ]
    }
  }
 ]
```

SSMParameterWithSlashPrefixReadPolicy

Gives permission to access a parameter from an Amazon EC2 Systems Manager (SSM) parameter store to load secrets in this account. Use when parameter name has slash prefix.

> ### ⓘ 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"
          }
        }
      ]
    }
  }
]
```

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": "*"
  }
]
```

TextractPolicy

Gives full access to Amazon Textract.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "textract:*"
    ],
    "Resource": "*"
  }
]
```

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": "*"
```

```
      }
   ]
```

Managing permissions with AWS CloudFormation mechanisms

To control access to AWS resources, the AWS Serverless Application Model (AWS SAM) can use the same mechanisms as AWS CloudFormation. For more information, see Controlling access with AWS Identity and Access Management in the *AWS CloudFormation User Guide*.

There are three main options for granting a user permission to manage serverless applications. Each option provides users with different levels of access control.

- Grant administrator permissions.
- Attach necessary AWS managed policies.
- Grant specific AWS Identity and Access Management (IAM) permissions.

Depending on which option you choose, users can manage only serverless applications containing AWS resources that they have permission to access.

The following sections describe each option in more detail.

Grant administrator permissions

If you grant administrator permissions to a user, they can manage serverless applications that contain any combination of AWS resources. This is the simplest option, but it also grants users the broadest set of permissions, which therefore enables them to perform actions with the highest impact.

For more information about granting administrator permissions to a user, see Creating your first IAM admin user and group in the *IAM User Guide*.

Attach necessary AWS managed policies

You can grant users a subset of permissions using AWS managed policies, rather than granting full administrator permissions. If you use this option, make sure that the set of AWS managed policies covers all of the actions and resources required for the serverless applications that the users manage.

For example, the following AWS managed policies are sufficient to deploy the sample Hello World application:

- AWSCloudFormationFullAccess

- IAMFullAccess

- AWSLambda_FullAccess

- AmazonAPIGatewayAdministrator

- AmazonS3FullAccess

- AmazonEC2ContainerRegistryFullAccess

For information about attaching policies to an IAM user, see Changing permissions for an IAM user in the *IAM User Guide*.

Grant specific IAM permissions

For the most granular level of access control, you can grant specific IAM permissions to users using policy statements. If you use this option, make sure that the policy statement includes all of the actions and resources required for the serverless applications that the users manage.

The best practice with this option is to deny users the permission to create roles, including Lambda execution roles, so they can't grant themselves escalated permissions. So, you as the administrator must first create a Lambda execution role that will be specified in the serverless applications that users will manage. For information about creating Lambda execution roles, see Creating an execution role in the IAM console.

For the sample Hello World application the **AWSLambdaBasicExecutionRole** is sufficient to run the application. After you've created a Lambda execution role, modify the AWS SAM template file of the sample Hello World application to add the following property to the `AWS::Serverless::Function` resource:

```
  Role: lambda-execution-role-arn
```

With the modified Hello World application in place, the following policy statement grants sufficient permissions for users to deploy, update, and delete the application:

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

```
            "Action": [
                "cloudformation:CreateChangeSet"
            ],
            "Resource": [
                "arn:aws:cloudformation:*:aws:transform/Serverless-2016-10-31"
            ]
        },
        {
            "Sid": "CloudFormationStack",
            "Effect": "Allow",
            "Action": [
                "cloudformation:CreateChangeSet",
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:DescribeChangeSet",
                "cloudformation:DescribeStackEvents",
                "cloudformation:DescribeStacks",
                "cloudformation:ExecuteChangeSet",
                "cloudformation:GetTemplateSummary",
                "cloudformation:ListStackResources",
                "cloudformation:UpdateStack"
            ],
            "Resource": [
                "arn:aws:cloudformation:*:111122223333:stack/*"
            ]
        },
        {
            "Sid": "S3",
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::*/*"
            ]
        },
        {
            "Sid": "ECRRepository",
            "Effect": "Allow",
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:BatchGetImage",
```

```
                "ecr:CompleteLayerUpload",
                "ecr:CreateRepository",
                "ecr:DeleteRepository",
                "ecr:DescribeImages",
                "ecr:DescribeRepositories",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetRepositoryPolicy",
                "ecr:InitiateLayerUpload",
                "ecr:ListImages",
                "ecr:PutImage",
                "ecr:SetRepositoryPolicy",
                "ecr:UploadLayerPart"
            ],
            "Resource": [
                "arn:aws:ecr:*:111122223333:repository/*"
            ]
        },
        {
            "Sid": "ECRAuthToken",
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "Lambda",
            "Effect": "Allow",
            "Action": [
                "lambda:AddPermission",
                "lambda:CreateFunction",
                "lambda:DeleteFunction",
                "lambda:GetFunction",
                "lambda:GetFunctionConfiguration",
                "lambda:ListTags",
                "lambda:RemovePermission",
                "lambda:TagResource",
                "lambda:UntagResource",
                "lambda:UpdateFunctionCode",
                "lambda:UpdateFunctionConfiguration"
            ],
            "Resource": [
```

```
                "arn:aws:lambda:*:111122223333:function:*"
            ]
        },
        {
            "Sid": "IAM",
            "Effect": "Allow",
            "Action": [
                "iam:CreateRole",
                "iam:AttachRolePolicy",
                "iam:DeleteRole",
                "iam:DetachRolePolicy",
                "iam:GetRole",
                "iam:TagRole"
            ],
            "Resource": [
                "arn:aws:iam::111122223333:role/*"
            ]
        },
        {
            "Sid": "IAMPassRole",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "lambda.amazonaws.com"
                }
            }
        },
        {
            "Sid": "APIGateway",
            "Effect": "Allow",
            "Action": [
                "apigateway:DELETE",
                "apigateway:GET",
                "apigateway:PATCH",
                "apigateway:POST",
                "apigateway:PUT"
            ],
            "Resource": [
                "arn:aws:apigateway:*::*"
            ]
        }
    ]
```

```
}
```

> **ⓘ Note**
>
> The example policy statement in this section grants sufficient permission for you to
> deploy, update, and delete the the [sample Hello World application](). If you add additional
> resource types to your application, you need to update the policy statement to include the
> following:
>
> 1. Permission for your application to call the service's actions.
>
> 2. The service principal, if needed for the service's actions.
>
>
> For example, if you add a Step Functions workflow, you may need to add permissions for
> actions listed [here](), and the `states.amazonaws.com` service principal.

For more information about IAM policies, see [Managing IAM policies]() in the *IAM User Guide*.

Scheduling events with EventBridge Scheduler

What is Amazon EventBridge Scheduler?

Amazon EventBridge Scheduler is a scheduling service that lets you create, initiate, and manage
tens of millions of events and tasks across all AWS services. To learn more about Amazon
EventBridge Scheduler, see [What is Amazon EventBridge Scheduler?]() in the *EventBridge Scheduler
User Guide*.

Topics

- [EventBridge Scheduler support in AWS SAM]()

- [Creating EventBridge Scheduler events in AWS SAM]()

- [Examples]()

- [Learn more]()

EventBridge Scheduler support in AWS SAM

The AWS Serverless Application Model (AWS SAM) template specification provides a simple, short-hand syntax that you can use to schedule events with EventBridge Scheduler for AWS Lambda and AWS Step Functions.

Creating EventBridge Scheduler events in AWS SAM

Set the `ScheduleV2` property as the event type in your AWS SAM template to define your EventBridge Scheduler event. This property supports the `AWS::Serverless::Function` and `AWS::Serverless::StateMachine` resource types.

```
MyFunction:
  Type: AWS::Serverless::Function
  Properties:
    Events:
      CWSchedule:
        Type: ScheduleV2
        Properties:
          ScheduleExpression: 'rate(1 minute)'
          Name: TestScheduleV2Function
          Description: Test schedule event

MyStateMachine:
  Type: AWS::Serverless::StateMachine
  Properties:
    Events:
      CWSchedule:
        Type: ScheduleV2
        Properties:
          ScheduleExpression: 'rate(1 minute)'
          Name: TestScheduleV2StateMachine
          Description: Test schedule event
```

EventBridge Scheduler event scheduling also supports *dead-letter queues (DLQ)* for unprocessed events. For more information on dead-letter queues, see Configuring a dead-letter queue for EventBridge Scheduler in the *EventBridge Scheduler User Guide*.

When a DLQ ARN is specified, AWS SAM configures permissions for the Scheduler schedule to send messages to the DLQ. When a DLQ ARN is not specified, AWS SAM will create the DLQ resource.

Examples

Basic example of defining an EventBridge Scheduler event with AWS SAM

```yaml
Transform: AWS::Serverless-2016-10-31
Resources:
  MyLambdaFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.handler
      Runtime: python3.8
      InlineCode: |
        def handler(event, context):
            print(event)
            return {'body': 'Hello World!', 'statusCode': 200}
      MemorySize: 128
      Events:
        Schedule:
          Type: ScheduleV2
          Properties:
            ScheduleExpression: rate(1 minute)
            Input: '{"hello": "simple"}'

  MySFNFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.handler
      Runtime: python3.8
      InlineCode: |
        def handler(event, context):
            print(event)
            return {'body': 'Hello World!', 'statusCode': 200}
      MemorySize: 128

  StateMachine:
    Type: AWS::Serverless::StateMachine
    Properties:
      Type: STANDARD
      Definition:
        StartAt: MyLambdaState
        States:
          MyLambdaState:
            Type: Task
```

```
          Resource: !GetAtt MySFNFunction.Arn
          End: true
    Policies:
      - LambdaInvokePolicy:
          FunctionName: !Ref MySFNFunction
    Events:
      Events:
      Schedule:
        Type: ScheduleV2
        Properties:
          ScheduleExpression: rate(1 minute)
          Input: '{"hello": "simple"}'
```

Learn more

To learn more about defining the `ScheduleV2` EventBridge Scheduler property, see:

- [ScheduleV2](#) for `AWS::Serverless::Function`.

- [ScheduleV2](#) for `AWS::Serverless::StateMachine`.

Validating AWS SAM template files

Validate your templates with [`sam validate`](#). 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-template>/template.yaml is a valid SAM Template
```

> ⓘ **Note**
>
> The `sam validate` command requires AWS credentials to be configured. For more information, see [Configuring the AWS SAM CLI](#).

Working with layers

Using AWS SAM, you can include layers in your serverless applications. For more information about layers, see [AWS Lambda layers](#) in the *AWS Lambda Developer Guide*.

This topic provides information about the following:

- Including layers in your application
- How layers are cached locally

For information about building custom layers, see [Building layers](#).

Including layers in your application

To include layers in your application, use the `Layers` property of the [AWS::Serverless::Function](#) resource type.

Following is an example AWS SAM template with a Lambda function that includes a layer:

```
ServerlessFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: .
    Handler: my_handler
    Runtime: Python3.7
    Layers:
        - <LayerVersion ARN>
```

How layers are cached locally

When you invoke your function using one of the `sam local` commands, the layers package of your function is downloaded and cached on your local host.

The following table shows the default cache directory locations for different operating systems.

OS	Location
Windows 7	C:\Users\<user>\AppData\Roaming\AWS SAM
Windows 8	C:\Users\<user>\AppData\Roaming\AWS SAM

OS	Location
Windows 10	`C:\Users\<user>\AppData\Roaming\AWS SAM`
macOS	`~/.aws-sam/layers-pkg`
Unix	`~/.aws-sam/layers-pkg`

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:

```
ARN = arn:aws:lambda:us-west-2:111111111111:layer:myLayer:1
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:
        - arn:aws:lambda:us-west-2:111111111111:layer:myLayer:1
        - arn:aws:lambda:us-west-2:111111111111:layer:mySecondLayer:1
```

Unique names are computed the same as the Layer Caching Directory name schema:

```
arn:aws:lambda:us-west-2:111111111111:layer:myLayer:1 = myLayer-1-926eeb5ff1
arn:aws:lambda:us-west-2:111111111111:layer:mySecondLayer:1 =
 mySecondLayer-1-6bc1022bdf
```

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 = 2dd7ac5ffb30d515926aef
```

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

```
python3.7-x86_64-2dd7ac5ffb30d515926aefffd
```

Using nested applications

A serverless application can include one or more **nested applications**. You can deploy a nested application as a stand-alone artifact or as a component of a larger application.

As serverless architectures grow, common patterns emerge in which the same components are defined in multiple application templates. You can now separate out common patterns as dedicated applications, and then nest them as part of new or existing application templates. With nested applications, you can stay more focused on the business logic that's unique to your application.

To define a nested application in your serverless application, use the AWS::Serverless::Application resource type.

You can define nested applications from the following two sources:

- An **AWS Serverless Application Repository application** – You can define nested applications by using applications that are available to your account in the AWS Serverless Application

Repository. These can be *private* applications in your account, applications that are *privately shared* with your account, or applications that are *publicly shared* in the AWS Serverless Application Repository. For more information about the different deployment permissions levels, see [Application Deployment Permissions](#) and [Publishing Applications](#) in the *AWS Serverless Application Repository Developer Guide*.

- A **local application** – You can define nested applications by using applications that are stored on your local file system.

See the following sections for details on how to use AWS SAM to define both of these types of nested applications in your serverless application.

> **ⓘ Note**
>
> The maximum number of applications that can be nested in a serverless application is 200. The maximum number of parameters a nested application can have is 60.

Defining a nested application from the AWS Serverless Application Repository

You can define nested applications by using applications that are available in the AWS Serverless Application Repository. You can also store and distribute applications that contain nested applications using the AWS Serverless Application Repository. To review details of a nested application in the AWS Serverless Application Repository, you can use the AWS SDK, the AWS CLI, or the Lambda console.

To define an application that's hosted in the AWS Serverless Application Repository in your serverless application's AWS SAM template, use the **Copy as SAM Resource** button on the detail page of every AWS Serverless Application Repository application. To do this, follow these steps:

1. Make sure that you're signed in to the AWS Management Console.

2. Find the application that you want to nest in the AWS Serverless Application Repository by using the steps in the [Browsing, Searching, and Deploying Applications](#) section of the *AWS Serverless Application Repository Developer Guide*.

3. Choose the **Copy as SAM Resource** button. The SAM template section for the application that you're viewing is now in your clipboard.

4. Paste the SAM template section into the `Resources:` section of the SAM template file for the
    application that you want to nest in this application.

The following is an example SAM template section for a nested application that's hosted in the
AWS Serverless Application Repository:

```
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:

```
Transform: AWS::Serverless-2016-10-31

Resources:
  applicationaliasname:
    Type: AWS::Serverless::Application
    Properties:
      Location: ../my-other-app/template.yaml
      Parameters:
        # Optional parameter that can have default value overridden
        # ParameterName1: 15 # Uncomment to override default value
        # Required parameter that needs value to be provided
        ParameterName2: YOUR_VALUE
```

If there are no parameter settings, you can omit the `Parameters:` section of the template.

Deploying nested applications

You can deploy your nested application by using the AWS SAM CLI command `sam deploy`. For more details, see Deploying serverless applications.

> **ⓘ Note**
>
> When you deploy an application that contains nested applications, you must acknowledge that. You do this by passing CAPABILITY_AUTO_EXPAND to the CreateCloudFormationChangeSet API, or using the `aws serverlessrepo create-cloud-formation-change-set` AWS CLI command.
> For more information about acknowledging nested applications, see Acknowledging IAM Roles, Resource Policies, and Nested Applications when Deploying Applications in the *AWS Serverless Application Repository Developer Guide*.

Controlling access to API Gateway APIs

To control who can access your Amazon API Gateway APIs, you can enable authorization within your AWS SAM template.

AWS SAM supports several mechanisms for controlling access to your API Gateway APIs. The set of supported mechanisms differs between `AWS::Serverless::HttpApi` and `AWS::Serverless::Api` resource types.

The following table summarizes the mechanisms that each resource type supports.

Mechanisms for controlling access	AWS::Serverless::HttpApi	AWS::Serverless::Api
Lambda authorizers	✓	✓
IAM permissions		✓
Amazon Cognito user pools	✓ *	✓
API keys		✓
Resource policies		✓
OAuth 2.0/JWT authorizers	✓	

* You can use Amazon Cognito as a JSON Web Token (JWT) issuer with the `AWS::Serverless::HttpApi` resource type.

- **Lambda authorizers** – A Lambda authorizer (formerly known as a *custom authorizer*) is a Lambda function that you provide to control access to your API. When your API is called, this Lambda function is invoked with a request context or an authorization token that the client application provides. The Lambda function responds whether the caller is authorized to perform the requested operation.

  Both the `AWS::Serverless::HttpApi` and `AWS::Serverless::Api` resource types support Lambda authorizers.

For more information about Lambda authorizers with `AWS::Serverless::HttpApi`, see [Working with AWS Lambda authorizers for HTTP APIs](#) in the *API Gateway Developer Guide*. For more information about Lambda authorizers with `AWS::Serverless::Api`, see [Use API Gateway Lambda authorizers](#) in the *API Gateway Developer Guide*.

For examples of Lambda authorizers for either resource type, see [Lambda authorizer examples](#).

- **IAM permissions** – You can control who can invoke your API using [AWS Identity and Access Management (IAM) permissions](#). Users calling your API must be authenticated with IAM credentials. Calls to your API succeed only if there is an IAM policy attached to the IAM user that represents the API caller, an IAM group that contains the user, or an IAM role that the user assumes.

  Only the `AWS::Serverless::Api` resource type supports IAM permissions.

  For more information, see [Control access to an API with IAM permissions](#) in the *API Gateway Developer Guide*. For an example, see [IAM permission example](#).

- **Amazon Cognito user pools** – Amazon Cognito user pools are user directories in Amazon Cognito. A client of your API must first sign in a user to the user pool and obtain an identity or access token for the user. Then the client calls your API with one of the returned tokens. The API call succeeds only if the required token is valid.

  The `AWS::Serverless::Api` resource type supports Amazon Cognito user pools. The `AWS::Serverless::HttpApi` resource type supports the use of Amazon Cognito as a JWT issuer.

  For more information, see [Control access to a REST API using Amazon Cognito user pools as authorizer](#) in the *API Gateway Developer Guide*. For an example, see [Amazon Cognito user pool example](#).

- **API keys** – API keys are alphanumeric string values that you distribute to application developer customers to grant access to your API.

  Only the `AWS::Serverless::Api` resource type supports API keys.

  For more information about API keys, see [Creating and using usage plans with API keys](#) in the *API Gateway Developer Guide*. For an example of API keys, see [API key example](#).

- **Resource policies** – Resource policies are JSON policy documents that you can attach to an API Gateway API. Use resource policies to control whether a specified principal (typically an IAM user or role) can invoke the API.

  Only the `AWS::Serverless::Api` resource type supports resource policies as a mechanism for controlling access to API Gateway APIs.

  For more information about resource policies, see [Controlling access to an API with API Gateway resource policies](#) in the *API Gateway Developer Guide*. For an example of resource policies, see [Resource policy example](#).

- **OAuth 2.0/JWT authorizers** – You can use JWTs as a part of [OpenID Connect (OIDC)](#) and [OAuth 2.0](#) frameworks to control access to your APIs. API Gateway validates the JWTs that clients submit with API requests, and allows or denies requests based on token validation and, optionally, scopes in the token.

  Only the `AWS::Serverless::HttpApi` resource type supports OAuth 2.0/JWT authorizers.

  For more information, see [Controlling access to HTTP APIs with JWT authorizers](#) in the *API Gateway Developer Guide*. For an example, see [OAuth 2.0/JWT authorizer example](#).

Choosing a mechanism to control access

The mechanism that you choose to use for controlling access to your API Gateway APIs depends on a few factors. For example, if you have a greenfield project without either authorization or access control set up, then Amazon Cognito user pools might be your best option. This is because when you set up user pools, you also automatically set up both authentication and access control.

However, if your application already has authentication set up, then using Lambda authorizers might be your best option. This is because you can call your existing authentication service and return a policy document based on the response. Also, if your application requires custom authentication or access control logic that user pools don't support, then Lambda authorizers might be your best option.

When you've chosen which mechanism to use, see the corresponding section in [Examples](#) for how to use AWS SAM to configure your application to use that mechanism.

Customizing error responses

You can use AWS SAM to customize the content of some API Gateway error responses. Only the `AWS::Serverless::Api` resource type supports customized API Gateway responses.

For more information about API Gateway responses, see [Gateway responses in API Gateway](#) in the *API Gateway Developer Guide*. For an example of customized responses, see [Customized response example](#).

Examples

- [Lambda authorizer examples](#)
- [IAM permission example](#)
- [Amazon Cognito user pool example](#)
- [API key example](#)
- [Resource policy example](#)
- [OAuth 2.0/JWT authorizer example](#)
- [Customized response example](#)

Lambda authorizer examples

The `AWS::Serverless::Api` resource type supports two types of Lambda authorizers: TOKEN authorizers and REQUEST authorizers. The `AWS::Serverless::HttpApi` resource type supports only REQUEST authorizers. The following are examples of each type.

Lambda TOKEN authorizer example (AWS::Serverless::Api)

You can control access to your APIs by defining a Lambda TOKEN authorizer within your AWS SAM template. To do this, you use the [ApiAuth](#) data type.

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

> **ⓘ Note**
>
> In the following example, the SAM `FunctionRole` is implicitly generated.

```
Resources:
```

```yaml
  MyApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      Auth:
        DefaultAuthorizer: MyLambdaTokenAuthorizer
        Authorizers:
          MyLambdaTokenAuthorizer:
            FunctionArn: !GetAtt MyAuthFunction.Arn

  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: ./src
      Handler: index.handler
      Runtime: nodejs12.x
      Events:
        GetRoot:
          Type: Api
          Properties:
            RestApiId: !Ref MyApi
            Path: /
            Method: get

  MyAuthFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: ./src
      Handler: authorizer.handler
      Runtime: nodejs12.x
```

For more information about Lambda authorizers, see [Use API Gateway Lambda authorizers](#) in the *API Gateway Developer Guide*.

Lambda REQUEST authorizer example (AWS::Serverless::Api)

You can control access to your APIs by defining a Lambda REQUEST authorizer within your AWS SAM template. To do this, you use the [ApiAuth](#) 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
      Handler: authorizer.handler
      Runtime: nodejs12.x
```

For more information about Lambda authorizers, see Use API Gateway Lambda authorizers in the
API Gateway Developer Guide.

Lambda authorizer example (AWS::Serverless::HttpApi)

You can control access to your HTTP APIs by defining a Lambda authorizer within your AWS SAM
template. To do this, you use the HttpApiAuth data type.

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

```
Resources:
  MyApi:
    Type: AWS::Serverless::HttpApi
    Properties:
      StageName: Prod
      Auth:
        DefaultAuthorizer: MyLambdaRequestAuthorizer
        Authorizers:
          MyLambdaRequestAuthorizer:
            FunctionArn: !GetAtt MyAuthFunction.Arn
            FunctionInvokeRole: !GetAtt MyAuthFunctionRole.Arn
            Identity:
              Headers:
                - Authorization
            AuthorizerPayloadFormatVersion: 2.0
            EnableSimpleResponses: true

  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: ./src
      Handler: index.handler
      Runtime: nodejs12.x
      Events:
        GetRoot:
          Type: HttpApi
          Properties:
            ApiId: !Ref MyApi
            Path: /
            Method: get
            PayloadFormatVersion: "2.0"

  MyAuthFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: ./src
      Handler: authorizer.handler
      Runtime: nodejs12.x
```

IAM permission example

You can control access to your APIs by defining IAM permissions within your AWS SAM template. To do this, you use the [ApiAuth](#) data type.

The following is an example AWS SAM template that uses for IAM permissions:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
  MyApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      Description: 'API with IAM authorization'
      Auth:
        DefaultAuthorizer: AWS_IAM #sets AWS_IAM auth for all methods in this API
  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.handler
      Runtime: python3.10
      Events:
        GetRoot:
          Type: Api
          Properties:
            RestApiId: !Ref MyApi
            Path: /
            Method: get
      InlineCode: |
        def handler(event, context):
          return {'body': 'Hello World!', 'statusCode': 200}
```

For more information about IAM permissions, see [Control access for invoking an API](#) in the *API Gateway Developer Guide*.

Amazon Cognito user pool example

You can control access to your APIs by defining Amazon Cognito user pools within your AWS SAM template. To do this, you use the [ApiAuth](#) data type.

The following is an example AWS SAM template section for a user pool:

```
Resources:
  MyApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      Cors: "'*'"
      Auth:
        DefaultAuthorizer: MyCognitoAuthorizer
        Authorizers:
          MyCognitoAuthorizer:
            UserPoolArn: !GetAtt MyCognitoUserPool.Arn

  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: ./src
      Handler: lambda.handler
      Runtime: nodejs12.x
      Events:
        Root:
          Type: Api
          Properties:
            RestApiId: !Ref MyApi
            Path: /
            Method: GET

  MyCognitoUserPool:
    Type: AWS::Cognito::UserPool
    Properties:
      UserPoolName: !Ref CognitoUserPoolName
      Policies:
        PasswordPolicy:
          MinimumLength: 8
      UsernameAttributes:
        - email
      Schema:
        - AttributeDataType: String
          Name: email
          Required: false

  MyCognitoUserPoolClient:
    Type: AWS::Cognito::UserPoolClient
    Properties:
```

```
        UserPoolId: !Ref MyCognitoUserPool
        ClientName: !Ref CognitoUserPoolClientName
        GenerateSecret: false
```

For more information about Amazon Cognito user pools, see Control access to a REST API using Amazon Cognito user pools as authorizer in the *API Gateway Developer Guide*.

API key example

You can control access to your APIs by requiring API keys within your AWS SAM template. To do this, you use the ApiAuth data type.

The following is an example AWS SAM template section for API keys:

```
Resources:
  MyApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      Auth:
        ApiKeyRequired: true # sets for all methods

  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: .
      Handler: index.handler
      Runtime: nodejs12.x
      Events:
        ApiKey:
          Type: Api
          Properties:
            RestApiId: !Ref MyApi
            Path: /
            Method: get
            Auth:
              ApiKeyRequired: true
```

For more information about API keys, see Creating and using usage plans with API keys in the *API Gateway Developer Guide.*

Resource policy example

You can control access to your APIs by attaching a resource policy within your AWS SAM template. To do this, you use the [ApiAuth](#) data type.

The following is an example AWS SAM template for a private API. A private API must have a resource policy to deploy.

```yaml
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
  MyPrivateApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      EndpointConfiguration: PRIVATE  # Creates a private API. Resource policies are
 required for all private APIs.
      Auth:
        ResourcePolicy:
          CustomStatements: {
              Effect: 'Allow',
              Action: 'execute-api:Invoke',
              Resource: ['execute-api:/*/*/*'],
              Principal: '*'
            }
  MyFunction:
    Type: 'AWS::Serverless::Function'
    Properties:
      InlineCode: |
        def handler(event, context):
          return {'body': 'Hello World!', 'statusCode': 200}
      Handler: index.handler
      Runtime: python3.10
      Events:
        AddItem:
          Type: Api
          Properties:
            RestApiId:
              Ref: MyPrivateApi
            Path: /
            Method: get
```

For more information about resource policies, see [Controlling access to an API with API Gateway resource policies](#) in the *API Gateway Developer Guide*. For more information about private APIs, see [Creating a private API in Amazon API Gateway](#) in the *API Gateway Developer Guide*.

OAuth 2.0/JWT authorizer example

You can control access to your APIs using JWTs as part of [OpenID Connect (OIDC)](#) and [OAuth 2.0](#) frameworks. To do this, you use the [HttpApiAuth](#) data type.

The following is an example AWS SAM template section for an OAuth 2.0/JWT authorizer:

```
Resources:
  MyApi:
    Type: AWS::Serverless::HttpApi
    Properties:
      Auth:
        Authorizers:
          MyOauth2Authorizer:
            AuthorizationScopes:
              - scope
            IdentitySource: $request.header.Authorization
            JwtConfiguration:
              audience:
                - audience1
                - audience2
              issuer: "https://www.example.com/v1/connect/oidc"
        DefaultAuthorizer: MyOauth2Authorizer
      StageName: Prod
  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: ./src
      Events:
        GetRoot:
          Properties:
            ApiId: MyApi
            Method: get
            Path: /
            PayloadFormatVersion: "2.0"
          Type: HttpApi
      Handler: index.handler
      Runtime: nodejs12.x
```

For more information about OAuth 2.0/JWT authorizers, see Controlling access to HTTP APIs with JWT authorizers in the *API Gateway Developer Guide*.

Customized response example

You can customize some API Gateway error responses by defining response headers within your AWS SAM template. To do this, you use the Gateway Response Object data type.

The following is an example AWS SAM template that creates a customized response for the DEFAULT_5XX error.

```yaml
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
  MyApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      GatewayResponses:
        DEFAULT_5XX:
          ResponseParameters:
            Headers:
              Access-Control-Expose-Headers: "'WWW-Authenticate'"
              Access-Control-Allow-Origin: "'*'"
              ErrorHeader: "'MyCustomErrorHeader'"
          ResponseTemplates:
            application/json: "{\"message\": \"Error on the $context.resourcePath
 resource\" }"

  GetFunction:
    Type: AWS::Serverless::Function
    Properties:
      Runtime: python3.10
      Handler: index.handler
      InlineCode: |
        def handler(event, context):
          raise Exception('Check out the new response!')
      Events:
        GetResource:
          Type: Api
          Properties:
            Path: /error
            Method: get
```

```
            RestApiId: !Ref MyApi
```

For more information about API Gateway responses, see [Gateway responses in API Gateway](#) in the
API Gateway Developer Guide.

Orchestrating AWS resources with AWS Step Functions

You can use [AWS Step Functions](#) to orchestrate AWS Lambda functions and other AWS resources to
form complex and robust workflows.

> **ⓘ Note**
>
> To manage AWS SAM templates that contain Step Functions state machines, you must use
> version 0.52.0 or later of the AWS SAM CLI. To check which version you have, execute the
> command `sam --version`.

Step Functions is based on the concepts of [tasks](#) and [state machines](#). You define state machines
using the JSON-based [Amazon States Language](#). The [Step Functions console](#) displays a graphical
view of your state machine's structure so you can visually check your state machine's logic and
monitor executions.

With Step Functions support in AWS Serverless Application Model (AWS SAM), you can do the
following:

- Define state machines, either directly within an AWS SAM template or in a separate file
- Create state machine execution roles through AWS SAM policy templates, inline policies, or
  managed policies
- Trigger state machine executions with API Gateway or Amazon EventBridge events, on a schedule
  within an AWS SAM template, or by calling APIs directly
- Use available [AWS SAM Policy Templates](#) for common Step Functions development patterns.

Example

The following example snippet from a AWS SAM template file defines a Step Functions state
machine in a definition file. Note that the `my_state_machine.asl.json` file must be written in
[Amazon States Language](#).

```
AWSTemplateFormatVersion: "2010-09-09"
Transform: AWS::Serverless-2016-10-31
Description: Sample SAM template with Step Functions State Machine

Resources:
  MyStateMachine:
    Type: AWS::Serverless::StateMachine
    Properties:
      DefinitionUri: statemachine/my_state_machine.asl.json
      ...
```

To download a sample AWS SAM application that includes a Step Functions state machine, see
Create a Step Functions State Machine Using AWS SAM in the *AWS Step Functions Developer Guide*.

More information

To learn more about Step Functions and using it with AWS SAM, see the following:

- How AWS Step Functions works

- AWS Step Functions and AWS Serverless Application Model

- Tutorial: Create a Step Functions State Machine Using AWS SAM

- AWS SAM Specification: AWS::Serverless::StateMachine

Configuring code signing for AWS SAM applications

You can use AWS SAM to enable code signing with your serverless applications to help ensure
that only trusted code is deployed. For more information about the code signing feature, see
Configuring code signing for Lambda functions in the *AWS Lambda Developer Guide*.

Before you can configure code signing for your serverless application, you must create a signing
profile using AWS Signer. You use this signing profile for the following tasks:

1. **Creating a code signing configuration** – Declare an `AWS::Lambda::CodeSigningConfig`
   resource to specify the signing profiles of trusted publishers and to set the policy action
   for validation checks. You can declare this object in the same AWS SAM template as your
   serverless function, in a different AWS SAM template, or in an AWS CloudFormation
   template. You then enable code signing for a serverless function by specify the

       `CodeSigningConfigArn` property the function with the Amazon Resource Name (ARN) of an `AWS::Lambda::CodeSigningConfig` resource.

2. **Signing your code** – Use the `sam package` or `sam deploy` command with the `--signing-profiles` option.

> **ⓘ Note**
>
> In order to successfully sign your code with the `sam package` or `sam deploy` commands, versioning must be enabled for the Amazon S3 bucket you use with these commands. If you are using the Amazon S3 Bucket that AWS SAM creates for you, versioning is enabled automatically. For more information about Amazon S3 bucket versioning and instructions for enabling versioning on an Amazon S3 bucket that you provide, see Using versioning in Amazon S3 buckets in the *Amazon Simple Storage Service User Guide*.

When you deploy a serverless application, Lambda performs validation checks on all functions that you've enabled code signing for. Lambda also performs validation checks on any layers that those functions depend on. For more information about Lambda's validation checks, see Signature validation in the *AWS Lambda Developer Guide*.

Example

Creating a signing profile

To create a signing profile, run the following command:

```
aws signer put-signing-profile --platform-id "AWSLambda-SHA384-ECDSA" --profile-
name MySigningProfile
```

If the previous command is successful, you see the signing profile's ARN returned. For example:

```
{
    "arn": "arn:aws:signer:us-east-1:111122223333:/signing-profiles/MySigningProfile",
    "profileVersion": "SAMPLEverx",
    "profileVersionArn": "arn:aws:signer:us-east-1:111122223333:/signing-
profiles/MySigningProfile/SAMPLEverx"
}
```

Example   629

The `profileVersionArn` field contains the ARN to use when you create the code signing configuration.

Creating a code signing configuration and enabling code signing for a function

The following example AWS SAM template declares an [AWS::Lambda::CodeSigningConfig](#) resource and enables code signing for a Lambda function. In this example, there is one trusted profile, and deployments are rejected if the signature checks fail.

```
Resources:
  HelloWorld:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: hello_world/
      Handler: app.lambda_handler
      Runtime: python3.7
      CodeSigningConfigArn: !Ref MySignedFunctionCodeSigningConfig

  MySignedFunctionCodeSigningConfig:
    Type: AWS::Lambda::CodeSigningConfig
    Properties:
      Description: "Code Signing for MySignedLambdaFunction"
      AllowedPublishers:
        SigningProfileVersionArns:
          - MySigningProfile-profileVersionArn
      CodeSigningPolicies:
        UntrustedArtifactOnDeployment: "Enforce"
```

Signing your code

You can sign your code when packaging or deploying your application. Specify the `--signing-profiles` option with either the `sam package` or `sam deploy` command, as shown in the following example commands.

Signing your function code when packaging your application:

```
sam package --signing-profiles HelloWorld=MySigningProfile --s3-bucket test-bucket --
output-template-file packaged.yaml
```

Signing both your function code and a layer that your function depends on, when packaging your application:

Example   630

```
sam package --signing-profiles HelloWorld=MySigningProfile MyLayer=MySigningProfile --
s3-bucket test-bucket --output-template-file packaged.yaml
```

Signing your function code and a layer, then performing a deployment:

```
sam deploy --signing-profiles HelloWorld=MySigningProfile MyLayer=MySigningProfile --
s3-bucket test-bucket --template-file packaged.yaml --stack-name --region us-east-1 --
capabilities CAPABILITY_IAM
```

> **ⓘ Note**
>
> In order to successfully sign your code with the `sam package` or `sam deploy` commands,
> versioning must be enabled for the Amazon S3 bucket you use with these commands. If
> you are using the Amazon S3 Bucket that AWS SAM creates for you, versioning is enabled
> automatically. For more information about Amazon S3 bucket versioning and instructions
> for enabling versioning on an Amazon S3 bucket that you provide, see Using versioning in
> Amazon S3 buckets in the *Amazon Simple Storage Service User Guide*.

Providing signing profiles with `sam deploy --guided`

When you run the `sam deploy --guided` command with a serverless application that's
configured with code signing, AWS SAM prompts you to provide the signing profile to use for code
signing. For more information about `sam deploy --guided` prompts, see sam deploy in the AWS
SAM CLI command reference.

Building serverless applications

Building your serverless application involves taking your AWS SAM template file, application code, and any applicable language-specific files and dependencies, and placing all build artifacts in the proper format and location for subsquent 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.

For an introduction to using the `sam build` command, see Using sam build.

Topics

- Building applications
- Building layers
- Building custom runtimes

Building applications

To build your serverless application, use the `sam build` command. This command also gathers the build artifacts of your application's dependencies and places them in the proper format and location for next steps, such as locally testing, packaging, and deploying.

You specify your application's dependencies in a manifest file, such as `requirements.txt` (Python) or `package.json` (Node.js), or by using the `Layers` property of a function resource. The `Layers` property contains a list of AWS Lambda layer resources that the Lambda function depends on.

The format of your application's build artifacts depends on each function's `PackageType` property. The options for this property are:

- `Zip` – A .zip file archive, which contains your application code and its dependencies. If you package your code as a .zip file archive, you must specify a Lambda runtime for your function.

- **Image** – A container image, which includes the base operating system, runtime, and extensions, in addition to your application code and its dependencies.

For more information about Lambda package types, see [Lambda deployment packages](#) in the *AWS Lambda Developer Guide*.

Topics

- [Building a .zip file archive](#)
- [Building a container image](#)
- [Container environment variable file](#)
- [Speed up build times by building your project in the source folder](#)
- [Examples](#)
- [Building functions outside of AWS SAM](#)
- [Building Node.js Lambda functions with esbuild](#)
- [Building .NET 7 Lambda functions with Native AOT compilation](#)
- [Building Rust Lambda functions with Cargo Lambda](#)

Building a .zip file archive

To build your serverless application as a .zip file archive, declare `PackageType: Zip` for your serverless function.

AWS SAM builds your application for the [architecture](#) that you specify. If you don't specify an architecture, AWS SAM uses `x86_64` by default.

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

When you use the `--use-container` option, by default AWS SAM pulls the container image from [Amazon ECR Public](#). If you would like to pull a container image from another repository, for example DockerHub, you can use the `--build-image` option and provide the URI of an alternate container image. Following are two example commands for building applications using container images from the DockerHub repository:

```
# Build a Node.js 20 application using a container image pulled from DockerHub
sam build --use-container --build-image amazon/aws-sam-cli-build-image-nodejs20.x

# Build a function resource using the Python 3.12 container image pulled from DockerHub
sam build --use-container --build-image Function1=amazon/aws-sam-cli-build-image-
python3.12
```

For a list of URIs you can use with `--build-image`, see [Image repositories](#) which contains DockerHub URIs for a number of supported runtimes.

For additional examples of building a .zip file archive application, see the Examples section later in this topic.

Building a container image

To build your serverless application as a container image, declare `PackageType: Image` for your serverless function. You must also declare the `Metadata` resource attribute with the following entries:

`Dockerfile`

  The name of the Dockerfile associated with the Lambda function.

`DockerContext`

  The location of the Dockerfile.

`DockerTag`

  (Optional) A tag to apply to the built image.

`DockerBuildArgs`

  Build arguments for the build.

The following is an example `Metadata` resource attribute section:

```
    Metadata:
      Dockerfile: Dockerfile
      DockerContext: ./hello_world
      DockerTag: v1
```

To download a sample application that's configured with the `Image` package type, see Tutorial: Deploying a Hello World application in **Tutorial: Deploying a Hello World application**. At the prompt asking which package type you want to install, choose `Image`.

> ⓘ **Note**
>
> If you specify a multi-architecture base image in your Dockerfile, AWS SAM builds your container image for your host machine's architecture. To build for a different architecture, specify a base image that uses the specific target architecture.

Container environment variable file

To provide a JSON file that contains environment variables for the build container, use the `--container-env-var-file` argument with the `sam build` command. You can provide a single environment variable that applies to all serverless resources, or different environment variables for each resource.

Format

The format for passing environment variables to a build container depends on how many environment variables you provide for your resources.

To provide a single environment variable for all resources, specify a `Parameters` object like the following:

```
{
  "Parameters": {
    "GITHUB_TOKEN": "TOKEN_GLOBAL"
  }
}
```

To provide different environment variables for each resource, specify objects for each resource like the following:

```
{
  "MyFunction1": {
    "GITHUB_TOKEN": "TOKEN1"
  },
```

```
    "MyFunction2": {
      "GITHUB_TOKEN": "TOKEN2"
    }
  }
}
```

Save your environment variables as a file, for example, named `env.json`. The following command uses this file to pass your environment variables to the build container:

```
sam build --use-container --container-env-var-file env.json
```

Precedence

- The environment variables that you provide for specific resources take precedence over the single environment variable for all resources.

- Environment variables that you provide on the command line take precedence over environment variables in a file.

Speed up build times by building your project in the source folder

For supported runtimes and build methods, you can use the `--build-in-source` option to build your project directly in the source folder. By default, the AWS SAM CLI builds in a temporary directory, which involves copying over source code and project files. With `--build-in-source`, the AWS SAM CLI builds directly in your source folder, which speeds up the build process by removing the need to copy files to a temporary directory.

For a list of supported runtimes and build methods, see [--build-in-source](#).

Examples

Example 1: .zip file archive

The following `sam build` commands build a .zip file archive:

```
# Build all functions and layers, and their dependencies
sam build

# Run the build process inside a Docker container that functions like a Lambda
  environment
```

```
sam build --use-container

# Build a Node.js 20 application using a container image pulled from DockerHub
sam build --use-container --build-image amazon/aws-sam-cli-build-image-nodejs20.x

# Build a function resource using the Python 3.12 container image pulled from DockerHub
sam build --use-container --build-image Function1=amazon/aws-sam-cli-build-image-
python3.12

# Build and run your functions locally
sam build && sam local invoke

# For more options
sam build --help
```

Example 2: Container image

The following AWS SAM template builds as a container image:

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      PackageType: Image
      ImageConfig:
        Command: ["app.lambda_handler"]
    Metadata:
      Dockerfile: Dockerfile
      DockerContext: ./hello_world
      DockerTag: v1
```

The following is an example Dockerfile:

```
FROM public.ecr.aws/lambda/python:3.12

COPY app.py requirements.txt ./

RUN python3.12 -m pip install -r requirements.txt

# Overwrite the command by providing a different command directly in the template.
CMD ["app.lambda_handler"]
```

Example 3: npm ci

For Node.js applications, you can use npm ci instead of npm install to install dependencies. To use npm ci, specify UseNpmCi: True under BuildProperties in your Lambda function's Metadata resource attribute. To use npm ci, your application must have a package-lock.json or npm-shrinkwrap.json file present in the CodeUri for your Lambda function.

The following example uses npm ci to install dependencies when you run sam build:

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: hello-world/
      Handler: app.handler
      Runtime: nodejs20.x
      Architectures:
        - x86_64
      Events:
        HelloWorld:
          Type: Api
          Properties:
            Path: /hello
            Method: get
    Metadata:
      BuildProperties:
        UseNpmCi: True
```

Building functions outside of AWS SAM

By default, when you run **sam build**, AWS SAM builds all of your function resources. Other options include:

- **Build all function resources outside of AWS SAM** – If you build all of your function resources manually or through another tool, **sam build** is not required. You can skip **sam build** and move on to the next step in your process, such as performing local testing or deploying your application.

- **Build some function resources outside of AWS SAM** – If you want AWS SAM to build some of your function resources while having other function resources built outside of AWS SAM, you can specify this in your AWS SAM template.

Build some function resources outside of AWS SAM

To have AWS SAM skip a function when using **sam build**, configure the following in your AWS SAM template:

1. Add the `SkipBuild: True` metadata property to your function.

2. Specify the path to your built function resources.

Here is an example, with `TestFunction` configured to be skipped. Its built resources are located at `built-resources/TestFunction.zip`.

```
TestFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: built-resources/TestFunction.zip
    Handler: TimeHandler::handleRequest
    Runtime: java11
  Metadata:
    SkipBuild: True
```

Now, when you run **sam build**, AWS SAM will do the following:

1. AWS SAM will skip functions configured with `SkipBuild: True`.

2. AWS SAM will build all other function resources and cache them in the `.aws-sam` build directory.

3. For skipped functions, their template in the `.aws-sam` build directory will automatically be updated to reference the specified path to your built function resources.

   Here is an example of the cached template for `TestFunction` in the `.aws-sam` build directory:

   ```
   TestFunction:
     Type: AWS::Serverless::Function
     Properties:
       CodeUri: ../../built-resources/TestFunction.zip
       Handler: TimeHandler::handleRequest
       Runtime: java11
     Metadata:
       SkipBuild: True
   ```

Building Node.js Lambda functions with esbuild

To build and package Node.js AWS Lambda functions, you can use the AWS SAM CLI with the esbuild JavaScript bundler. The esbuild bundler supports Lambda functions that you write in TypeScript.

To build a Node.js Lambda function with esbuild, add a `Metadata` object to your `AWS:Serverless::Function` resource and specify `esbuild` for the `BuildMethod`. When you run the **sam build** command, AWS SAM uses esbuild to bundle your Lambda function code.

Metadata properties

The `Metadata` object supports the following properties for esbuild.

BuildMethod

Specifies the bundler for your application. The only supported value is `esbuild`.

BuildProperties

Specifies the build properties for your Lambda function code.

The `BuildProperties` object supports the following properties for esbuild. All of the properties are optional. By default, AWS SAM uses your Lambda function handler for the entry point.

EntryPoints

Specifies entry points for your application.

External

Specifies the list of packages to omit from the build. For more information, see [External](#) in the *esbuild website*.

Format

Specifies the output format of the generated JavaScript files in your application. For more information, see [Format](#) in the *esbuild website*.

Loader

Specifies the list of configurations for loading data for a given file type.

MainFields

Specifies which `package.json` fields to try to import when resolving a package. The default value is `main,module`.

Minify

Specifies whether to minify the bundled output code. The default value is `true`.

OutExtension

Customize the file extension of the files that esbuild generates. For more information, see Out extension in the *esbuild website*.

Sourcemap

Specifies whether the bundler produces a source map file. The default value is `false`.

When set to `true`, `NODE_OPTIONS: --enable-source-maps` is appended to the Lambda function's environment variables, and a source map is generated and included in the function.

Alternatively, when `NODE_OPTIONS: --enable-source-maps` is included in the function's environment variables, `Sourcemap` is automatically set to `true`.

When conflicting, `Sourcemap: false` takes precedence over `NODE_OPTIONS: --enable-source-maps`.

> ⓘ **Note**
>
> By default, Lambda encrypts all environment variables at rest with AWS Key Management Service (AWS KMS). When using source maps, for the deployment to succeed, your function's execution role must have permission to perform the `kms:Encrypt` action.

SourcesContent

Specifies whether to include your source code in your source map file. Configure this property when `Sourcemap` is set to `'true'`.

- Specify `SourcesContent: 'true'` to include all source code.
- Specify `SourcesContent: 'false'` to exclude all source code. This results in smaller source maps file sizes, which is useful in production by reducing start-up times. However, source code won't be available in the debugger.

The default value is `SourcesContent: true`.

For more information, see [Sources content](#) in the *esbuild website*.

Target

Specifies the target ECMAScript version. The default value is `es2020`.

TypeScript Lambda function example

The following example AWS SAM template snippet uses esbuild to create a Node.js Lambda function from TypeScript code in `hello-world/app.ts`.

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: hello-world/
      Handler: app.handler
      Runtime: nodejs20.x
      Architectures:
        - x86_64
      Events:
        HelloWorld:
          Type: Api
          Properties:
            Path: /hello
            Method: get
      Environment:
        Variables:
          NODE_OPTIONS: --enable-source-maps
    Metadata:
      BuildMethod: esbuild
      BuildProperties:
        Format: esm
        Minify: false
        OutExtension:
          - .js=.mjs
        Target: "es2020"
        Sourcemap: true
        EntryPoints:
          - app.ts
        External:
```

```
        - "<package-to-exclude>"
```

Building .NET 7 Lambda functions with Native AOT compilation

Build and package your .NET 7 AWS Lambda functions with the AWS Serverless Application Model (AWS SAM), utilizing Native Ahead-of-Time (AOT) compilation to improve AWS Lambda cold-start times.

Topics

- .NET 7 Native AOT overview
- Using AWS SAM with your .NET 7 Lambda functions
- Install prerequisites
- Define .NET 7 Lambda functions in your AWS SAM template
- Build your application with the AWS SAM CLI
- Learn more

.NET 7 Native AOT overview

Historically, .NET Lambda functions have cold-start times which impact user experience, system latency, and usage costs of your serverless applications. With .NET 7, Microsoft adds support for Native AOT compilation, which speeds up performance by compiling .NET code directly to machine and operating system native code, eliminating Just-in-Time (JIT) compiling at runtime. With .NET 7 Native AOT compilation, you can improve cold-start times of your Lambda functions. To learn more about Native AOT for .NET 7, see Using Native AOT in the *Dotnet GitHub repository*.

Using AWS SAM with your .NET 7 Lambda functions

Do the following to configure your .NET 7 Lambda functions with the AWS Serverless Application Model (AWS SAM):

- Install prerequisites on your development machine.
- Define .NET 7 Lambda functions in your AWS SAM template.
- Build your application with the AWS SAM CLI.

Install prerequisites

The following are required prerequisites:

- The AWS SAM CLI

- The .NET Core CLI

- The Amazon.Lambda.Tools .NET Core Global Tool

- Docker

Install the AWS SAM CLI

1. To check if you already have the AWS SAM CLI installed, run the following:

    ```
    sam --version
    ```

2. To install the AWS SAM CLI, see Installing the AWS SAM CLI.

3. To upgrade an installed version of the AWS SAM CLI, see Upgrading the AWS SAM CLI.

Install the .NET Core CLI

1. To download and install the .NET Core CLI, see Download .NET from Microsoft's website.

2. For more information on the .NET Core CLI, see .NET Core CLI in the *AWS Lambda Developer Guide*.

Install the Amazon.Lambda.Tools .NET Core Global Tool

1. Run the following command:

    ```
    dotnet tool install -g Amazon.Lambda.Tools
    ```

2. If you already have the tool installed, you can make sure that it is the latest version using the following command:

    ```
    dotnet tool update -g Amazon.Lambda.Tools
    ```

3. For more information about the Amazon.Lambda.Tools .NET Core Global Tool, see the AWS Extensions for .NET CLI repository on GitHub.

Install Docker

- Building with Native AOT, requires Docker to be installed. For installation instructions, see
  [Installing Docker to use with the AWS SAM CLI](#).

Define .NET 7 Lambda functions in your AWS SAM template

To define a .NET7 Lambda function in your AWS SAM template, do the following:

- Define a custom runtime by setting `Runtime` to `provided.al2`. We define a custom runtime
  because a .NET 7 runtime option is not available.

- Add a `Metadata` object to your `AWS::Serverless::Function` resource and specify `dotnet7`
  for the `BuildMethod`.

```
Resources:
HelloWorldFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: function-source-folder
    Handler: app.handler
    Runtime: provided.al2
    Architectures:
      - x86_64
    Events:
      HelloWorld:
        Type: Api
        Properties:
          Path: /hello
          Method: get
  Metadata:
    BuildMethod: dotnet7
```

> **ⓘ Note**
>
> For the `Architectures` property, `x86_64` is the only supported option because .NET 7
> cannot run on `arm64`. For more information, see the [Reconsider .NET 7 build environment
> GitHub issue](#).

Build your application with the AWS SAM CLI

From your project's root directory, run `sam build` to begin building your application. If the `PublishAot` property has been defined in your .NET 7 project file, the AWS SAM CLI will build with Native AOT compilation. To learn more about the `PublishAot` property, see [Native AOT Deployment](#) in Microsoft's *.NET documentation*.

To build your function, the AWS SAM CLI invokes the .NET Core CLI which uses the Amazon.Lambda.Tools .NET Core Global Tool.

> ⓘ **Note**
>
> When building, if a `.sln` file exists in the same or parent directory of your project, the directory containing the `.sln` file will be mounted to the container. If a `.sln` file is not found, only the project folder is mounted. Therefore, if you are building a multi-project application, ensure the `.sln` file is property located.

Learn more

For more information on building .NET 7 Lambda functions, see [Building serverless applications on Lambda using .NET 7](#).

For a reference of the **sam build** command, see [sam build](#).

Building Rust Lambda functions with Cargo Lambda

This feature is in preview release for AWS SAM and is subject to change.

Use the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) with your Rust AWS Lambda functions.

Topics

- [Prerequisites](#)
- [Configuring AWS SAM to use with Rust Lambda functions](#)
- [Examples](#)

Prerequisites

Rust language

To install Rust, see [Install Rust](#) in the *Rust language website*.

Cargo Lambda

The AWS SAM CLI requires installation of [Cargo Lambda](#), a subcommand for Cargo. For installation instructions, see [Installation](#) in the *Cargo Lambda documentation*.

Docker

Building and testing Rust Lambda functions requires Docker. For installation instructions, see [Installing Docker](#).

Opt in to AWS SAM CLI beta feature

Since this feature is in preview, you must opt in using one of the following methods:

1. Use the environment variable: SAM_CLI_BETA_RUST_CARGO_LAMBDA=1.

2. Add the following to your `samconfig.toml` file:

   ```
   [default.build.parameters]
   beta_features = true
   [default.sync.parameters]
   beta_features = true
   ```

3. Use the `--beta-features` option when using a supported AWS SAM CLI command. For example:

   ```
   $ sam build --beta-features
   ```

4. Choose option y when the AWS SAM CLI prompts you to opt in. The following is an example:

   ```
   $ sam build
   Starting Build use cache
   Build method "rust-cargolambda" is a beta feature.
   Please confirm if you would like to proceed
   You can also enable this beta feature with "sam build --beta-features". [y/N]: y
   ```

Configuring AWS SAM to use with Rust Lambda functions

Step 1: Configure your AWS SAM template

Configure your AWS SAM template with the following:

- **Binary** – Optional. Specify when your template contains multiple Rust Lambda functions.

- **BuildMethod** – `rust-cargolambda`.

- **CodeUri** – path to your `Cargo.toml` file.

- **Handler** – `bootstrap`.

- **Runtime** – `provided.al2`.

To learn more about custom runtimes, see [Custom AWS Lambda runtimes](#) in the *AWS Lambda Developer Guide*.

Here is an example of a configured AWS SAM template:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Metadata:
      BuildMethod: rust-cargolambda
      BuildProperties: function_a
    Properties:
      CodeUri: ./rust_app
      Handler: bootstrap
      Runtime: provided.al2
...
```

Step 2: Use the AWS SAM CLI with your Rust Lambda function

Use any AWS SAM CLI command with your AWS SAM template. For more information, see [Using the AWS SAM CLI](#).

Examples

Hello World example

In this example, we build the sample Hello World application using Rust as our runtime.

First, we initialize a new serverless application using `sam init`. During the interactive flow, we select the **Hello World application** and choose the **Rust** runtime.

```
$ sam init
...
Which template source would you like to use?
        1 - AWS Quick Start Templates
        2 - Custom Template Location
Choice: 1

Choose an AWS Quick Start application template
        1 - Hello World Example
        2 - Multi-step workflow
        3 - Serverless API
        ...
Template: 1

Use the most popular runtime and package type? (Python and zip) [y/N]: ENTER

Which runtime would you like to use?
        1 - aot.dotnet7 (provided.al2)
        2 - dotnet6
        3 - dotnet5.0
        ...
        18 - python3.7
        19 - python3.10
        20 - ruby2.7
        21 - rust (provided.al2)
Runtime: 21

Based on your selections, the only Package type available is Zip.
We will proceed to selecting the Package type as Zip.

Based on your selections, the only dependency manager available is cargo.
We will proceed copying the template using cargo.

Would you like to enable X-Ray tracing on the function(s) in your application?  [y/
N]: ENTER
```

```
Would you like to enable monitoring using CloudWatch Application Insights?
For more info, please view https://docs.aws.amazon.com/AmazonCloudWatch/latest/
monitoring/cloudwatch-application-insights.html [y/N]: ENTER

Project name [sam-app]: hello-rust


    ----------------------
    Generating application:
    ----------------------
    Name: hello-rust
    Runtime: rust (provided.al2)
    Architectures: x86_64
    Dependency Manager: cargo
    Application Template: hello-world
    Output Directory: .
    Configuration file: hello-rust/samconfig.toml


    Next steps can be found in the README file at hello-rust/README.md


Commands you can use next
=========================
[*] Create pipeline: cd hello-rust && sam pipeline init --bootstrap
[*] Validate SAM template: cd hello-rust && sam validate
[*] Test Function in the Cloud: cd hello-rust && sam sync --stack-name {stack-name} --
watch
```

The following is the structure of our Hello World application:

```
hello-rust
### README.md
### events
#   ### event.json
### rust_app
#   ### Cargo.toml
#   ### src
#       ### main.rs
### samconfig.toml
### template.yaml
```

In our AWS SAM template, our Rust function is defined as the following:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Metadata:
      BuildMethod: rust-cargolambda
    Properties:
      CodeUri: ./rust_app
      Handler: bootstrap
      Runtime: provided.al2
      Architectures:
        - x86_64
      Events:
        HelloWorld:
          Type: Api
            Path: /hello
            Method: get
```

Next, we run `sam  build` to build our application and prepare for deployment. The AWS SAM CLI creates a `.aws-sam` directory and organizes our build artifacts there. Our function is built using Cargo Lambda and stored as an executable binary at `.aws-sam/build/HelloWorldFunction/bootstrap`.

```
hello-rust$ sam build
Starting Build use cache
Build method "rust-cargolambda" is a beta feature.
Please confirm if you would like to proceed
You can also enable this beta feature with "sam build --beta-features". [y/N]: y

Experimental features are enabled for this session.
Visit the docs page to learn more about the AWS Beta terms https://aws.amazon.com/
service-terms/.

Cache is invalid, running build and copying resources for following functions
 (HelloWorldFunction)
Building codeuri: /Users/.../hello-rust/rust_app runtime: provided.al2 metadata:
 {'BuildMethod': 'rust-cargolambda'} architecture: x86_64 functions: HelloWorldFunction
Running RustCargoLambdaBuilder:CargoLambdaBuild
Running RustCargoLambdaBuilder:RustCopyAndRename
```

```
Build Succeeded

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


Commands you can use next
=========================
[*] Validate SAM template: sam validate
[*] Invoke Function: sam local invoke
[*] Test Function in the Cloud: sam sync --stack-name {{stack-name}} --watch
[*] Deploy: sam deploy --guided
```

Next, we deploy our application using sam deploy --guided.

```
hello-rust$ sam deploy --guided

Configuring SAM deploy
======================

        Looking for config file [samconfig.toml] :  Found
        Reading default arguments  :  Success

        Setting default arguments for 'sam deploy'
        =========================================
        Stack Name [hello-rust]: ENTER
        AWS Region [us-west-2]: ENTER
        #Shows you resources changes to be deployed and require a 'Y' to initiate
 deploy
        Confirm changes before deploy [Y/n]: ENTER
        #SAM needs permission to be able to create roles to connect to the resources in
 your template
        Allow SAM CLI IAM role creation [Y/n]: ENTER
        #Preserves the state of previously provisioned resources when an operation
 fails
        Disable rollback [y/N]: ENTER
        HelloWorldFunction may not have authorization defined, Is this okay? [y/N]: y
        Save arguments to configuration file [Y/n]: ENTER
        SAM configuration file [samconfig.toml]: ENTER
        SAM configuration environment [default]: ENTER

        Looking for resources needed for deployment:

        ...
```

```
        Uploading to hello-rust/56ba6585d80577dd82a7eaaee5945c0b  817973 / 817973
 (100.00%)


        Deploying with following values
        ===============================
        Stack name                   : hello-rust
        Region                       : us-west-2
        Confirm changeset            : True
        Disable rollback             : False
        Deployment s3 bucket         : aws-sam-cli-managed-default-
samclisourcebucket-1a4x26zbcdkqr
        Capabilities                 : ["CAPABILITY_IAM"]
        Parameter overrides          : {}
        Signing Profiles             : {}


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


        Uploading to hello-rust/a4fc54cb6ab75dd0129e4cdb564b5e89.template  1239 / 1239
 (100.00%)



Waiting for changeset to be created..

CloudFormation stack changeset
-------------------------------------------------------------------------------------------------
Operation                  LogicalResourceId          ResourceType
 Replacement
-------------------------------------------------------------------------------------------------
+ Add                      HelloWorldFunctionHelloW    AWS::Lambda::Permission    N/A

                           orldPermissionProd

...
-------------------------------------------------------------------------------------------------


Changeset created successfully. arn:aws:cloudformation:us-
west-2:012345678910:changeSet/samcli-deploy1681427201/
f0ef1563-5ab6-4b07-9361-864ca3de6ad6



Previewing CloudFormation changeset before deployment
=====================================================
```

```
Deploy this changeset? [y/N]: y


2023-04-13 13:07:17 - Waiting for stack create/update to complete


CloudFormation events from stack operations (refresh every 5.0 seconds)
-------------------------------------------------------------------------------
ResourceStatus            ResourceType              LogicalResourceId
  ResourceStatusReason
-------------------------------------------------------------------------------
CREATE_IN_PROGRESS        AWS::IAM::Role            HelloWorldFunctionRole    -


CREATE_IN_PROGRESS        AWS::IAM::Role            HelloWorldFunctionRole
  Resource creation
...
-------------------------------------------------------------------------------


CloudFormation outputs from deployed stack
-------------------------------------------------------------------------------
Outputs


-------------------------------------------------------------------------------
Key                 HelloWorldFunctionIamRole


Description         Implicit IAM Role created for Hello World function


Value               arn:aws:iam::012345678910:role/hello-rust-
HelloWorldFunctionRole-10II2P13AUDUY


Key                 HelloWorldApi


Description         API Gateway endpoint URL for Prod stage for Hello World function


Value               https://ggdxec9le9.execute-api.us-west-2.amazonaws.com/Prod/hello/


Key                 HelloWorldFunction


Description         Hello World Lambda Function ARN


Value               arn:aws:lambda:us-west-2:012345678910:function:hello-rust-
HelloWorldFunction-
yk4HzGzYeZBj


-------------------------------------------------------------------------------
```

```
Successfully created/updated stack - hello-rust in us-west-2
```

To test, we can invoke our Lambda function using the API endpoint.

```
$ curl https://ggdxec9le9.execute-api.us-west-2.amazonaws.com/Prod/hello/
Hello World!%
```

To test our function locally, first we ensure our function's `Architectures` property matches our local machine.

```
...
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function # More info about Function Resource:
 https://github.com/awslabs/serverless-application-model/blob/master/
versions/2016-10-31.md#awsserverlessfunction
    Metadata:
      BuildMethod: rust-cargolambda # More info about Cargo Lambda: https://github.com/
cargo-lambda/cargo-lambda
    Properties:
      CodeUri: ./rust_app   # Points to dir of Cargo.toml
      Handler: bootstrap     # Do not change, as this is the default executable name
 produced by Cargo Lambda
      Runtime: provided.al2
      Architectures:
        - arm64
...
```

Since we modified our architecture from `x86_64` to `arm64` in this example, we run `sam build` to update our build artifacts. We then run `sam local invoke` to locally invoke our function.

```
hello-rust$ sam local invoke
Invoking bootstrap (provided.al2)
Local image was not found.
Removing rapid images for repo public.ecr.aws/sam/emulation-provided.al2
Building
 image...............................................................................................
Using local image: public.ecr.aws/lambda/provided:al2-rapid-arm64.
```

```
Mounting /Users/.../hello-rust/.aws-sam/build/HelloWorldFunction as /var/
task:ro,delegated, inside runtime container
START RequestId: fbc55e6e-0068-45f9-9f01-8e2276597fc6 Version: $LATEST
{"statusCode":200,"body":"Hello World!"}END RequestId:
 fbc55e6e-0068-45f9-9f01-8e2276597fc6
REPORT RequestId: fbc55e6e-0068-45f9-9f01-8e2276597fc6  Init Duration: 0.68 ms
 Duration: 130.63 ms     Billed Duration: 131 ms     Memory Size: 128 MB     Max Memory
 Used: 128 MB
```

Single Lambda function project

Here is an example of a serverless application containing one Rust Lambda function.

Project directory structure:

```
.
### Cargo.lock
### Cargo.toml
### src
#   ### main.rs
### template.yaml
```

AWS SAM template:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Metadata:
      BuildMethod: rust-cargolambda
    Properties:
      CodeUri: ./
      Handler: bootstrap
      Runtime: provided.al2
...
```

Multiple Lambda function project

Here is an example of a serverless application containing multiple Rust Lambda functions.

Project directory structure:

```
.
### Cargo.lock
### Cargo.toml
### src
#    ### function_a.rs
#    ### function_b.rs
### template.yaml
```

AWS SAM template:

```yaml
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  FunctionA:
    Type: AWS::Serverless::Function
    Metadata:
      BuildMethod: rust-cargolambda
      BuildProperties:
        Binary: function_a
    Properties:
      CodeUri: ./
      Handler: bootstrap
      Runtime: provided.al2
  FunctionB:
    Type: AWS::Serverless::Function
    Metadata:
      BuildMethod: rust-cargolambda
      BuildProperties:
        Binary: function_b
    Properties:
      CodeUri: ./
      Handler: bootstrap
      Runtime: provided.al2
```

`Cargo.toml` file:

```toml
[package]
name = "test-handler"
version = "0.1.0"
edition = "2021"
```

```
[dependencies]
lambda_runtime = "0.6.0"
serde = "1.0.136"
tokio = { version = "1", features = ["macros"] }
tracing = { version = "0.1", features = ["log"] }
tracing-subscriber = { version = "0.3", default-features = false, features = ["fmt"] }

[[bin]]
name = "function_a"
path = "src/function_a.rs"

[[bin]]
name = "function_b"
path = "src/function_b.rs"
```

Building layers

You can use AWS SAM to build custom layers. For information about layers, see AWS Lambda layers in the *AWS Lambda Developer Guide*.

To build a custom layer, declare it in your AWS Serverless Application Model (AWS SAM) template file and include a `Metadata` resource attribute section with a `BuildMethod` entry. Valid values for `BuildMethod` are identifiers for an AWS Lambda runtime, or `makefile`. Include a `BuildArchitecture` entry to specify the instruction set architectures that your layer supports. Valid values for `BuildArchitecture` are Lambda instruction set architectures.

If you specify `makefile`, provide the custom makefile, where you declare a build target of the form `build-`*`layer-logical-id`* that contains the build commands for your layer. Your makefile is responsible for compiling the layer if necessary, and copying the build artifacts into the proper location required for subsequent steps in your workflow. The location of the makefile is specified by the `ContentUri` property of the layer resource, and must be named `Makefile`.

> ⓘ **Note**
>
> When you create a custom layer, AWS Lambda depends on environment variables to find your layer code. Lambda runtimes include paths in the `/opt` directory where your layer code is copied into. Your project's build artifact folder structure must match the runtime's expected folder structure so your custom layer code can be found.
>
> For example, for Python you can place your code in the `python/` subdirectory. For NodeJS, you can place your code in the `nodejs/node_modules/` subdirectory.

> For more information, see [Including library dependencies in a layer](#) in the *AWS Lambda Developer Guide*.

The following is an example `Metadata` resource attribute section.

```
Metadata:
  BuildMethod: python3.8
  BuildArchitecture: arm64
```

> ⓘ **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](#) property of the `AWS::Serverless::LayerVersion` resource type.

When you include the `Metadata` resource attribute section, you can use the [sam build](#) command to build the layer, both as an independent object, or as a dependency of an AWS Lambda function.

- **As an independent object.** You might want to build just the layer object, for example when you're locally testing a code change to the layer and don't need to build your entire application. To build the layer independently, specify the layer resource with the `sam build` *layer-logical-id* command.

- **As a dependency of a Lambda function.** When you include a layer's logical ID in the `Layers` property of a Lambda function in the same AWS SAM template file, the layer is a dependency of that Lambda function. When that layer also includes a `Metadata` resource attribute section with a `BuildMethod` entry, you build the layer either by building the entire application with the `sam build` command or by specifying the function resource with the `sam build` *function-logical-id* command.

Examples

Template example 1: Build a layer against the Python 3.9 runtime environment

The following example AWS SAM template builds a layer against the Python 3.9 runtime environment.

```
Resources:
  MyLayer:
    Type: AWS::Serverless::LayerVersion
    Properties:
      ContentUri: my_layer
      CompatibleRuntimes:
        - python3.9
    Metadata:
      BuildMethod: python3.9   # 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. Note that the `ContentUri` property is set to `my_layer`, so the makefile must be located in the root of the `my_layer` subdirectory, and the filename must be `Makefile`. Note also that the build artifacts are copied into the `python/` subdirectory so that AWS Lambda will be able to find the layer code.

```
build-MyLayer:
  mkdir -p "$(ARTIFACTS_DIR)/python"
  cp *.py "$(ARTIFACTS_DIR)/python"
  python -m pip install -r requirements.txt -t "$(ARTIFACTS_DIR)/python"
```

Example sam build commands

The following `sam build` commands build layers that include the `Metadata` resource attribute sections.

```
# Build the 'layer-logical-id' resource independently
$ sam build layer-logical-id

# Build the 'function-logical-id' resource and layers that this function depends on
$ sam build function-logical-id

# Build the entire application, including the layers that any function depends on
$ sam build
```

Building custom runtimes

You can use the <u>sam build</u> command to build custom runtimes required for your Lambda function. You declare your Lambda function to use a custom runtime by specifying `Runtime: provided` for the function.

To build a custom runtime, declare the `Metadata` resource attribute with a `BuildMethod: makefile` entry. You provide a custom makefile, where you declare a build target of the form `build-function-logical-id` that contains the build commands for your runtime. Your makefile is responsible for compiling the custom runtime if necessary, and copying the build artifacts into the proper location required for subsequent steps in your workflow. The location of the makefile is specified by the `CodeUri` property of the function resource, and must be named `Makefile`.

Examples

Example 1: Custom runtime for a function written in Rust

> ⓘ **Note**
>
> We recommend building Lambda functions with Cargo Lambda. To learn more, see Building Rust Lambda functions with Cargo Lambda.

The following AWS SAM template declares a function that uses a custom runtime for a Lambda function written in Rust, and instructs `sam build` to execute the commands for the `build-HelloRustFunction` build target.

```
Resources:
  HelloRustFunction:
    Type: AWS::Serverless::Function
    Properties:
      FunctionName: HelloRust
      Handler: bootstrap.is.real.handler
      Runtime: provided
      MemorySize: 512
      CodeUri: .
    Metadata:
      BuildMethod: makefile
```

The following makefile contains the build target and commands that will be executed. Note that the `CodeUri` property is set to `.`, so the makefile must be located in the project root directory (that is, the same directory as the application's AWS SAM template file). The filename must be `Makefile`.

```
build-HelloRustFunction:
  cargo build --release --target x86_64-unknown-linux-musl
  cp ./target/x86_64-unknown-linux-musl/release/bootstrap $(ARTIFACTS_DIR)
```

For more information about setting up your development environment in order to execute the `cargo build` command in the previous `makefile`, see the [Rust Runtime for AWS Lambda](#) blog post.

Example 2: Makefile builder for Python3.12 (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 Python3.12 runtime with a makefile builder.

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: hello_world/
```

```
      Handler: app.lambda_handler
      Runtime: python3.12
    Metadata:
      BuildMethod: makefile
```

The following makefile contains the build target and commands that will be executed. Note that the `CodeUri` property is set to `hello_world`, so the makefile must be located in the root of the `hello_world` subdirectory, and the filename must be `Makefile`.

```
build-HelloWorldFunction:
    cp *.py $(ARTIFACTS_DIR)
    cp requirements.txt $(ARTIFACTS_DIR)
    python -m pip install -r requirements.txt -t $(ARTIFACTS_DIR)
    rm -rf $(ARTIFACTS_DIR)/bin
```

Testing and debugging serverless applications

With the AWS SAM command line interface (CLI), you can locally test and "step-through" debug your serverless applications before uploading your application to the AWS Cloud. You can verify whether your application is behaving as expected, debug what's wrong, and fix any issues, before going through the steps of packaging and deploying your application.

When you locally invoke a Lambda function in debug mode within the AWS SAM CLI, you can then attach a debugger to it. With the debugger, you can step through your code line by line, see the values of various variables, and fix issues the same way you would for any other application.

> **ⓘ Note**
>
> If your application includes one or more layers, when you locally run and debug your application the layers package is downloaded and cached on your local host. For more information, see How layers are cached locally.

Topics

- Invoking Lambda functions locally
- Running API Gateway locally
- Integrating with automated tests
- Generating sample event payloads
- Step-through debugging Lambda functions locally
- Passing additional runtime debug arguments
- Validate your AWS SAM applications with AWS CloudFormation Linter

Invoking Lambda functions locally

You can invoke your AWS Lambda function locally by using the `sam local invoke` AWS SAM CLI command and providing the function's logical ID and an event file. Alternatively, **sam local invoke** also accepts `stdin` as an event. For more information about events, see Event in the *AWS Lambda Developer Guide*. For information about event message formats from different AWS services, see Using AWS Lambda with other services in the *AWS Lambda Developer Guide*.

> ℹ️ **Note**
>
> The **sam local invoke** command corresponds to the AWS Command Line Interface (AWS CLI) command <u>aws lambda invoke</u>. You can use either command to invoke a Lambda function.

You must run the **sam local invoke** command in the project directory that contains the function that 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
```

Environment variable file

To declare environment variables locally that override values defined in your templates, do the following:

1. Create a JSON file that contains the environment variables to override.
2. Use the `--env-vars` argument to override values defined in your templates.

Declaring environment variables

To declare environment variables that apply globally to all resources, specify a `Parameters` object like the following:

```
{
    "Parameters": {
        "TABLE_NAME": "localtable",
        "BUCKET_NAME": "testBucket",
        "STAGE": "dev"
```

```
        }
 }
```

To declare different environment variables for each resource, specify objects for each resource like the following:

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

When specifying objects for each resource, you can use the following identifiers, listed in order of highest to lowest precedence:

1. `logical_id`

2. `function_id`

3. `function_name`

4. Full path identifier

You can use both of the preceding methods of declaring environment variables together in a single file. When doing so, environment variables that you provided for specific resources take precedence over global environment variables.

Save your environment variables in a JSON file, such as `env.json`.

Overriding environment variable values

To override environment variables with those defined in your JSON file, use the `--env-vars` argument with the **invoke** or **start-api** commands. For example:

```
sam local invoke --env-vars env.json
```

Layers

If your application includes layers, for information about how to debug issues with layers on your local host, see [Working with layers](#).

Learn more

For a hands-on example of invoking functions locally, see [Module 2 - Run locally](#) in *The Complete AWS SAM Workshop*.

Running API Gateway locally

To start a local instance of Amazon API Gateway that you can use to test HTTP request/response functionality, use the [`sam local start-api`](#) AWS SAM CLI command. This functionality features hot reloading so that you can quickly develop and iterate over your functions.

> **ⓘ Note**
>
> *Hot reloading* is when only the files that changed are refreshed, and the state of the application remains the same. In contrast, *live reloading* is when the entire application is refreshed, and the state of the application is lost.

For instructions on using the `sam local start-api` command, see [Using sam local start-api](#).

By default, AWS SAM uses AWS Lambda proxy integrations and supports both `HttpApi` and `Api` resource types. For more information about proxy integrations for `HttpApi` resource types, see [Working with AWS Lambda proxy integrations for HTTP APIs](#) in the *API Gateway Developer Guide*. For more information about proxy integrations with `Api` resource types, see [Understand API Gateway Lambda proxy integration](#) in the *API Gateway Developer Guide*.

Example:

```
$ sam local start-api
```

AWS SAM automatically finds any functions within your AWS SAM template that have `HttpApi` or `Api` event sources defined. Then, it mounts the function at the defined HTTP paths.

In the following `Api` example, the `Ratings` function mounts `ratings.py:handler()` at `/ratings` for GET requests:

```
Ratings:
  Type: AWS::Serverless::Function
  Properties:
    Handler: ratings.handler
    Runtime: python3.9
    Events:
      Api:
        Type: Api
        Properties:
          Path: /ratings
          Method: get
```

Here is an example `Api` response:

```
// 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"
    });
}
```

If you modify your function's code, run the `sam build` command for `sam local start-api` to detect your changes.

Environment variable file

To declare environment variables locally that override values defined in your templates, do the following:

1. Create a JSON file that contains the environment variables to override.
2. Use the `--env-vars` argument to override values defined in your templates.

Declaring environment variables

To declare environment variables that apply globally to all resources, specify a `Parameters` object like the following:

```
{
```

```
    "Parameters": {
        "TABLE_NAME": "localtable",
        "BUCKET_NAME": "testBucket",
        "STAGE": "dev"
    }
}
```

To declare different environment variables for each resource, specify objects for each resource like the following:

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

When specifying objects for each resource, you can use the following identifiers, listed in order of highest to lowest precedence:

1. `logical_id`
2. `function_id`
3. `function_name`
4. Full path identifier

You can use both of the preceding methods of declaring environment variables together in a single file. When doing so, environment variables that you provided for specific resources take precedence over global environment variables.

Save your environment variables in a JSON file, such as `env.json`.

Overriding environment variable values

To override environment variables with those defined in your JSON file, use the `--env-vars` argument with the **invoke** or **start-api** commands. For example:

```
$ sam local start-api --env-vars env.json
```

Layers

If your application includes layers, for information about how to debug issues with layers on your local host, see [Working with layers](#).

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=15,
            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 variety of AWS toolkits and debuggers to test and debug your serverless applications locally.

For example, you can perform local step-through debugging of your Lambda functions by setting breakpoints, inspecting variables, and executing function code one line at a time. Local step-through debugging 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 integrated development environment (IDE) plugins that provide you with the ability to perform many common debugging tasks, like setting breakpoints, inspecting

variables, and executing function code one line at a time. AWS Toolkits make it easier for you to develop, debug, and deploy serverless applications that are built using AWS SAM. They provide an experience for building, testing, debugging, deploying, and invoking Lambda functions that's integrated into your IDE.

For more information about AWS Toolkits that you can use with AWS SAM, see the following:

- AWS Toolkit for Visual Studio Code
- AWS Cloud9
- AWS Toolkit for JetBrains

There are a variety AWS Toolkits that work with different combinations of IDEs and runtimes. The following table lists common IDE/runtime combinations that support step-through debugging of AWS SAM applications:

IDE	Runtime	AWS Toolkit	Instructions for step-through debugging
Visual Studio Code	- Node.js - Python .NET - Java - Go	AWS Toolkit for Visual Studio Code	Working with AWS Serverless Application in the *AWS Toolkit for Visual Studio Code User Guide*
AWS Cloud9	- Node.js - Python	AWS Cloud9, with AWS Toolkit enabled[1]	Working with AWS serverless applications using the AWS Toolkit in the *AWS Cloud9 User Guide*.
WebStorm	Node.js	AWS Toolkit for JetBrains[2]	Running (invoking) or debugging a local function in the *AWS Toolkit for JetBrains*
PyCharm	Python	AWS Toolkit for JetBrains[2]	Running (invoking) or debugging a local

IDE	Runtime	AWS Toolkit	Instructions for step-through debugging
			function in the *AWS Toolkit for JetBrains*
Rider	.NET	AWS Toolkit for JetBrains[2]	Running (invoking) or debugging a local function in the *AWS Toolkit for JetBrains*
IntelliJ	Java	AWS Toolkit for JetBrains[2]	Running (invoking) or debugging a local function in the *AWS Toolkit for JetBrains*
GoLand	Go	AWS Toolkit for JetBrains[2]	Running (invoking) or debugging a local function in the *AWS Toolkit for JetBrains*

Notes:

1. To use AWS Cloud9 to step-through debug AWS SAM applications, the AWS Toolkit must be enabled. For more information, see Enabling the AWS Toolkit in the *AWS Cloud9 User Guide*.
2. To use the AWS Toolkit for JetBrains to step-through debug AWS SAM applications, you must first install and configure it by following the instructions found in Installing the AWS Toolkit for JetBrains in the *AWS Toolkit for JetBrains*.

Running AWS SAM locally in debug mode

In addition to integrating with AWS Toolkits, you can also run AWS SAM in "debug mode" to attach to third-party debuggers like ptvsd or delve.

To run AWS SAM in debug mode, use commands sam local invoke or sam local start-api with the `--debug-port` or `-d` option.

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.

Passing additional runtime debug arguments

To pass additional runtime arguments when you're debugging your function, use the environment
variable DEBUGGER_ARGS. This passes a string of arguments directly into the run command that
the AWS SAM CLI uses to start your function.

For example, if you want to load a debugger like iKPdb at the runtime of your Python function,
you could pass the following as DEBUGGER_ARGS: `-m ikpdb --ikpdb-port=5858 --ikpdb-working-directory=/var/task/ --ikpdb-client-working-directory=/myApp --ikpdb-address=0.0.0.0`. This would load iKPdb at runtime with the other arguments you've
specified.

In this case, your full AWS SAM CLI command would be:

```
DEBUGGER_ARGS="-m ikpdb --ikpdb-port=5858 --ikpdb-working-directory=/var/task/ --ikpdb-
client-working-directory=/myApp --ikpdb-address=0.0.0.0" echo {} | sam local invoke -d
 5858 myFunction
```

You can pass debugger arguments to the functions of all runtimes.

Validate your AWS SAM applications with AWS CloudFormation Linter

AWS CloudFormation Linter (cfn-lint) is an open-source tool that you can use to perform detailed validation on your AWS CloudFormation templates. Cfn-lint contains rules that are guided by the AWS CloudFormation resource specification. Use cfn-lint to compare your resources against those rules to receive detailed messages on errors, warnings, or informational suggestions. Alternatively, create your own custom rules to validate against. To learn more about cfn-lint, see cfn-lint in the *AWS CloudFormation GitHub repository*.

You can use cfn-lint to validate your AWS Serverless Application Model (AWS SAM) templates through the AWS SAM Command Line Interface (AWS SAM CLI) by running **sam validate** with the **--lint** option.

```
sam validate --lint
```

To customize cfn-lint behavior, such as creating custom rules or specifying validation options, you can define a configuration file. To learn more, see Config File in the *cfn-lint AWS CloudFormation GitHub repository*. When you run **sam validate --lint**, cfn-lint behavior defined in your configuration file will be applied.

Examples

Perform cfn-lint validation on an AWS SAM template

```
sam validate --lint --template myTemplate.yaml
```

Learn more

To learn more about the **sam validate** command, see sam validate.

Deploying serverless applications

AWS SAM uses AWS CloudFormation as the underlying deployment mechanism. For more information, see What is AWS CloudFormation? in the *AWS CloudFormation User Guide*. The standard inputs to deploying serverless applications are the build artifacts created using the **sam build** command. For more information about **sam build**, see Building serverless applications.

You can deploy your application manually using AWS SAM command line interface (CLI) commands. You can also automate the deployments of your application using a continuous integration and continuous deployment (CI/CD) system. You can use many common CI/CD systems for deploying AWS SAM applications, including AWS CodePipeline, Jenkins, GitLab CI/CD, and GitHub Actions.

Deploying using CI/CD systems

AWS SAM helps organizations create pipelines for their preferred CI/CD systems, so that they can realize the benefits of CI/CD with minimal effort, such as accelerating deployment frequency, shortening lead time for changes, and reducing deployment errors.

AWS SAM simplifies CI/CD tasks for serverless applications with the help of build container images. The images that AWS SAM provides include the AWS SAM CLI and build tools for a number of supported AWS Lambda runtimes. This makes it easier to build and package serverless applications using the AWS SAM CLI. These images also alleviate the need for teams to create and manage their own images for CI/CD systems. For more information about AWS SAM build container images, see Image repositories.

Multiple CI/CD systems support AWS SAM build container images. Which CI/CD system you should use depends on several factors. These include whether your application uses a single runtime or multiple runtimes, or whether you want to build your application within a container image or directly on a host machine, either a virtual machine (VM) or bare metal host.

AWS SAM also provides a set of default pipeline templates for multiple CI/CD systems that encapsulate AWS's deployment best practices. These default pipeline templates use standard JSON/YAML pipeline configuration formats, and the built-in best practices help perform multi-account and multi-region deployments, and verify that pipelines cannot make unintended changes to infrastructure.

You have two main options for using AWS SAM to deploy your serverless applications: 1) Modify your existing pipeline configuration to use AWS SAM CLI commands, or 2) Generate an example CI/CD pipeline configuration that you can use as a starting point for your own application.

For more information about these options, see the following topics:

- [Modifying your existing CI/CD pipelines](#)
- [Generating starter CI/CD pipelines](#)

Deploying using the AWS SAM CLI

After you develop and test your serverless application locally, you can deploy your application using the **sam deploy** command.

To have AWS SAM 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 either to Amazon Simple Storage Service (Amazon S3) (for .zip file archives) or to Amazon Elastic Container Registry (Amazon ECR) (for container images). The command then deploys your application to the AWS Cloud.

Example:

```
# Deploy an application using prompts:
sam deploy --guided
```

Troubleshooting deployments using the AWS SAM CLI

AWS SAM CLI error: "Security Constraints Not Satisfied"

When running **sam deploy --guided**, you're 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 might have an Amazon API Gateway API configured without authorization. By responding **N** to this prompt, you're saying that this is not OK.

To fix this, you have the following options:

- Configure your application with authorization. For information about configuring authorization, see Controlling access to API Gateway APIs.
- Respond to this question with **Y** to indicate that you're OK with deploying an application that has an API Gateway API configured without authorization.

Gradual deployments

If you want to deploy your AWS SAM application gradually rather than all at once, you can specify deployment configurations that AWS CodeDeploy provides. For more information, see Working with deployment configurations in CodeDeploy in the *AWS CodeDeploy User Guide*.

For information about configuring your AWS SAM application to deploy gradually, see Deploying serverless applications gradually.

Learn more

For hands-on examples of deploying serverless applications, see the following from *The Complete AWS SAM Workshop*:

- Module 3 - Deploy manually – Learn how to build, package, and deploy a serverless application using the AWS SAM CLI.
- Module 4 - CI/CD – Learn how to automate the build, package, and deployment phases by creating a *continuous integration and delivery (CI/CD)* pipeline.

Modifying your existing CI/CD pipelines

The procedures for your existing CI/CD pipeline to deploy serverless applications using AWS SAM are slightly different depending on which CI/CD system you are using.

The following topics provide examples for configuring your CI/CD system to build serverless applications within an AWS SAM build container image:

Topics

- [Deploying using AWS CodePipeline](#)

- [Deploying using Bitbucket Pipelines](#)

- [Deploying using Jenkins](#)

- [Deploying using GitLab CI/CD](#)

- [Deploying using GitHub Actions](#)

Deploying using AWS CodePipeline

To configure your [AWS CodePipeline](#) pipeline to automate the build and deployment of your AWS
SAM application, your AWS CloudFormation template and `buildspec.yml` file must contain lines
that do the following:

1. Reference a build container image with the necessary runtime from the available images. The
   following example uses the `public.ecr.aws/sam/build-nodejs20.x` build container
   image.

2. Configure the pipeline stages to run the necessary AWS SAM command line interface (CLI)
   commands. The following example runs two AWS SAM CLI commands: **sam build** and **sam
   deploy** (with necessary options).

This example assumes that you have declared all functions and layers in your AWS SAM template
file with `runtime: nodejs20.x`.

AWS CloudFormation template snippet:

```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
    Environment:
      ComputeType: BUILD_GENERAL1_SMALL
      Image: public.ecr.aws/sam/build-nodejs20.x
      Type: LINUX_CONTAINER
    ...
```

`buildspec.yml` snippet:

```
version: 0.2
```

```
phases:
  build:
    commands:
      - sam build
      - sam deploy --no-confirm-changeset --no-fail-on-empty-changeset
```

For a list of available Amazon Elastic Container Registry (Amazon ECR) build container images for different runtimes, see Image repositories.

Deploying using Bitbucket Pipelines

To configure your Bitbucket Pipeline to automate the build and deployment of your AWS SAM application, your `bitbucket-pipelines.yml` file must contain lines that do the following:

1. Reference a build container image with the necessary runtime from the available images. The following example uses the `public.ecr.aws/sam/build-nodejs20.x` build container image.

2. Configure the pipeline stages to run the necessary AWS SAM command line interface (CLI) commands. The following example runs two AWS SAM CLI commands: **sam build** and **sam deploy** (with necessary options).

This example assumes that you have declared all functions and layers in your AWS SAM template file with `runtime: nodejs20.x`.

```
image: public.ecr.aws/sam/build-nodejs20.x

pipelines:
  branches:
    main: # branch name
      - step:
          name: Build and Package
          script:
            - sam build
            - sam deploy --no-confirm-changeset --no-fail-on-empty-changeset
```

For a list of available Amazon Elastic Container Registry (Amazon ECR) build container images for different runtimes, see Image repositories.

Deploying using Jenkins

To configure your [Jenkins](#) pipeline to automate the build and deployment of your AWS SAM application, your `Jenkinsfile` must contain lines that do the following:

1. Reference a build container image with the necessary runtime from the available images. The following example uses the `public.ecr.aws/sam/build-nodejs20.x` build container image.

2. Configure the pipeline stages to run the necessary AWS SAM command line interface (CLI) commands. The following example runs two AWS SAM CLI commands: **sam build** and **sam deploy** (with necessary options).

This example assumes that you have declared all functions and layers in your AWS SAM template file with `runtime: nodejs20.x`.

```
pipeline {
    agent { docker { image 'public.ecr.aws/sam/build-nodejs20.x' } }
    stages {
        stage('build') {
            steps {
                sh 'sam build'
                sh 'sam deploy --no-confirm-changeset --no-fail-on-empty-changeset'
            }
        }
    }
}
```

For a list of available Amazon Elastic Container Registry (Amazon ECR) build container images for different runtimes, see [Image repositories](#).

Deploying using GitLab CI/CD

To configure your [GitLab](#) pipeline to automate the build and deployment of your AWS SAM application, your `gitlab-ci.yml` file must contain lines that do the following:

1. Reference a build container image with the necessary runtime from the available images. The following example uses the `public.ecr.aws/sam/build-nodejs20.x` build container image.

2. Configure the pipeline stages to run the necessary AWS SAM command line interface (CLI) commands. The following example runs two AWS SAM CLI commands: **sam build** and **sam deploy** (with necessary options).

This example assumes that you have declared all functions and layers in your AWS SAM template file with `runtime: nodejs20.x`.

```
image: public.ecr.aws/sam/build-nodejs20.x
deploy:
  script:
    - sam build
    - sam deploy --no-confirm-changeset --no-fail-on-empty-changeset
```

For a list of available Amazon Elastic Container Registry (Amazon ECR) build container images for different runtimes, see [Image repositories](#).

Deploying using GitHub Actions

To configure your [GitHub](#) pipeline to automate the build and deployment of your AWS SAM application, you must first install the AWS SAM command line interface (CLI) on your host. You can use [GitHub Actions](#) in your GitHub workflow to help with this setup.

The following example GitHub workflow sets up an Ubuntu host using a series of GitHub Actions, then runs AWS SAM CLI commands to build and deploy an AWS SAM application:

```
on:
  push:
    branches:
      - main
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-python@v3
      - uses: aws-actions/setup-sam@v2
      - uses: aws-actions/configure-aws-credentials@v1
        with:
          aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
          aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
```

```
          aws-region: us-east-2
      - run: sam build --use-container
      - run: sam deploy --no-confirm-changeset --no-fail-on-empty-changeset
```

For a list of available Amazon Elastic Container Registry (Amazon ECR) build container images for different runtimes, see Image repositories.

Generating starter CI/CD pipelines

When you are ready to deploy your serverless application in an automated manner, you can generate a deployment pipeline for your CI/CD system of choice. AWS SAM provides a set of starter pipeline templates with which you can generate pipelines in minutes using the sam pipeline init command.

The starter pipeline templates use the familiar JSON/YAML syntax of the CI/CD system, and incorporate best practices such as managing artifacts across multiple accounts and regions, and using the minimum amount of permissions required to deploy the application. Currently, the AWS SAM CLI supports generating starter CI/CD pipeline configurations for AWS CodePipeline, Jenkins, GitLab CI/CD, GitHub Actions, and Bitbucket Pipelines.

Here are the high-level tasks you need to perform to generate a starter pipeline configuration:

1. **Create infrastructure resources** – Your pipeline requires certain AWS resources, for example the IAM user and roles with necessary permissions, an Amazon S3 bucket, and optionally an Amazon ECR repository.

2. **Connect your Git repository with your CI/CD system** – Your CI/CD system needs to know which Git repository will trigger the pipeline to run. Note that this step may not be necessary, depending on which combination of Git repository and CI/CD system you are using.

3. **Generate your pipeline configuration** – This step generates a starter pipeline configuration that includes two deployment stages.

4. **Commit your pipeline configuration to your Git repository** – This step is necessary to ensure your CI/CD system is aware of your pipeline configuration, and will run when changes are committed.

After you've generated the starter pipeline configuration and committed it to your Git repository, whenever someone commits a code change to that repository your pipeline will be triggered to run automatically.

The ordering of these steps, and details of each step, vary based on your CI/CD system:

- If you are using AWS CodePipeline, see Generating starter pipeline for AWS CodePipeline.

- If you are using Jenkins, GitLab CI/CD, GitHub Actions, or Bitbucket Pipelines, see Generating starter pipelines for Jenkins, GitLab CI/CD, GitHub Actions, or Bitbucket Pipelines.

Generating starter pipeline for AWS CodePipeline

To generate a starter pipeline configuration for AWS CodePipeline, perform the following tasks in this order:

1. Create infrastructure resources

2. Generate the pipeline configuration

3. Commit your pipeline configuration to Git

4. Connect your Git repository with your CI/CD system

> ⓘ **Note**
>
> The following procedure utilizes two AWS SAM CLI commands, `sam pipeline bootstrap` and `sam pipeline init`. The reason there are two commands is to handle the use case where administrators (that is, users who need permission to set up infrastructure AWS resource like IAM users and roles) have more permission that developers (that is, users who just need permission to set up individual pipelines, but not the required infrastructure AWS resources).

Step 1: Create infrastructure resources

Pipelines that use AWS SAM require certain AWS resources, like an IAM user and roles with necessary permissions, an Amazon S3 bucket, and optionally an Amazon ECR repository. You must have a set of infrastructure resources for each deployment stage of the pipeline.

You can run the following command to help with this setup:

```
sam pipeline bootstrap
```

> **ⓘ Note**
>
> Run the previous command for each deployment stage of your pipeline.

Step 2: Generate the pipeline configuration

To generate the pipeline configuration, run the following command:

```
sam pipeline init
```

Step 3: Commit your pipeline configuration to Git repository

This step is necessary to ensure your CI/CD system is aware of your pipeline configuration, and will run when changes are committed.

Step 4: Connect your Git repository with your CI/CD system

For AWS CodePipeline you can now create the connection by running the following command:

```
sam deploy -t codepipeline.yaml --stack-name <pipeline-stack-name> --
capabilities=CAPABILITY_IAM --region <region-X>
```

If you are using GitHub or Bitbucket, after running the **sam deploy** command previously, complete the connection by following the steps under **To complete a connection** found on the [Update a pending connection](#) topic in the *Developer Tools console user guide*. In addition, store a copy of the CodeStarConnectionArn from the output of the **sam deploy** command, because you will need it if you want to use AWS CodePipeline with another branch than main.

Configuring other branches

By default, AWS CodePipeline uses the main branch with AWS SAM. If you want to use a branch other than main, you must run the **sam deploy** command again. Note that depending on which Git repository you are using, you may also need to provide the CodeStarConnectionArn:

```
# For GitHub and Bitbucket
sam deploy -t codepipeline.yaml --stack-name <feature-pipeline-stack-name> --
capabilities=CAPABILITY_IAM --parameter-overrides="FeatureGitBranch=<branch-name>
 CodeStarConnectionArn=<codestar-connection-arn>"
```

```
# For AWS CodeCommit
sam deploy -t codepipeline.yaml --stack-name <feature-pipeline-stack-name> --
capabilities=CAPABILITY_IAM --parameter-overrides="FeatureGitBranch=<branch-name>"
```

Learn more

For a hands-on example of setting up a CI/CD pipeline, see CI/CD with AWS CodePipeline in *The Complete AWS SAM Workshop*.

Generating starter pipelines for Jenkins, GitLab CI/CD, GitHub Actions, or Bitbucket Pipelines

To generate a starter pipeline configuration for Jenkins, GitLab CI/CD, GitHub Actions, or Bitbucket Pipelines perform the following tasks in this order:

1. Create infrastructure resources

2. Connect your Git repository with your CI/CD system

3. Create credential objects

4. Generate the pipeline configuration

5. Commit your pipeline configuration to Git repository

> ⓘ **Note**
>
> The following procedure utilizes two AWS SAM CLI commands, `sam pipeline bootstrap` and `sam pipeline init`. The reason there are two commands is to handle the use case where administrators (that is, users who need permission to set up infrastructure AWS resource like IAM users and roles) have more permission that developers (that is, users who just need permission to set up individual pipelines, but not the required infrastructure AWS resources).

Step 1: Create infrastructure resources

Pipelines that use AWS SAM require certain AWS resources, like an IAM user and roles with necessary permissions, an Amazon S3 bucket, and optionally an Amazon ECR repository. You must have a set of infrastructure resources for each deployment stage of the pipeline.

You can run the following command to help with this setup:

```
sam pipeline bootstrap
```

> **ⓘ Note**
>
> Run the previous command for each deployment stage of your pipeline.

You must capture the AWS credentials (key id and secret key) for the pipeline users for each deployment stage of your pipeline, because they are needed for subsequent steps.

Step 2: Connect your Git repository with your CI/CD system

Connecting your Git repository to your CI/CD system is necessary so that the CI/CD system is able to access your application source code for builds and deployments.

> **ⓘ Note**
>
> You can skip this step if you are using one of the following combinations, because the connection is done for you automatically:
>
> 1. GitHub Actions with GitHub repository
> 2. GitLab CI/CD with GitLab repository
> 3. Bitbucket Pipelines with a Bitbucket repository

To connect your Git repository with your CI/CD system, do one of the following:

- If you're using Jenkins, see the [Jenkins documentation](#) for "Adding a branch source."
- If you're using GitLab CI/CD and a Git repository other than GitLab, see the [GitLab documentation](#) for "connecting an external repository."

Step 3: Create credential objects

Each CI/CD system has its own way of managing credentials needed for the CI/CD system to access your Git repository.

To create the necessary credential objects, do one of the following:

- If you're using Jenkins, create a single "credential" that stores both the key id and secret key. Follow the instructions in the [Building a Jenkins Pipeline with AWS SAM](#) blog, in the **Configure Jenkins** section. You will need the "Credential id" for the next step.

- If you're using GitLab CI/CD, create two "protected variables", one for each of key id and secret key. Follow the instructions in the [GitLab documentation](#) – you will need two "variable keys" for the next step.

- If you're using GitHub Actions, create two "encrypted secrets", one for each of key and secret key. Follow the instructions in the [GitHub documentation](#) - you will need two "secret names" for the next step.

- If you're using Bitbucket Pipelines, create two "secure variables", one for each of key id and secret key. Follow the instructions in the [Variables and secrets](#) - you will need two "secret names" for the next step.

Step 4: Generate the pipeline configuration

To generate the pipeline configuration, run the following command. You will need to input the credential object that you created in the previous step:

```
sam pipeline init
```

Step 5: Commit your pipeline configuration to Git repository

This step is necessary to ensure your CI/CD system is aware of your pipeline configuration, and will run when changes are committed.

Learn more

For a hands-on example of setting up a CI/CD pipeline using GitHub Actions, see [CI/CD with GitHub](#) in *The Complete AWS SAM Workshop*.

Customizing starter pipelines

As a CI/CD administrator, you may want to customize a starter pipeline template, and associated guided prompts, that developers in your organization can use to create pipeline configurations.

The AWS SAM CLI uses Cookiecutter templates when creating starter templates. For details about cookie cutter templates, Cookiecutter.

You can also customize the prompts that the AWS SAM CLI displays to users when creating pipeline configurations using the `sam pipeline init` command. To customize user prompts, do the following:

1. **Create a `questions.json` file** – The `questions.json` file must be in the root of the project respository. This is the same directory as the `cookiecutter.json` file. To view the schema for the `questions.json` file, see questions.json.schema. To view an example `questions.json` file, see questions.json.

2. **Map question keys with cookiecutter names** – Each object in the `questions.json` file needs a key that matches a name in the cookiecutter template. This key matching is how the AWS SAM CLI maps user prompt responses to the cookie cutter template. To see examples of this key matching, see the Example files section later in this topic.

3. **Create a `metadata.json` file** – Declare the number of stages the pipeline will have in the `metadata.json` file. The number of stages instructs the `sam pipeline init` command how many stages to prompt information about, or in the case of the `--bootstrap` option, how many stages to create infrastructure resources for. To view an example `metadata.json` file that declares a pipeline with two stages, see metadata.json.

Example projects

Here are example projects, which each include a Cookiecutter template, a `questions.json` file, and a `metadata.json` file:

- Jenkins example: Two-stage Jenkins pipeline template
- CodePipeline example: Two stage CodePipeline pipeline template

Example files

The following set of files show how questions in the `questions.json` file are associated with entries in the Cookiecutter template file. Note that these examples are file snippets, not full files. To see examples of full files, see the Example projects section earlier in this topic.

Example **`questions.json`**:

```
{
  "questions": [{
    "key": "intro",
    "question": "\nThis template configures a pipeline that deploys a serverless
 application to a testing and a production stage.\n",
    "kind": "info"
  }, {
    "key": "pipeline_user_jenkins_credential_id",
    "question": "What is the Jenkins credential ID (via Jenkins plugin \"aws-
credentials\") for pipeline user access key?",
    "isRequired": true
  }, {
    "key": "sam_template",
    "question": "What is the template file path?",
    "default": "template.yaml"
  }, {
    ...
```

Example **cookiecutter.json**:

```
{
  "outputDir": "aws-sam-pipeline",
  "pipeline_user_jenkins_credential_id": "",
  "sam_template": "",
    ...
```

Example **Jenkinsfile**:

```
pipeline {
  agent any
  environment {
    PIPELINE_USER_CREDENTIAL_ID =
 '{{cookiecutter.pipeline_user_jenkins_credential_id}}'
    SAM_TEMPLATE = '{{cookiecutter.sam_template}}'
    ...
```

Using OIDC authentication with AWS SAM pipeline

AWS Serverless Application Model (AWS SAM) supports OpenID Connect (OIDC) user
authentication for Bitbucket, GitHub Actions, and GitLab continuous integration and continuous

delivery (CI/CD) platforms. With this support, you can use authorized CI/CD user accounts from any of these platforms to manage your serverless application pipelines. Otherwise, you would need to create and manage multiple AWS Identity and Access Management (IAM) users to control access to AWS SAM pipelines.

Set up OIDC with AWS SAM pipeline

During the `sam pipeline bootstrap` configuration process, do the following to set up OIDC with your AWS SAM pipeline.

1. When prompted to choose an identity provider, select **OIDC**.

2. Next, select a supported OIDC provider.

3. Enter the OIDC provider URL, beginning with **https://**.

> ⓘ **Note**
>
> AWS SAM references this URL when it generates the `AWS::IAM::OIDCProvider` resource type.

4. Next, follow the prompts and enter the CI/CD platform information needed to access the selected platform. These details vary by platform and can include:

   - OIDC client ID.

   - Code repository name or universally unique identifier (UUID).

   - Group or Organization name associated with the repository.

   - GitHub organization that the code repository belongs to.

   - GitHub repository name.

   - Branch that deployments will occur from.

5. AWS SAM displays a summary of the entered OIDC configuration. Enter the number for a setting to edit it, or press **Enter** to continue.

6. When prompted to confirm the creation of resources needed to support the entered OIDC connection, press **Y** to continue.

AWS SAM generates an `AWS::IAM::OIDCProvider` AWS CloudFormation resource with the provided configuration that assumes the pipeline execution role. To learn more about this AWS CloudFormation resource type, see AWS::IAM::OIDCProvider in the *AWS CloudFormation User Guide.*

> **ⓘ Note**
>
> If the identity provider (IdP) resource already exists in your AWS account, AWS SAM references it instead of creating a new resource.

Example

The following is an example of setting up OIDC with AWS SAM pipeline.

```
Select a permissions provider:
    1 - IAM (default)
    2 - OpenID Connect (OIDC)
Choice (1, 2): 2
Select an OIDC provider:
    1 - GitHub Actions
    2 - GitLab
    3 - Bitbucket
Choice (1, 2, 3): 1
Enter the URL of the OIDC provider [https://token.actions.githubusercontent.com]:
Enter the OIDC client ID (sometimes called audience) [sts.amazonaws.com]:
Enter the GitHub organization that the code repository belongs to. If there is no
 organization enter your username instead: my-org
Enter GitHub repository name: testing
Enter the name of the branch that deployments will occur from [main]:

[3] Reference application build resources
Enter the pipeline execution role ARN if you have previously created one, or we will
 create one for you []:
Enter the CloudFormation execution role ARN if you have previously created one, or we
 will create one for you []:
Please enter the artifact bucket ARN for your Lambda function. If you do not have a
 bucket, we will create one for you []:
Does your application contain any IMAGE type Lambda functions? [y/N]:

[4] Summary
Below is the summary of the answers:
    1 - Account: 123456
    2 - Stage configuration name: dev
    3 - Region: us-east-1
    4 - OIDC identity provider URL: https://token.actions.githubusercontent.com
```

Example   693

```
    5 - OIDC client ID: sts.amazonaws.com
    6 - GitHub organization: my-org
    7 - GitHub repository: testing
    8 - Deployment branch: main
    9 - Pipeline execution role: [to be created]
    10 - CloudFormation execution role: [to be created]
    11 - Artifacts bucket: [to be created]
    12 - ECR image repository: [skipped]
 Press enter to confirm the values above, or select an item to edit the value:

 This will create the following required resources for the 'dev' configuration:
    - IAM OIDC Identity Provider
    - Pipeline execution role
    - CloudFormation execution role
    - Artifact bucket
 Should we proceed with the creation? [y/N]:
```

Learn more

For more information on using OIDC with AWS SAM pipeline, see [sam pipeline bootstrap](#).

Using the AWS SAM CLI to upload local files at deployment

When developing, you will often find it beneficial to break up your application code into separate files to better organize and manage your application. A basic example of this is separating your AWS Lambda function code from your infrastructure code. You do this by organizing your Lambda function code in a subdirectory of your project and referencing its local path within your AWS Serverless Application Model (AWS SAM) template.

When you deploy your application to the AWS Cloud, AWS CloudFormation requires that your local files are first uploaded to an accessible AWS service, such as Amazon Simple Storage Service (Amazon S3). You can use the AWS SAM CLI to automatically facilitate this process. Use the `sam deploy` or `sam package` command to do the following:

1. Automatically upload your local files to an accessible AWS service.

2. Automatically update your application template to reference the new file path.

Topics

- [Demo: Use the AWS SAM CLI to upload Lambda function code](#)

- [Supported use cases](#)

- [Learn more](#)

Demo: Use the AWS SAM CLI to upload Lambda function code

In this demo, we initialize the sample Hello World application using a .zip package type for our Lambda function. We use the AWS SAM CLI to automatically upload our Lambda function code to Amazon S3 and reference its new path in our application template.

First, we run `sam init` to initialize our Hello World application.

```
$ sam init
...
Which template source would you like to use?
        1 - AWS Quick Start Templates
        2 - Custom Template Location
Choice: 1


Choose an AWS Quick Start application template
        1 - Hello World Example
        2 - Multi-step workflow
        ...
Template: 1


Use the most popular runtime and package type? (Python and zip) [y/N]: y


Would you like to enable X-Ray tracing on the function(s) in your application?  [y/
N]: ENTER


Would you like to enable monitoring using CloudWatch Application Insights?
For more info, please view https://docs.aws.amazon.com/AmazonCloudWatch/latest/
monitoring/cloudwatch-application-insights.html [y/N]: ENTER


Project name [sam-app]: demo


    ----------------------
    Generating application:
    ----------------------
    Name: demo
    Runtime: python3.9
    Architectures: x86_64
    Dependency Manager: pip
```

```
        Application Template: hello-world
        Output Directory: .
        Configuration file: demo/samconfig.toml


...
```

Our Lambda function code is organized in the `hello_world` subdirectory of our project.

```
demo
### README.md
### hello_world
#    ### __init__.py
#    ### app.py
#    ### requirements.txt
### template.yaml
### tests
```

Within our AWS SAM template, we reference the local path to our Lambda function code using the `CodeUri` property.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function # More info about Function Resource:
 https://github.com/awslabs/serverless-application-model/blob/master/
versions/2016-10-31.md#awsserverlessfunction
    Properties:
      CodeUri: hello_world/
      Handler: app.lambda_handler
      Runtime: python3.9
      ...
```

Next, we run `sam build` to build our application and prepare for deployment.

```
$ sam build
Starting Build use cache
Manifest file is changed (new hash: 3298f13049d19cffaa37ca931dd4d421) or dependency
  folder (.aws-sam/deps/7896875f-9bcc-4350-8adb-2c1d543627a1) is missing for
  (HelloWorldFunction), downloading dependencies and copying/building source
```

```
Building codeuri: /Users/.../demo/hello_world runtime: python3.9 metadata: {}
 architecture: x86_64 functions: HelloWorldFunction
Running PythonPipBuilder:CleanUp
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource
Running PythonPipBuilder:CopySource


Build Succeeded

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

Next, we run `sam deploy --guided` to deploy our application.

```
$ sam deploy --guided

Configuring SAM deploy
======================

        Looking for config file [samconfig.toml] :  Found
        Reading default arguments  :  Success

        Setting default arguments for 'sam deploy'
        =========================================
        Stack Name [demo]: ENTER
        AWS Region [us-west-2]: ENTER
        #Shows you resources changes to be deployed and require a 'Y' to initiate
 deploy
        Confirm changes before deploy [Y/n]: n
        #SAM needs permission to be able to create roles to connect to the resources in
 your template
        Allow SAM CLI IAM role creation [Y/n]: ENTER
        #Preserves the state of previously provisioned resources when an operation
 fails
        Disable rollback [y/N]: ENTER
        HelloWorldFunction may not have authorization defined, Is this okay? [y/N]: y
        Save arguments to configuration file [Y/n]: ENTER
        SAM configuration file [samconfig.toml]: ENTER
        SAM configuration environment [default]: ENTER

        Looking for resources needed for deployment:
        ...
```

```
        Saved arguments to config file
        Running 'sam deploy' for future deployments will use the parameters saved
  above.
        The above parameters can be changed by modifying samconfig.toml
        Learn more about samconfig.toml syntax at
        https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/
serverless-sam-cli-config.html


File with same data already exists at demo/da3c598813f1c2151579b73ad788cac8, skipping
  upload


        Deploying with following values
        ===============================
        Stack name                 : demo
        Region                     : us-west-2
        Confirm changeset          : False
        Disable rollback           : False
        Deployment s3 bucket       : aws-sam-cli-managed-default-
samclisourcebucket-1a4x26zbcdkqr
        Capabilities               : ["CAPABILITY_IAM"]
        Parameter overrides        : {}
        Signing Profiles           : {}


Initiating deployment
=====================
...
Waiting for changeset to be created..
CloudFormation stack changeset
-------------------------------------------------------------------------------------
Operation                LogicalResourceId      ResourceType           Replacement

-------------------------------------------------------------------------------------
+ Add                    HelloWorldFunctionHell  AWS::Lambda::Permissio  N/A

                         oWorldPermissionProd    n

+ Add                    HelloWorldFunctionRole  AWS::IAM::Role          N/A

...
-------------------------------------------------------------------------------------


Changeset created successfully. arn:aws:cloudformation:us-
west-2:012345678910:changeSet/samcli-deploy1680906292/1164338d-72e7-4593-a372-
  f2b3e67f542f
```

```
2023-04-07 12:24:58 - Waiting for stack create/update to complete


CloudFormation events from stack operations (refresh every 5.0 seconds)
-------------------------------------------------------------------------------
ResourceStatus            ResourceType              LogicalResourceId
  ResourceStatusReason
-------------------------------------------------------------------------------
CREATE_IN_PROGRESS        AWS::IAM::Role            HelloWorldFunctionRole    -

CREATE_IN_PROGRESS        AWS::IAM::Role            HelloWorldFunctionRole    Resource
  creation

                                                                            Initiated


...
-------------------------------------------------------------------------------
CloudFormation outputs from deployed stack
-------------------------------------------------------------------------------
Outputs


-------------------------------------------------------------------------------
Key                 HelloWorldFunctionIamRole

Description         Implicit IAM Role created for Hello World function

Value               arn:aws:iam::012345678910:role/demo-HelloWorldFunctionRole-
VQ4CU7UY7S2K

Key                 HelloWorldApi

Description         API Gateway endpoint URL for Prod stage for Hello World function

Value               https://satnon55e9.execute-api.us-west-2.amazonaws.com/Prod/hello/


Key                 HelloWorldFunction

Description         Hello World Lambda Function ARN

Value               arn:aws:lambda:us-west-2:012345678910:function:demo-

HelloWorldFunction-G14inKTmSQvK

-------------------------------------------------------------------------------
```

```
Successfully created/updated stack - demo in us-west-2
```

During deployment, the AWS SAM CLI automatically uploads our Lambda function code to Amazon S3 and updates our template. Our modified template in the AWS CloudFormation console reflects the Amazon S3 bucket path.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: s3://aws-sam-cli-managed-default-samclisourcebucket-1a4x26zbcdkqr/demo/
da3c598813f1c2151579b73ad788cac8
      Handler: app.lambda_handler
      ...
```

Supported use cases

The AWS SAM CLI can automatically facilitate this process for a number of file types, AWS CloudFormation resource types, and AWS CloudFormation macros.

File types

Application files and Docker images are supported.

AWS CloudFormation resource types

The following is a list of the supported resource types and their properties:

Resource	Properties
AWS::ApiGateway::RestApi	BodyS3Location
AWS::ApiGatewayV2::Api	BodyS3Location
AWS::AppSync:FunctionConfiguration	CodeS3Location RequestMappingTemplateS3Location

Resource	Properties
	ResponseMappingTemplateS3Location
AWS::AppSync::GraphQLSchema	DefinitionS3Location
AWS::AppSync::Resolver	CodeS3Location RequestMappingTemplateS3Location ResponseMappingTemplateS3Location
AWS::CloudFormation::Module Version	ModulePackage
AWS::CloudFormation::ResourceVersion	SchemaHandlerPackage
AWS::ECR::Repository	RepositoryName
AWS::ElasticBeanstalk::ApplicationVersion	SourceBundle
AWS::Glue::Job	Command.ScriptLocation
AWS::Lambda::Function	Code Code.ImageUri
AWS::Lambda::LayerVersion	Content
AWS::Serverless::Api	DefinitionUri
AWS::Serverless::Function	CodeUri ImageUri

Resource	Properties
AWS::Serverless::GraphQLApi	SchemaUri Function.CodeUri Resolver.CodeUri
AWS::Serverless::HttpApi	DefinitionUri
AWS::Serverless::LayerVersion	ContentUri
AWS::Serverless::StateMachine	DefinitionUri
AWS::StepFunctions::StateMachine	DefinitionS3Location

AWS CloudFormation macros

Files referenced using the AWS::Include transform macro are supported.

Learn more

To learn more about the AWS::Include transform, see AWS::Include transform in the *AWS CloudFormation User Guide.*

To see an example of using the AWS::Include transform in an AWS SAM template, see the API Gateway HTTP API to SQS pattern at *Serverless Land.*

Monitoring serverless applications

Configure and perform monitoring of your AWS Serverless Application Model (AWS SAM) applications.

Topics

- Monitor your serverless applications with CloudWatch Application Insights
- Working with logs

Monitor your serverless applications with CloudWatch Application Insights

Amazon CloudWatch Application Insights helps you monitor the AWS resources in your applications to help identify potential issues. It can analyze AWS resource data for signs of problems and build automated dashboards to visualize them. You can configure CloudWatch Application Insights to use with your AWS Serverless Application Model (AWS SAM) applications. To learn more about CloudWatch Application Insights, see Amazon CloudWatch Application Insights in the *Amazon CloudWatch User Guide*.

Topics

- Configuring CloudWatch Application Insights with AWS SAM
- Next steps

Configuring CloudWatch Application Insights with AWS SAM

Configure CloudWatch Application Insights for your AWS SAM applications through the AWS SAM Command Line Interface (AWS SAM CLI) or through your AWS SAM templates.

Configure through the AWS SAM CLI

When initializing your application with **sam init**, activate CloudWatch Application Insights through the interactive flow or by using the **--application-insights** option.

To activate CloudWatch Application Insights through the AWS SAM CLI interactive flow, enter **y** when prompted.

```
Would you like to enable monitoring using CloudWatch Application Insights?
For more info, please view https://docs.aws.amazon.com/AmazonCloudWatch/latest/
monitoring/cloudwatch-application-insights.html [y/N]:
```

To activate CloudWatch Application Insights with the **--application-insights** option, do the following.

```
sam init --application-insights
```

To learn more about using the **sam init** command, see sam init.

Configure through AWS SAM templates

Activate CloudWatch Application Insights by defining the `AWS::ResourceGroups::Group` and `AWS::ApplicationInsights::Application` resources in your AWS SAM templates.

```yaml
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  ApplicationResourceGroup:
    Type: AWS::ResourceGroups::Group
    Properties:
      Name:
        Fn::Join:
        - ''
        - - ApplicationInsights-SAM-
          - Ref: AWS::StackName
      ResourceQuery:
        Type: CLOUDFORMATION_STACK_1_0
  ApplicationInsightsMonitoring:
    Type: AWS::ApplicationInsights::Application
    Properties:
      ResourceGroupName:
        Fn::Join:
          - ''
          - - ApplicationInsights-SAM-
            - Ref: AWS::StackName
      AutoConfigurationEnabled: 'true'
    DependsOn: ApplicationResourceGroup
```

- `AWS::ResourceGroups::Group` – Creates a group to organize your AWS resources in order to manage and automate tasks on a large number of resources at one time. Here, you create a resource group to use with CloudWatch Application Insights. For more information on this resource type, see `AWS::ResourceGroups::Group` in the *AWS CloudFormation User Guide*.

- `AWS::ApplicationInsights::Application` – Configures CloudWatch Application Insights for the resource group. For more information on this resource type, see `AWS::ApplicationInsights::Application` in the *AWS CloudFormation User Guide*.

Both resources are automatically passed through to AWS CloudFormation at application deployment. You can use the AWS CloudFormation syntax in your AWS SAM template to configure CloudWatch Application Insights further. For more information, see Use AWS CloudFormation templates in the *Amazon CloudWatch User Guide*.

When using the **sam init --application-insights** command, both of these resources are automatically generated in your AWS SAM template. Here is an example of a generated template.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: >
  sam-app-test

  Sample SAM Template for sam-app-test

# More info about Globals: https://github.com/awslabs/serverless-application-model/
blob/master/docs/globals.rst
Globals:
  Function:
    Timeout: 3
    MemorySize: 128

Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function # More info about Function Resource:
 https://github.com/awslabs/serverless-application-model/blob/master/
versions/2016-10-31.md#awsserverlessfunction
    Properties:
      CodeUri: hello_world/
      Handler: app.lambda_handler
      Runtime: python3.9
      Architectures:
      - x86_64
```

```
      Events:
        HelloWorld:
          Type: Api # More info about API Event Source: https://github.com/awslabs/
serverless-application-model/blob/master/versions/2016-10-31.md#api
          Properties:
            Path: /hello
            Method: get

  ApplicationResourceGroup:
    Type: AWS::ResourceGroups::Group
    Properties:
      Name:
        Fn::Join:
        - ''
        - - ApplicationInsights-SAM-
          - Ref: AWS::StackName
      ResourceQuery:
        Type: CLOUDFORMATION_STACK_1_0
  ApplicationInsightsMonitoring:
    Type: AWS::ApplicationInsights::Application
    Properties:
      ResourceGroupName:
        Fn::Join:
        - ''
        - - ApplicationInsights-SAM-
          - Ref: AWS::StackName
      AutoConfigurationEnabled: 'true'
    DependsOn: ApplicationResourceGroup

Outputs:
  # ServerlessRestApi is an implicit API created out of Events key under
 Serverless::Function
  # Find out more about other implicit resources you can reference within SAM
  # https://github.com/awslabs/serverless-application-model/blob/master/docs/internals/
generated_resources.rst#api
  HelloWorldApi:
    Description: API Gateway endpoint URL for Prod stage for Hello World function
    Value: !Sub "https://${ServerlessRestApi}.execute-api.${AWS::Region}.amazonaws.com/
Prod/hello/"
  HelloWorldFunction:
    Description: Hello World Lambda Function ARN
    Value: !GetAtt HelloWorldFunction.Arn
  HelloWorldFunctionIamRole:
    Description: Implicit IAM Role created for Hello World function
```

```
      Value: !GetAtt HelloWorldFunctionRole.Arn
```

Next steps

After configuring CloudWatch Application Insights, use **sam build** to build your application and **sam deploy** to deploy your application. All CloudWatch Application Insights supported resources will be configured for monitoring.

- For a list of supported resources, see [Supported logs and metrics](#) in the *Amazon CloudWatch User Guide*.

- To learn how to access CloudWatch Application Insights, see [Access CloudWatch Application Insights](#) in the *Amazon CloudWatch User Guide*.

Working with logs

To simplify troubleshooting, the AWS SAM CLI has a command called `sam logs`. This command lets you fetch logs generated by your Lambda function from the command line.

> ⓘ **Note**
>
> The `sam logs` command works for all AWS Lambda functions, not just the ones you deploy using AWS SAM.

Fetching logs by AWS CloudFormation stack

When your function is a part of an AWS CloudFormation stack, you can fetch logs by using the function's logical ID:

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

Fetching logs by Lambda function name

Or, you can fetch logs by using the function's name:

```
sam logs -n mystack-HelloWorldFunction-1FJ8PD
```

Tailing logs

Add the `--tail` option to wait for new logs and see them as they arrive. This is helpful during deployment or when you're troubleshooting a production issue.

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

Viewing logs for a specific time range

You can view logs for a specific time range by using the `-s` and `-e` options:

```
sam logs -n HelloWorldFunction --stack-name mystack -s '10min ago' -e '2min ago'
```

Filtering logs

Use the `--filter` option to quickly find logs that match terms, phrases, or values in your log events:

```
sam logs -n HelloWorldFunction --stack-name mystack --filter "error"
```

In the output, the AWS SAM CLI underlines all occurrences of the word "error" so you can easily locate the filter keyword within the log output.

Error highlighting

When your Lambda function crashes or times out, the AWS SAM CLI highlights the timeout message in red. This helps you easily locate specific executions that are timing out within a giant stream of log output.

JSON pretty printing

If your log messages print JSON strings, the AWS SAM CLI automatically pretty prints the JSON to help you visually parse and understand the JSON.

Publishing serverless applications using the AWS SAM CLI

To make your AWS SAM application available for others to find and deploy, you can use the AWS SAM CLI to publish it to the AWS Serverless Application Repository. To publish your application using the AWS SAM CLI, you must define it using an AWS SAM template. You also must have tested it locally or in the AWS Cloud.

Follow the instructions in this topic to create a new application, create a new version of an existing application, or update the metadata of an existing application. (What you do depends on whether the application already exists in the AWS Serverless Application Repository, and whether any application metadata is changing.) For more information about application metadata, see AWS SAM template Metadata section properties.

Prerequisites

Before you publish an application to the AWS Serverless Application Repository using the AWS SAM CLI, you must have the following:

- The AWS SAM CLI installed. For more information, see Installing the AWS SAM CLI. To determine whether the AWS SAM CLI is installed, run the following command:

```
sam --version
```

- A valid AWS SAM template.

- Your application code and dependencies that the AWS SAM template references.

- A semantic version, required only to share your application publicly. This value can be as simple as 1.0.

- A URL that points to your application's source code.

- A README.md file. This file should describe how customers can use your application and how to configure it before deploying it in their own AWS accounts.

- A LICENSE.txt file, required only to share your application publicly.

- If your application contains any nested applications, you must have already published them to the AWS Serverless Application Repository.

- A valid Amazon Simple Storage Service (Amazon S3) bucket policy that grants the service read permissions for artifacts that you upload to Amazon S3 when you package your application. To set up this policy, do the following:

  1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.

  2. Choose the name of the Amazon S3 bucket that you used to package your application.

  3. Choose **Permissions**.

  4. On the **Permissions** tab, under **Bucket policy**, choose **Edit**.

  5. On the **Edit bucket policy** page, paste the following policy statement into the **Policy** editor. In the policy statement, make sure to use your bucket name in the `Resource` element and your AWS account ID in the `Condition` element. The expression in the `Condition` element ensures that AWS Serverless Application Repository has permission to access only applications from the specified AWS account. For more information about policy statements, see IAM JSON policy elements reference in the *IAM User Guide*.

     ```
     {
         "Version": "2012-10-17",
         "Statement": [
             {
                 "Effect": "Allow",
                 "Principal": {
                     "Service":  "serverlessrepo.amazonaws.com"
                 },
                 "Action": "s3:GetObject",
                 "Resource": "arn:aws:s3:::<your-bucket-name>/*",
                 "Condition" : {
                     "StringEquals": {
                         "aws:SourceAccount": "123456789012"
                     }
                 }
             }
         ]
     }
     ```

  6. Choose **Save changes**.

Publishing a new application

Step 1: Add a `Metadata` section to the AWS SAM template

First, add a `Metadata` section to your AWS SAM template. Provide the application information to be published to the AWS Serverless Application Repository.

The following is an example `Metadata` section:

```
Metadata:
  AWS::ServerlessRepo::Application:
    Name: my-app
    Description: hello world
    Author: user1
    SpdxLicenseId: Apache-2.0
    LicenseUrl: LICENSE.txt
    ReadmeUrl: README.md
    Labels: ['tests']
    HomePageUrl: https://github.com/user1/my-app-project
    SemanticVersion: 0.0.1
    SourceCodeUrl: https://github.com/user1/my-app-project

Resources:
  HelloWorldFunction:
    Type: AWS::Lambda::Function
      Properties:
        ...
        CodeUri: source-code1
        ...
```

For more information about the `Metadata` section of the AWS SAM template, see AWS SAM template Metadata section properties.

Step 2: Package the application

Run the following AWS SAM CLI command, which uploads the application's artifacts to Amazon S3 and outputs a new template file called `packaged.yaml`:

```
sam package --output-template-file packaged.yaml --s3-bucket <your-bucket-name>
```

You use the `packaged.yaml` template file in the next step to publish the application to the AWS Serverless Application Repository. This file is similar to the original template file (`template.yaml`), but it has a key difference—the `CodeUri`, `LicenseUrl`, and `ReadmeUrl` properties point to the Amazon S3 bucket and objects that contain the respective artifacts.

The following snippet from an example `packaged.yaml` template file shows the `CodeUri` property:

```
MySampleFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: s3://bucketname/fbd77a3647a4f47a352fcObjectGUID


...
```

Step 3: Publish the application

To publish a private version of your AWS SAM application to the AWS Serverless Application Repository, run the following AWS SAM CLI command:

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

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

Step 4: Share the application (optional)

By default, your application is set to private, so it isn't visible to other AWS accounts. To share your application with others, you must either make it public or grant permission to a specific list of AWS accounts.

For information about sharing your application using the AWS CLI, see AWS Serverless Application Repository Resource-Based Policy Examples in the *AWS Serverless Application Repository Developer Guide*. For information on sharing your application using the AWS Management Console, see Sharing an Application in the *AWS Serverless Application Repository Developer Guide*.

Publishing a new version of an existing application

After you've published an application to the AWS Serverless Application Repository, you might want to publish a new version of it. For example, you might have changed your Lambda function code or added a new component to your application architecture.

To update an application that you've previously published, publish the application again using the same process detailed previously. In the `Metadata` section of the AWS SAM template file, provide the same application name that you originally published it with, but include a new `SemanticVersion` value.

For example, consider an application published with the name `SampleApp` and a `SemanticVersion` of `1.0.0`. To update that application, the AWS SAM template must have the application name `SampleApp` and a `SemanticVersion` of `1.0.1` (or anything other than `1.0.0`).

Additional topics

- [AWS SAM template Metadata section properties](#)

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.

Property	Type	Required	Description
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: "^[a-z0-9](([a-z0-9]|-(?!-))*[a-z0-9])?$";
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

Property	Type	Required	Description
			an application, you must either delete it first, or publish a new application with a different name.
ReadmeUrl	String	FALSE	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 `sam package` command can have a reference to a local file for this property. However, to be published using the `sam publish` command, this property must be a reference to an Amazon S3 bucket. Maximum size: 5 MB.
Labels	String	FALSE	The labels that improve discovery of applications in search results. Minimum length=1. Maximum length=127. Maximum number of labels: 10. Pattern: "^[a-zA-Z0-9+\\-_:\\/@]+$";
HomePageUrl	String	FALSE	A URL with more information about the application—for example, the location of your GitHub repository for the application.
SemanticV ersion	String	FALSE	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.
SourceCodeUrl	String	FALSE	A link to a public repository for the source code of your application.

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   717

Working with third-party services

This section provides documentation on using the AWS Serverless Application Model (AWS SAM) with third-party services.

Topics

- AWS SAM CLI Terraform support

AWS SAM CLI Terraform support

This section covers using the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) with your Terraform projects and Terraform Cloud.

To provide feedback and submit feature requests, create a GitHub Issue.

Topics

- What is AWS SAM CLI support for Terraform?
- Getting started with Terraform support for AWS SAM CLI
- Using the AWS SAM CLI with Terraform for local debugging and testing
- Using the AWS SAM CLI with Serverless.tf for local debugging and testing
- AWS SAM CLI with Terraform reference

What is AWS SAM CLI support for Terraform?

Use the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) with your Terraform projects or Terraform Cloud to perform local debugging and testing of:

- AWS Lambda functions and layers.
- Amazon API Gateway APIs.

For an introduction to Terraform, see What is Terraform? at the *HashiCorp Terraform website*.

To provide feedback and submit feature requests, create a GitHub Issue.

Topics

- [What is the AWS SAM CLI?](#)

- [How do I use the AWS SAM CLI with Terraform?](#)

- [Next steps](#)

What is the AWS SAM CLI?

The AWS SAM CLI is a command line tool that you can use with AWS SAM templates and supported third-party integrations, such as Terraform, to build and run your serverless applications. For an introduction to the AWS SAM CLI, see [What is the AWS SAM CLI?](#).

The AWS SAM CLI supports the following commands for Terraform:

- `sam local invoke` – Initiate a one-time invocation of an AWS Lambda function resource locally. To learn more about this command, see [Using sam local invoke](#).

- `sam local start-api` – Run your Lambda resources locally and test through a local HTTP server host. This type of testing is helpful for Lambda functions that are invoked by an API Gateway endpoint. To learn more about this command, see [Using sam local start-api](#).

- `sam local start-lambda` – Start a local endpoint for your Lambda function in order to invoke your function locally using AWS Command Line Interface (AWS CLI) or SDKs. To learn more about this command, see [Using sam local start-lambda](#).

How do I use the AWS SAM CLI with Terraform?

The [core Terraform workflow](#) consists of three stages: **Write**, **Plan**, and **Apply**. With AWS SAM CLI support for Terraform, you can take advantage of the AWS SAM CLI `sam local` set of commands while continuing to use your Terraform workflows to manage your applications on AWS. Generally, this means the following:

- **Write** – Author your infrastructure as code using Terraform.

- **Test and debug** – Use the AWS SAM CLI to locally test and debug your applications.

- **Plan** – Preview changes before applying.

- **Apply** – Provision your infrastructure.

For an example of using the AWS SAM CLI with Terraform, see [Better together: AWS SAM CLI and HashiCorp Terraform](#) at the *AWS Compute Blog*.

Next steps

To complete all prerequisites and set up Terraform, see Getting started with Terraform support for AWS SAM CLI.

Getting started with Terraform support for AWS SAM CLI

This topic covers how to get started with using the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) with Terraform.

To provide feedback and submit feature requests, create a GitHub Issue.

Topics

- AWS SAM CLI Terraform prerequisites
- Using AWS SAM CLI commands with Terraform
- Set up for Terraform projects
- Set up for Terraform Cloud

AWS SAM CLI Terraform prerequisites

Complete all prerequisites to begin using the AWS SAM CLI with your Terraform projects.

1. **Install or upgrade the AWS SAM CLI**

   To check if you have the AWS SAM CLI installed, run the following:

   ```
   $ sam --version
   ```

   If the AWS SAM CLI is already installed, the output will display a version. To upgrade to the newest version, see Upgrading the AWS SAM CLI.

   For instructions on installing the AWS SAM CLI along with all of its prerequisites, see Installing the AWS SAM CLI.

2. **Install Terraform**

   To check if you have Terraform installed, run the following:

   ```
   $ terraform -version
   ```

> To install Terraform, see [Install Terraform](#) in the *Terraform registry*.

3. **Install Docker for local testing**

   The AWS SAM CLI requires Docker for local testing. To install Docker, see [Installing Docker to use with the AWS SAM CLI](#).

Using AWS SAM CLI commands with Terraform

When you run a supported AWS SAM CLI command, use the `--hook-name` option and provide the `terraform` value. The following is an example:

```
$ sam local invoke --hook-name terraform
```

You can configure this option in your AWS SAM CLI configuration file with the following:

```
hook_name = "terraform"
```

Set up for Terraform projects

Complete steps in this topic to use the AWS SAM CLI with Terraform projects.

No additional setup is required if you build your AWS Lambda artifacts outside of your Terraform project. See [Using the AWS SAM CLI with Terraform for local debugging and testing](#) to start using the AWS SAM CLI.

If you build your Lambda artifacts within your Terraform projects, you must do the following:

1. Install Python 3.8 or newer
2. Install the Make tool.
3. Define your Lambda artifacts build logic within your Terraform project.
4. Define a `sam metadata` resource to inform the AWS SAM CLI of your build logic.
5. Use the AWS SAM CLI `sam build` command to build your Lambda artifacts.

Install Python 3.8 or newer

Python 3.8 or newer is required for use with the AWS SAM CLI. When you run `sam build`, the AWS SAM CLI creates `makefiles` that contain Python commands to build your Lambda artifacts.

For installation instructions, see [Downloading Python](#) in Python's *Beginners Guide*.

Verify that Python 3.8 or newer is added to your machine path by running:

```
$ python --version
```

The output should display a version of Python that is 3.8 or newer.

Install the Make tool

GNU [Make](#) is a tool that controls the generation of executables and other non-source files for your project. The AWS SAM CLI creates `makefiles` which rely on this tool to build your Lambda artifacts.

If you do not have Make installed on your local machine, install it before moving forward.

For Windows, you can install using [Chocolatey](#). For instructions, see [Using Chocolatey](#) in *How to Install and Use "Make" in Windows*

Define the Lambda artifacts build logic

Use the `null_resource` Terraform resource type to define your Lambda build logic. The following is an example that uses a custom build script to build a Lambda function.

```
resource "null_resource" "build_lambda_function" {
    triggers = {
        build_number = "${timestamp()}"
    }

    provisioner "local-exec" {
        command =  substr(pathexpand("~"), 0, 1) == "/"? "./
py_build.sh \"${local.lambda_src_path}\" \"${local.building_path}\"
 \"${local.lambda_code_filename}\" Function" : "powershell.exe -File .\\PyBuild.ps1
 ${local.lambda_src_path} ${local.building_path} ${local.lambda_code_filename}
 Function"
    }
}
```

Define a sam metadata resource

The `sam metadata` resource is a `null_resource` Terraform resource type that provides the AWS SAM CLI with the information it needs to locate your Lambda artifacts. A unique `sam metadata`

resource is required for each Lambda function or layer in your project. To learn more about this resource type, see [null_resource](#) in the *Terraform registry*.

To define a sam metadata resource

1. Name your resource starting with `sam_metadata_` to identify the resource as a sam metadata resource.

2. Define your Lambda artifact properties within the `triggers` block of your resource.

3. Specify your `null_resource` that contains your Lambda build logic with the `depends_on` argument.

   The following is an example template:

   ```
   resource "null_resource" "sam_metadata_..." {
     triggers = {
       resource_name = resource_name
       resource_type = resource_type
       original_source_code = original_source_code
       built_output_path = built_output_path
     }
     depends_on = [
       null_resource.build_lambda_function # ref to your build logic
     ]
   }
   ```

   The following is an example `sam metadata` resource:

   ```
   resource "null_resource" "sam_metadata_aws_lambda_function_publish_book_review" {
       triggers = {
           resource_name = "aws_lambda_function.publish_book_review"
           resource_type = "ZIP_LAMBDA_FUNCTION"
           original_source_code = "${local.lambda_src_path}"
           built_output_path = "${local.building_path}/${local.lambda_code_filename}"
       }
       depends_on = [
           null_resource.build_lambda_function
       ]
   }
   ```

The contents of your `sam metadata` resource will vary based on the Lambda resource type (function or layer), and the packaging type (ZIP or image). For more information, along with examples, see [sam metadata resource](#).

When you configure a `sam metadata` resource and use a supported AWS SAM CLI command, the AWS SAM CLI will generate the metadata file before running the AWS SAM CLI command. Once you have generated this file, you can use the `--skip-prepare-infra` option with future AWS SAM CLI commands to skip the metadata generation process and save time. This option should only be used if you haven't made any infrastructure changes, such as creating new Lambda functions or new API endpoints.

Use the AWS SAM CLI to build your Lambda artifacts

Use the AWS SAM CLI `sam build` command to build your Lambda artifacts. When you run `sam build`, the AWS SAM CLI does the following:

1. Looks for `sam metadata` resources in your Terraform project to learn about and locate your Lambda resources.
2. Initiates your Lambda build logic to build your Lambda artifacts.
3. Creates an `.aws-sam` directory that organizes your Terraform project for use with the AWS SAM CLI `sam local` commands.

To build with sam build

1. From the directory containing your Terraform root module, run the following:

   ```
   $ sam build --hook-name terraform
   ```

2. To build a specific Lambda function or layer, run the following

   ```
   $ sam build --hook-name terraform lambda-resource-id
   ```

   The Lambda resource ID can be the Lambda function name or full Terraform resource address, such as `aws_lambda_function.list_books` or `module.list_book_function.aws_lambda_function.this[0]`.

If your function source code or other Terraform configuration files are located outside of the directory containing your Terraform root module, you need to specify the location. Use the `--`

`terraform-project-root-path` option to specify the absolute or relative path to the top-level directory containing these files. The following is an example:

```
$ sam build --hook-name terraform --terraform-project-root-path ~/projects/terraform/
demo
```

Build using a container

When running the AWS SAM CLI `sam build` command, you can configure the AWS SAM CLI to build your application using a local Docker container.

> **ⓘ Note**
>
> You must have Docker installed and configured. For instructions, see Installing Docker to use with the AWS SAM CLI.

To build using a container

1. Create a `Dockerfile` that contains the Terraform, Python, and Make tools. You should also include your Lambda function runtime.

   The following is an example `Dockerfile`:

   ```
   FROM public.ecr.aws/amazonlinux/amazonlinux:2

   RUN yum -y update \
       && yum install -y unzip tar gzip bzip2-devel ed gcc gcc-c++ gcc-gfortran \
       less libcurl-devel openssl openssl-devel readline-devel xz-devel \
       zlib-devel glibc-static libcxx libcxx-devel llvm-toolset-7 zlib-static \
       && rm -rf /var/cache/yum

   RUN yum -y install make \
       && yum -y install zip

   RUN yum install -y yum-utils \
       && yum-config-manager --add-repo https://rpm.releases.hashicorp.com/
   AmazonLinux/hashicorp.repo \
       && yum -y install terraform \
       && terraform --version
   ```

```
# AWS Lambda Builders
RUN amazon-linux-extras enable python3.8
RUN yum clean metadata && yum -y install python3.8
RUN curl -L get-pip.io | python3.8
RUN pip3 install aws-lambda-builders
RUN ln -s /usr/bin/python3.8 /usr/bin/python3
RUN python3 --version

VOLUME /project
WORKDIR /project

ENTRYPOINT ["sh"]
```

2. Use [docker build](#) to build your Docker image.

   The following is an example:

   ```
   $ docker build --tag terraform-build:v1 <path-to-directory-containing-Dockerfile>
   ```

3. Run the AWS SAM CLI `sam build` command with the `--use-container` and `--build-image` options.

   The following is an example:

   ```
   $ sam build --use-container --build-image terraform-build:v1
   ```

Next steps

To start using the AWS SAM CLI with your Terraform projects, see [Using the AWS SAM CLI with Terraform for local debugging and testing](#).

Set up for Terraform Cloud

We recommend that you use Terraform v1.6.0 or newer. If you are using an older version, you must generate a Terraform plan file locally. The local plan file provides the AWS SAM CLI with the information it needs to perform local testing and debugging.

To generate a local plan file

> ⓘ **Note**
>
> These steps are not required for Terraform v1.6.0 or newer. To start using the AWS SAM CLI with Terraform Cloud, see Using AWS SAM CLI with Terraform.

1. **Configure an API token** – The type of token will depend on your access level. To learn more, see API Tokens in the *Terraform Cloud documentation*.

2. **Set your API token environment variable** – The following is an example from the command line:

   ```
   $ export TOKEN="<api-token-value>"
   ```

3. **Obtain your run ID** – From the Terraform Cloud console, locate the run ID for the Terraform run that you'd like to use with the AWS SAM CLI.

   The run ID is located in the breadcrumb path of your run.
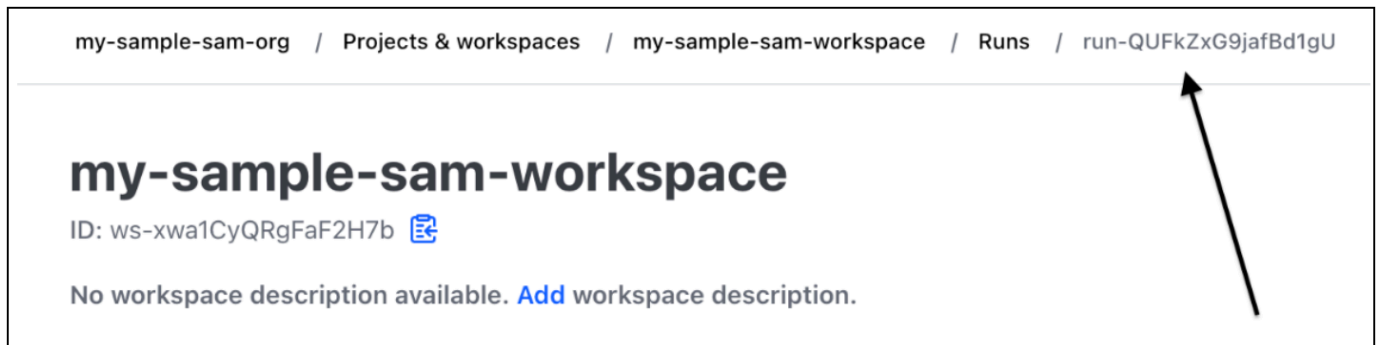
4. **Fetch the plan file** – Using your API token, obtain your local plan file. The following is an example from the command line:

   ```
   curl \
       --header "Authorization: Bearer $TOKEN" \
       --header "Content-Type: application/vnd.api+json" \
       --location \
       https://app.terraform.io/api/v2/runs/<run ID>/plan/json-output \
       > custom_plan.json
   ```

You are now ready to use the AWS SAM CLI with Terraform Cloud. When using a supported AWS SAM CLI command, use the `--terraform-plan-file` option to specify the name and path of your local plan file. The following is an example:

```
$ sam local invoke --hook-name terraform --terraform-plan-file custom-plan.json
```

The following is an example, using the `sam local start-api` command:

```
$ sam local start-api --hook-name terraform --terraform-plan-file custom-plan.json
```

For a sample application that you can use with these examples, see api_gateway_v2_tf_cloud in the *aws-samples GitHub repository*.

Next steps

To start using the AWS SAM CLI with Terraform Cloud, see Using the AWS SAM CLI with Terraform for local debugging and testing.

Using the AWS SAM CLI with Terraform for local debugging and testing

This topic covers how to use supported AWS Serverless Application Model Command Line Interface (AWS SAM CLI) commands with your Terraform projects and Terraform Cloud.

To provide feedback and submit feature requests, create a GitHub Issue.

Topics

- Local testing with sam local invoke
- Local testing with sam local start-api
- Local testing with sam local start-lambda
- Terraform limitations

Local testing with sam local invoke

> **ⓘ Note**
>
> To use the AWS SAM CLI to test locally, you must have Docker intalled and configured. For instructions, see Installing Docker to use with the AWS SAM CLI.

The following is an example of testing your Lambda function locally by passing in an event:

```
$ sam local invoke --hook-name terraform hello_world_function -e events/event.json -
```

To learn more about using this command, see Using sam local invoke.

Local testing with sam local start-api

To use sam local start-api with Terraform, run the following:

```
$ sam local start-api --hook-name terraform
```

The following is an example:

```
$ sam local start-api --hook-name terraform




Running Prepare Hook to prepare the current application


Executing prepare hook of hook "terraform"


Initializing Terraform application


...
Creating terraform plan and getting JSON output


....
Generating metadata file

```

```
Unresolvable attributes discovered in project, run terraform apply to resolve them.




Finished generating metadata file. Storing in...
Prepare hook completed and metadata file generated at: ...
Mounting HelloWorldFunction at http://127.0.0.1:3000/hello [GET]



Mounting None at http://127.0.0.1:3000/hello [POST]



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. If you
used sam build before running local commands, you will need to re-run sam build for the
 changes to be picked up. You only need to restart SAM CLI if you update your AWS SAM
 template
2023-06-26 13:21:20  * Running on http://127.0.0.1:3000/ (Press CTRL+C to quit)
```

To learn more about this command, see [Using sam local start-api](#).

Lambda functions that use Lambda authorizers

For Lambda functions configured to use Lambda authorizers, the AWS SAM CLI will automatically invoke your Lambda authorizer before invoking your Lambda function endpoint.

- To learn more about this feature in the AWS SAM CLI, see [Lambda functions that use Lambda authorizers](#).
- For more information on using Lambda authorizers in Terraform, see [Resource: aws_api_gateway_authorizer](#) in the *Terraform registry*.

Local testing with sam local start-lambda

The following is an example of testing your Lambda function locally with the AWS Command Line Interface (AWS CLI):

1. Use the AWS SAM CLI to create a local testing environment:

```
$ sam local start-lambda --hook-name terraform hello_world_function
```

2. Use the AWS CLI to invoke your function locally:

```
$ aws lambda invoke --function-name hello_world_function --endpoint-
url http://127.0.0.1:3001/ response.json --cli-binary-format raw-in-base64-out --
payload file://events/event.json
```

To learn more about this command, see [Using sam local start-lambda](#).

Terraform limitations

The following are limitations when using the AWS SAM CLI with Terraform:

- Lambda functions linked to multiple layers.

- Terraform local variables that define links between resources.

- Referencing a Lambda function that hasn't been created yet. This includes functions that are defined in the body attribute of the REST API resource.

To avoid these limitations, you can run `terraform apply` when a new resource is added.

Using the AWS SAM CLI with Serverless.tf for local debugging and testing

The AWS Serverless Application Model Command Line Interface (AWS SAM CLI) can be used with Serverless.tf modules for local debugging and testing of your AWS Lambda functions and layers. The following AWS SAM CLI commands are supported:

- `sam build`

- `sam local invoke`

- `sam local start-api`

- `sam local start-lambda`

> ⓘ **Note**
>
> Serverless.tf version 4.6.0 and newer supports AWS SAM CLI integration.

To begin using the AWS SAM CLI with your Serverless.tf modules, update to the latest version Serverless.tf and the AWS SAM CLI.

Starting from **serverless.tf version 6.0.0**, you must set the `create_sam_metadata` parameter as `true`. This generates the metadata resources that the AWS SAM CLI `sam build` command requires.

To learn more about Serverless.tf, see the [terraform-aws-lambda-module](#).

AWS SAM CLI with Terraform reference

This section is the reference for using the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) with Terraform for local debugging and testing.

To provide feedback and submit feature requests, create a [GitHub Issue](#).

AWS SAM supported features reference

Reference documentation for AWS SAM CLI features that are supported for use with Terraform can be found here:

- [sam build](#)
- [sam local invoke](#)
- [sam local start-api](#)
- [sam local start-lambda](#)

Terraform specific reference

Reference documentation specific to using AWS SAM CLI with Terraform can be found here:

- [sam metadata resource](#)

sam metadata resource

This page contains reference information for the **sam metadata resource** resource type used with Terraform projects.

- For an introduction to using the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) with Terraform, see [What is AWS SAM CLI support for Terraform?](#).

- To use the AWS SAM CLI with Terraform, see [Using the AWS SAM CLI with Terraform for local debugging and testing](#).

Topics

- [Arguments](#)
- [Examples](#)

Arguments

Argument	Description
`built_output_path`	The path to your AWS Lambda function's built artifacts.
`docker_build_args`	Decoded string of the Docker build arguments JSON object. This argument is optional.
`docker_context`	The path to the directory containing the Docker image build context.
`docker_file`	The path to the Docker file. This path is relative to the `docker_context` path. This argument is optional. Default value is `Dockerfile`.
`docker_tag`	The value of the created Docker image tag. This value is optional.
`depends_on`	The path to the building resource for your Lambda function or layer. To learn more, see [The **depends_on** argument](#) in the *Terraform registry*.

Argument	Description
original_source_code	The path to where your Lambda function is defined. This value can be a string, array of strings, or a decoded JSON object as a string. • For string arrays, only the first value is used since multiple code paths are not supported. • For JSON objects, the `source_code_property` must also be defined.
resource_name	The Lambda function name.
resource_type	The format of your Lambda function package type. Accepted values are: • `IMAGE_LAMBDA_FUNCTION` • `LAMBDA_LAYER` • `ZIP_LAMBDA_FUNCTION`
source_code_property	The path to the Lambda resource code in the JSON object. Define this property when `original_source_code` is a JSON object.

Examples

sam metadata resource referencing a Lambda function using the ZIP package type

```
# Lambda function resource
resource "aws_lambda_function" "tf_lambda_func" {
  filename = "${path.module}/python/hello-world.zip"
  handler = "index.lambda_handler"
  runtime = "python3.8"
  function_name = "function_example"
  role = aws_iam_role.iam_for_lambda.arn
  depends_on = [
    null_resource.build_lambda_function # function build logic
  ]
}

# sam metadata resource
resource "null_resource" "sam_metadata_function_example" {
```

```
  triggers = {
    resource_name = "aws_lambda_function.function_example"
    resource_type = "ZIP_LAMBDA_FUNCTION"
    original_source_code = "${path.module}/python"
    built_output_path = "${path.module}/building/function_example"
  }
  depends_on = [
    null_resource.build_lambda_function # function build logic
  ]
}
```

sam metadata resource referencing a Lambda function using the image package type

```
resource "null_resource" "sam_metadata_function {
  triggers = {
    resource_name = "aws_lambda_function.image_function"
    resource_type = "IMAGE_LAMBDA_FUNCTION"
    docker_context = local.lambda_src_path
    docker_file = "Dockerfile"
    docker_build_args = jsonencode(var.build_args)
    docker_tag = "latest"
  }
}
```

sam metadata resource referencing a Lambda layer

```
resource "null_resource" "sam_metadata_layer1" {
  triggers = {
    resource_name = "aws_lambda_layer_version.layer"
    resource_type = "LAMBDA_LAYER"
    original_source_code = local.layer_src
    built_output_path = "${path.module}/${layer_build_path}"
  }
  depends_on = [null_resource.layer_build]
}
```

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](#)
- [Process Amazon S3 events](#)

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](#).

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
    ```

Note that `gh:` in the command above gets expanded to the GitHub url `https://` `github.com/`.

2. Review the contents of the directory that the command created (`dynamodb_event_reader/`):

   - `template.yaml` – Defines two AWS resources that the Read DynamoDB application needs: a Lambda function and a DynamoDB table. The template also defines mapping between the two resources.

   - `read_dynamodb_event/` directory – Contains the DynamoDB application code.

Step 2: Test the application locally

For local testing, use the AWS SAM CLI to generate a sample DynamoDB event and invoke the Lambda function:

```
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.

   ```
   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 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.

```
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.

Step 1: Initialize the application

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

To initialize the application

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

    ```
    sam init \
    --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.

    - `SampleEvent.json` – The sample event source, which is used for local testing.

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 Cloud Development Kit (AWS CDK)

You can use the AWS SAM CLI to locally test and build serverless applications defined using the AWS Cloud Development Kit (AWS CDK). Because the AWS SAM CLI works within the AWS CDK project structure, you can still use the AWS CDK Toolkit for creating, modifying, and deploying your AWS CDK applications.

For information about installing and configuring the AWS CDK, see Getting started with the AWS CDK in the *AWS Cloud Development Kit (AWS CDK) Developer Guide*.

> ℹ️ **Note**
>
> The AWS SAM CLI supports AWS CDK v1 starting from version 1.135.0 and AWS CDK v2 starting from version 2.0.0.

Topics

- Getting started with AWS SAM and the AWS CDK
- Locally testing AWS CDK applications
- Building AWS CDK applications
- Deploying AWS CDK applications

Getting started with AWS SAM and the AWS CDK

This topic describes what you need to use the AWS SAM CLI with AWS CDK applications, and provides instructions for building and locally testing a simple AWS CDK application.

Prerequisites

To use the AWS SAM CLI with AWS CDK, you must install the AWS CDK, and the AWS SAM CLI.

- For information about installing the AWS CDK, see Getting started with the AWS CDK in the *AWS Cloud Development Kit (AWS CDK) Developer Guide*.
- For information about installing the AWS SAM CLI, see Installing the AWS SAM CLI.

Creating and locally testing an AWS CDK application

To locally test an AWS CDK application using the AWS SAM CLI, you must have an AWS CDK application that contains a Lambda function. Use the following steps to create a basic AWS CDK application with a Lambda function. For more information, see Creating a serverless application using the AWS CDK in the *AWS Cloud Development Kit (AWS CDK) Developer Guide*.

> ⓘ **Note**
>
> The AWS SAM CLI supports AWS CDK v1 starting from version 1.135.0 and AWS CDK v2 starting from version 2.0.0.

Step 1: Create an AWS CDK application

For this tutorial, initialize an AWS CDK application that uses TypeScript.

Command to run:

AWS CDK v2

```
mkdir cdk-sam-example
cd cdk-sam-example
cdk init app --language typescript
```

AWS CDK v1

```
mkdir cdk-sam-example
cd cdk-sam-example
cdk init app --language typescript
npm install @aws-cdk/aws-lambda
```

Step 2: Add a Lambda function to your application

Replace the code in `lib/cdk-sam-example-stack.ts` with the following:

AWS CDK v2

```
import { Stack, StackProps } from 'aws-cdk-lib';
```

```
import { Construct } from 'constructs';
import * as lambda from 'aws-cdk-lib/aws-lambda';

export class CdkSamExampleStack extends Stack {
  constructor(scope: Construct, id: string, props?: StackProps) {
    super(scope, id, props);

    new lambda.Function(this, 'MyFunction', {
      runtime: lambda.Runtime.PYTHON_3_9,
      handler: 'app.lambda_handler',
      code: lambda.Code.fromAsset('./my_function'),
    });
  }
}
```

AWS CDK v1

```
import * as cdk from '@aws-cdk/core';
import * as lambda from '@aws-cdk/aws-lambda';

export class CdkSamExampleStack extends cdk.Stack {
  constructor(scope: Construct, id: string, props?: StackProps) {
    super(scope, id, props);

    new lambda.Function(this, 'MyFunction', {
      runtime: lambda.Runtime.PYTHON_3_9,
      handler: 'app.lambda_handler',
      code: lambda.Code.fromAsset('./my_function'),
    });
  }
}
```

Step 3: Add your Lambda function code

Create a directory named my_function. In that directory, create a file named app.py.

Command to run:

```
mkdir my_function
cd my_function
touch app.py
```

Add the following code to `app.py`:

```python
def lambda_handler(event, context):
    return "Hello from SAM and the CDK!"
```

Step 4: Test your Lambda function

You can use the AWS SAM CLI to locally invoke a Lambda function that you define in an AWS CDK application. To do this, you need the function construct identifier and the path to your synthesized AWS CloudFormation template.

Command to run:

```
cdk synth --no-staging
```

```
sam local invoke MyFunction --no-event -t ./cdk.out/CdkSamExampleStack.template.json
```

Example output:

```
Invoking app.lambda_handler (python3.9)

START RequestId: 5434c093-7182-4012-9b06-635011cac4f2 Version: $LATEST
"Hello from SAM and the CDK!"
END RequestId: 5434c093-7182-4012-9b06-635011cac4f2
REPORT RequestId: 5434c093-7182-4012-9b06-635011cac4f2 Init Duration: 0.32 ms Duration:
 177.47 ms Billed Duration: 178 ms Memory Size: 128 MB Max Memory Used: 128 MB
```

For more information about options available to test AWS CDK applications using the AWS SAM CLI, see Locally testing AWS CDK applications.

Locally testing AWS CDK applications

You can use the AWS SAM CLI to locally test your AWS CDK applications by running the following commands from the project root directory of your AWS CDK application:

- sam local invoke
- sam local start-api

- [sam local start-lambda](#)

Before you run any of the **sam local** commands with a AWS CDK application, you must run cdk synth.

When running **sam local invoke** you need the function construct identifier that you want to invoke, and the path to your synthesized AWS CloudFormation template. If your application uses nested stacks, to resolve naming conflicts, you also need the stack name where the function is defined.

Usage:

```
# Invoke the function FUNCTION_IDENTIFIER declared in the stack STACK_NAME
sam local invoke [OPTIONS] [STACK_NAME/FUNCTION_IDENTIFIER]

# Start all APIs declared in the AWS CDK application
sam local start-api -t ./cdk.out/CdkSamExampleStack.template.json [OPTIONS]

# Start a local endpoint that emulates AWS Lambda
sam local start-lambda -t ./cdk.out/CdkSamExampleStack.template.json [OPTIONS]
```

Example

Consider stacks and functions that are declared with the following sample:

```
app = new HelloCdkStack(app, "HelloCdkStack",
    ...
)
class HelloCdkStack extends cdk.Stack {
  constructor(scope: Construct, id: string, props?: cdk.StackProps) {
    ...
    new lambda.Function(this, 'MyFunction', {
        ...
    });

    new HelloCdkNestedStack(this, 'HelloNestedStack' ,{
        ...
    });
}

class HelloCdkNestedStack extends cdk.NestedStack {
  constructor(scope: Construct, id: string, props?: cdk.NestedStackProps) {
```

Example   747

```
    ...
    new lambda.Function(this, 'MyFunction', {
        ...
    });
    new lambda.Function(this, 'MyNestedFunction', {
        ...
    });
}
```

The following commands locally invokes the Lambda functions defined in example presented above:

```
# Invoke MyFunction from the HelloCdkStack
sam local invoke -t ./cdk.out/HelloCdkStack.template.json MyFunction
```

```
# Invoke MyNestedFunction from the HelloCdkNestedStack
sam local invoke -t ./cdk.out/HelloCdkStack.template.json MyNestedFunction
```

```
# Invoke MyFunction from the HelloCdkNestedStack
sam local invoke -t ./cdk.out/HelloCdkStack.template.json HelloNestedStack/MyFunction
```

Building AWS CDK applications

The AWS SAM CLI provides support for building Lambda functions and layers defined in your AWS CDK application with sam build.

For Lambda functions that use zip artifacts, run cdk synth before you run sam local commands. sam build isn't required.

If your AWS CDK application uses functions with the image type, run cdk synth and then run sam build before you run sam local commands. When you run sam build, AWS SAM doesn't build Lambda functions or layers that use runtime-specific constructs, for example, NodejsFunction. sam build doesn't support bundled assets.

Example

Running the following command from the AWS CDK project root directory builds the application.

```
sam build -t ./cdk.out/CdkSamExampleStack.template.json
```

Deploying AWS CDK applications

The AWS SAM CLI doesn't support deploying AWS CDK applications. Use cdk deploy to deploy your application. For more information, see AWS CDK Toolkit (cdk command) in the *AWS Cloud Development Kit (AWS CDK) Developer Guide*

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](#).

AWS SAM templates are an extension of AWS CloudFormation templates. For the full reference for AWS CloudFormation templates, see [Template reference](#) in the *AWS CloudFormation User Guide*.

AWS SAM CLI command reference

The AWS Serverless Application Model Command Line Interface (AWS SAM CLI) is a command line tool that you can use with AWS SAM templates and supported third-party integrations to build and run your serverless applications.

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 that your Lambda function generates. 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*. You need the deployment package to upload your application to the AWS Cloud.
- `sam deploy` – Use this command to deploy your serverless application to the AWS Cloud. It creates the AWS resources and sets permissions and other configurations that are defined in the AWS SAM template.

For instructions about installing the AWS SAM CLI, see [Installing the AWS SAM CLI](#).

AWS SAM policy templates

With AWS SAM, you can choose from a list of policy templates to scope your AWS Lambda function's permissions to the resources that your application uses.

Topics

- [AWS Serverless Application Model (AWS SAM) specification](#)
- [AWS SAM CLI command reference](#)
- [AWS SAM CLI configuration file](#)
- [AWS SAM connector reference](#)
- [AWS SAM policy templates](#)
- [Image repositories](#)
- [Telemetry in the AWS SAM CLI](#)
- [Managing resource access and permissions](#)

AWS SAM CLI reference

This section is the reference for the AWS Serverless Application Model Command Line Interface (AWS SAM CLI).

Topics

- [AWS SAM CLI command reference](#)
- [AWS SAM CLI configuration file](#)
- [Managing AWS SAM CLI versions](#)
- [Verify the integrity of the AWS SAM CLI installer](#)
- [Setting up AWS credentials](#)
- [AWS SAM CLI troubleshooting](#)

AWS SAM CLI command reference

This section is the reference for the AWS SAM CLI commands. For instructions about installing the AWS SAM CLI, see [Installing the AWS SAM CLI](#).

Topics

- [sam build](#)

- [sam delete](#)

- [sam deploy](#)

- [sam init](#)

- [sam list](#)

- [sam local generate-event](#)

- [sam local invoke](#)

- [sam local start-api](#)

- [sam local start-lambda](#)

- [sam logs](#)

- [sam package](#)

- [sam pipeline bootstrap](#)

- [sam pipeline init](#)

- [sam publish](#)

- [sam remote invoke](#)

- [sam remote test-event](#)

- [sam sync](#)

- [sam traces](#)

- [sam validate](#)

sam build

This page provides reference information for the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam build` command.

- For an introduction to the AWS SAM CLI, see [What is the AWS SAM CLI?](#).

- For documentation on using the AWS SAM CLI `sam build` command, see [Using sam build](#).

The `sam build` command prepares an application for subsequent steps in the developer workflow, such as local testing or deploying to the AWS Cloud.

Usage

```
$ sam build <arguments> <options>
```

Arguments

Resource ID

Optional. Instructs AWS SAM to build a single resource declared in an [AWS SAM template](#). The build artifacts for the specified resource will be the only ones available for subsequent commands in the workflow, i.e. `sam package` and `sam deploy`.

Options

`--base-dir, -s DIRECTORY`

Resolves relative paths to the function's or layer's source code with respect to this directory. Use this option if you want to change how relative paths to source code folders are resolved. By default, relative paths are resolved with respect to the AWS SAM template's location.

In addition to the resources in the root application or stack you are building, this option also applies nested applications or stacks.

This option applies to the following resource types and properties:

- Resource type: `AWS::Serverless::Function` Property: `CodeUri`
- Resource type: `AWS::Serverless::Function` Resource attribute: `Metadata` Entry: `DockerContext`
- Resource type: `AWS::Serverless::LayerVersion` Property: `ContentUri`
- Resource type: `AWS::Lambda::Function` Property: `Code`
- Resource type: `AWS::Lambda::LayerVersion` Property: `Content`

`--beta-features | --no-beta-features`

Allow or deny beta features.

`--build-dir, -b DIRECTORY`

The path to a directory where the built artifacts are stored. This directory and all of its content are removed with this option.

`--build-image` *TEXT*

The URI of the container image that you want to pull for the build. By default, AWS SAM pulls the container image from Amazon ECR Public. Use this option to pull the image from another location.

You can specify this option multiple times. Each instance of this option can take either a string or a key-value pair. If you specify a string, it is the URI of the container image to use for all resources in your application. For example, `sam build --use-container --build-image amazon/aws-sam-cli-build-image-python3.8`. If you specify a key-value pair, the key is the resource name, and the value is the URI of the container image to use for that resource. For example `sam build --use-container --build-image Function1=amazon/aws-sam-cli-build-image-python3.8`. With key-value pairs, you can specify different container images for different resources.

This option only applies if the `--use-container` option is specified, otherwise an error will result.

`--build-in-source | --no-build-in-source`

Provide `--build-in-source` to build your project directly in the source folder.

The `--build-in-source` option supports the following runtimes and build methods:

- **Runtimes** – Any Node.js runtime supported by the [sam init --runtime](#) option.
- **Build methods** – `Makefile`, `esbuild`.

The `--build-in-source` option is not compatible with the following options:

- `--hook-name`
- `--use-container`

Default: `--no-build-in-source`

`--cached | --no-cached`

Enable or disable cached builds. Use this option to reuse build artifacts that haven't changed from previous builds. AWS SAM evaluates whether you've changed any files in your project directory. By default, builds are not cached. If the `--no-cached` option is invoked, it overrides the `cached = true` setting in samcofig.toml.

> ⓘ **Note**
>
> AWS SAM doesn't evaluate whether you've changed third-party modules that your
> project depends on, where you haven't provided a specific version. For example, if your
> Python function includes a `requirements.txt` file with the entry `requests=1.x`,
> and the latest request module version changes from `1.1` to `1.2`, then AWS SAM doesn't
> pull the latest version until you run a non-cached build.

`--cache-dir`

The directory where the cache artifacts are stored when `--cached` is specified. The default
cache directory is `.aws-sam/cache`.

`--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.

`--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.

`--container-env-var, -e` *TEXT*

Environment variables to pass to the build container. You can specify this option multiple
times. Each instance of this option takes a key-value pair, where the key is the resource and
environment variable, and the value is the environment variable's value. For example: `--
container-env-var Function1.GITHUB_TOKEN=TOKEN1 --container-env-var
Function2.GITHUB_TOKEN=TOKEN2`.

This option only applies if the `--use-container` option is specified, otherwise an error will
result.

`--container-env-var-file, -ef` *PATH*

The path and file name of a JSON file that contains values for the container's environment
variables. For more information about container environment variable files, see Container
environment variable file.

This option only applies if the `--use-container` option is specified, otherwise an error will result.

`--debug`

Turns on debug logging to print debug messages that the AWS SAM CLI generates, and to display timestamps.

`--docker-network` *TEXT*

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.

`--exclude, -x`

The Name of the resource(s) to exclude from the `sam build`. For example, if your template contains `Function1`, `Function2`, and `Function3` and you run `sam build --exclude Function2`, only `Function1` and `Function3` will be built.

`--help`

Shows this message and exits.

`--hook-name` *TEXT*

The name of the hook that is used to extend AWS SAM CLI functionality.

Accepted values: `terraform`.

`--manifest , -m` *PATH*

The path to a custom dependency manifest file (for example, package.json) to use instead of the default.

`--parallel`

Enabled parallel builds. Use this option to build your AWS SAM template's functions and layers in parallel. By default, the functions and layers are built in sequence.

`--parameter-overrides`

(Optional) A string that contains AWS CloudFormation parameter overrides encoded as key-value pairs. Uses the same format as the AWS Command Line Interface (AWS CLI). For example: `'ParameterKey=KeyPairName, ParameterValue=MyKey ParameterKey=InstanceType, ParameterValue=t1.micro'`. This option is not compatible with `--hook-name`.

`--profile` *TEXT*

> The specific profile from your credential file that gets AWS credentials.

`--region` *TEXT*

> The AWS Region to deploy to. For example, us-east-1.

`--save-params`

> Save the parameters that you provide at the command line to the AWS SAM configuration file.

`--skip-prepare-infra`

> Skips the preparation stage if no infrastructure changes have been made. Use with the `--hook-name` option.

`--skip-pull-image`

> Specifies whether the command should skip pulling down the latest Docker image for the Lambda runtime.

`--template-file, --template, -t` *PATH*

> The path and file name of AWS SAM template file [`default: template.[yaml|yml]`]. This option is not compatible with `--hook-name`.

`--terraform-project-root-path`

> The relative or absolute path to the top-level directory containing your Terraform configuration files or function source code. If these files are located outside of the directory containing your Terraform root module, use this option to specify its absolute or relative path. This option requires that `--hook-name` be set to `terraform`.

`--use-container, -u`

> If your functions depend on packages that have natively compiled dependencies, use this option to build your function inside a Lambda-like Docker container.

sam delete

This page provides reference information for the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam delete` command.

For an introduction to the AWS SAM CLI, see [What is the AWS SAM CLI?](#).

The `sam delete` command deletes an AWS SAM application by deleting the AWS CloudFormation stack, the artifacts that were packaged and deployed to Amazon S3 and Amazon ECR, and the AWS SAM template file.

This command also checks whether there is an Amazon ECR companion stack deployed, and if so prompts the user about deleting that stack and Amazon ECR repositories. If `--no-prompts` is specified, then companion stacks and Amazon ECR repositories are deleted by default.

Usage

```
$ sam delete <options>
```

Options

`--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](#).

`--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](#).

`--debug`

Turns on debug logging to print the debug message that the AWS SAM CLI generates and to display timestamps.

`--help`

Shows this message and exits.

`--no-prompts`

Specify this option to have AWS SAM operate in non-interactive mode. The stack name must be provided, either with the `--stack-name` option, or in the configuration `toml` file.

`--profile` *TEXT*

The specific profile from your credential file that gets AWS credentials.

`--region` *TEXT*

> The AWS Region to deploy to. For example, us-east-1.

`--s3-bucket`

> The path of the Amazon S3 bucket you want to delete.

`--s3-prefix`

> The prefix of the Amazon S3 bucket you want to delete.

`--save-params`

> Save the parameters that you provide at the command line to the AWS SAM configuration file.

`--stack-name` *TEXT*

> The name of the AWS CloudFormation stack that you want to delete.

sam deploy

This page provides reference information for the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam deploy` command.

- For an introduction to the AWS SAM CLI, see [What is the AWS SAM CLI?](#).

- For documentation on using the AWS SAM CLI `sam deploy` command, see [Using sam deploy](#).

The `sam deploy` command deploys an application to the AWS Cloud using AWS CloudFormation.

Usage

```
$ <environment variables> sam deploy <options>
```

Environment variables

`SAM_CLI_POLL_DELAY`

> Set the `SAM_CLI_POLL_DELAY` environment variable with a value of seconds to configure how often the AWS SAM CLI checks the AWS CloudFormation stack state, which is useful when seeing throttling from AWS CloudFormation. This env variable is used for polling `describe_stack` API calls, which are made while running `sam deploy`.

The following is an example of this variable:

```
$ SAM_CLI_POLL_DELAY=5 sam deploy
```

Options

`--capabilities` *LIST*

A list of capabilities that you must specify to allow AWS CloudFormation to create certain stacks. Some stack templates might include resources that 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 option. The only valid values are CAPABILITY_IAM and CAPABILITY_NAMED_IAM. If you have IAM resources, then you can specify either capability. If you have IAM resources with custom names, then you must specify CAPABILITY_NAMED_IAM. If you don't specify this option, then the operation returns an InsufficientCapabilities error.

`--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.

`--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.

`--confirm-changeset | --no-confirm-changeset`

Prompt to confirm whether the AWS SAM CLI deploys the computed changeset.

`--debug`

Turn on debug logging to print the debug message that the AWS SAM CLI generates and to display timestamps.

`--disable-rollback | --no-disable-rollback`

Specify whether to roll back your AWS CloudFormation stack if an error occurs during a deployment. By default, if there's an error during a deployment, your AWS CloudFormation stack rolls back to the last stable state. If you specify `--disable-rollback` and an error

occurs during a deployment, then resources that were created or updated before the error occurred aren't rolled back.

`--fail-on-empty-changeset | --no-fail-on-empty-changeset`

Specify whether to return a non-zero exit code if there are no changes to make to the stack. The default behavior is to return a non-zero exit code.

`--force-upload`

Specify this option to upload artifacts even if they match existing artifacts in the Amazon S3 bucket. Matching artifacts are overwritten.

`--guided, -g`

Specify this option to have the AWS SAM CLI use prompts to guide you through the deployment.

`--help`

Show this message and exit.

`--image-repositories` *TEXT*

A mapping of functions to their Amazon ECR repository URI. Reference functions by their logical ID. The following is an example:

```
$ sam deploy --image-repositories Function1=123456789012.dkr.ecr.us-east-1.amazonaws.com/my-repo
```

You can specify this option multiple times in a single command.

`--image-repository` *TEXT*

The name of the Amazon ECR repository where this command uploads your function's image. This option is required for functions declared with the `Image` package type.

`--kms-key-id` *TEXT*

The ID of an AWS Key Management Service (AWS KMS) key used to encrypt artifacts that are at rest in the Amazon S3 bucket. If you don't specify this option, then AWS SAM uses Amazon S3-managed encryption keys.

`--metadata`

A map of metadata to attach to all artifacts that are referenced in your template.

`--no-execute-changeset`

Indicates whether to apply the changeset. Specify this option if you want to view your stack changes before applying the changeset. This command creates an AWS CloudFormation changeset and then exits without applying the changeset. To apply the changeset, run the same command without this option.

`--no-progressbar`

Do not display a progress bar when uploading artifacts to Amazon S3.

`--notification-arns` *LIST*

A list of Amazon Simple Notification Service (Amazon SNS) topic ARNs that AWS CloudFormation associates with the stack.

`--on-failure [ROLLBACK | DELETE | DO_NOTHING]`

Specify the action to take when a stack fails to create.

The following options are available:

- `ROLLBACK` – Rolls back the stack to a previous known good state.

- `DELETE` – Rolls back the stack to a previous known good state, if one exists. Otherwise, deletes the stack.

- `DO_NOTHING` – Neither rolls back nor deletes the stack. The effect is the same as that of `--disable-rollback`.

The default behavior is ROLLBACK.

> **ⓘ Note**
>
> You can specify either the `--disable-rollback` option or the `--on-failure` option, but not both.

`--parameter-overrides`

A string that contains AWS CloudFormation parameter overrides encoded as key-value pairs. Use the same format as the AWS Command Line Interface (AWS CLI). For example, `ParameterKey=ParameterValue InstanceType=t1.micro`.

`--profile` *TEXT*

The specific profile from your credential file that gets AWS credentials.

`--region` *TEXT*

The AWS Region to deploy to. For example, us-east-1.

`--resolve-image-repos`

Automatically create Amazon ECR repositories to use for packaging and deploying for non-guided deployments. This option applies only to functions and layers with `PackageType:` `Image` specified. If you specify the `--guided` option, then the AWS SAM CLI ignores `--resolve-image-repos`.

> **ⓘ Note**
>
> If AWS SAM automatically creates any Amazon ECR repositories for functions or layers with this option, and you later delete those functions or layers from your AWS SAM template, then the corresponding Amazon ECR repositories are automatically deleted.

`--resolve-s3`

Automatically create an Amazon S3 bucket to use for packaging and deploying for non-guided deployments. If you specify the `--guided` option, then the AWS SAM CLI ignores `--resolve-s3`. If you specify both the `--s3-bucket` and `--resolve-s3` options, then an error occurs.

`--role-arn` *TEXT*

The Amazon Resource Name (ARN) of an IAM role that AWS CloudFormation assumes when applying the changeset.

`--s3-bucket` *TEXT*

The name of the Amazon S3 bucket where this command uploads your AWS CloudFormation template. If your template is larger than 51,200 bytes, then either the `--s3-bucket` option or the `--resolve-s3` option is required. If you specify both the `--s3-bucket` and `--resolve-s3` options, then an error occurs.

`--s3-prefix` *TEXT*

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.

`--save-params`

Save the parameters that you provide at the command line to the AWS SAM configuration file.

`--signing-profiles` *LIST*

The list of signing profiles to sign your deployment packages with. This option takes a list of key-value pairs, where the key is the name of the function or layer to sign, and the value is the signing profile, with an optional profile owner delimited with :. For example, `FunctionNameToSign=SigningProfileName1 LayerNameToSign=SigningProfileName2:SigningProfileOwner`.

`--stack-name` *TEXT*

(Required) The name of the AWS CloudFormation stack that you're deploying to. If you specify an existing stack, then the command updates the stack. If you specify a new stack, then the command creates it.

`--tags` *LIST*

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 that support it.

`--template-file, --template, -t` *PATH*

The path and file name where your AWS SAM template is located.

> ⓘ **Note**
>
> If you specify this option, then AWS SAM deploys only the template and the local resources that it points to.

`--use-json`

Output JSON for the AWS CloudFormation template. The default output is YAML.

sam init

This page provides reference information for the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam init` command.

- For an introduction to the AWS SAM CLI, see What is the AWS SAM CLI?.

- For documentation on using the AWS SAM CLI `sam init` command, see [Using sam init](#).

The `sam init` command provides options to initialize a new serverless application.

Usage

```
$ sam init <options>
```

Options

`--app-template` *TEXT*

The identifier of the managed application template that you want to use. If you're not sure, call `sam init` without options for an interactive workflow.

This parameter is required if `--no-interactive` is specified and `--location` is not provided.

This parameter is available only in AWS SAM CLI version 0.30.0 and later. Specifying this parameter with an earlier version results in an error.

`--application-insights` | `--no-application-insights`

Activate Amazon CloudWatch Application Insights monitoring for your application. To learn more, see [Monitor your serverless applications with CloudWatch Application Insights](#).

The default option is `--no-application-insights`.

`--architecture, -a` *[x86_64 | arm64]*

The instruction set architecture for your application's Lambda functions. Specify one of `x86_64` or `arm64`.

`--base-image` *[amazon/dotnet8-base | amazon/dotnet6-base | amazon/ dotnetcore3.1-base | amazon/go1.x-base | amazon/java21-base | amazon/ java17-base | amazon/java11-base | amazon/java8.al2-base | amazon/java8- base | amazon/nodejs20.x-base | amazon/nodejs18.x-base | amazon/nodejs16.x- base | | amazon/python3.12-base | amazon/python3.11-base | amazon/ python3.10-base | amazon/python3.9-base | amazon/python3.8-base | amazon/ ruby3.3-base | amazon/ruby3.2-base]*

The base image of your application. This option applies only when the package type is `Image`.

This parameter is required if `--no-interactive` is specified, `--package-type` is specified as Image, and `--location` is not specified.

`--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](#).

`--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](#).

`--debug`

Turns on debug logging to print debug messages that the AWS SAM CLI generates, and to display timestamps.

`--dependency-manager, -d` *[gradle | mod | maven | bundler | npm | cli-package | pip]*

The dependency manager of your Lambda runtime.

`--extra-content`

Override any custom parameters in the template's `cookiecutter.json` configuration, for example, {"customParam1": "customValue1", "customParam2":"customValue2"}.

`--help, -h`

Shows this message and exits.

`--location, -l` *TEXT*

The template or application location (Git, Mercurial, HTTP/HTTPS, .zip file, path).

This parameter is required if `--no-interactive` is specified and `--runtime`, `--name`, and `--app-template` are not provided.

For Git repositories, you must use the location of the root of the repository.

For local paths, the template must be in either .zip file or [Cookiecutter](#) format.

`--name, -n` *TEXT*

The name of your project to be generated as a directory.

This parameter is required if `--no-interactive` is specified and `--location` is not provided.

`--no-input`

Disables Cookiecutter prompting and accepts the vcfdefault values that are defined in the template configuration.

`--no-interactive`

Disable interactive prompting for init parameters, and fail if any required values are missing.

`--output-dir, -o` *PATH*

The location where the initialized application is output.

`--package-type` *[Zip | Image]*

The package type of the example application. `Zip` creates a .zip file archive, and `Image` creates a container image.

`--runtime, -r` *[dotnet8 | dotnet6 | dotnetcore3.1 | go1.x | java21 | java17 | java11 | java8 | java8.al2 | nodejs20.x | nodejs18.x | nodejs16.x | python3.12 | python3.11 | python3.10 | python3.9 | python3.8 | ruby3.3 | ruby3.2]*

The Lambda runtime of your application. This option applies only when the package type is `Zip`.

This parameter is required if `--no-interactive` is specified, `--package-type` is specified as `Zip`, and `--location` is not specified.

`--save-params`

Save the parameters that you provide at the command line to the AWS SAM configuration file.

`--tracing | --no-tracing`

Activate AWS X-Ray tracing for your Lambda functions.

sam list

This page provides reference information for the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam list` command.

For an introduction to the AWS SAM CLI, see [What is the AWS SAM CLI?](#).

The `sam list` command outputs important information about the resources in your serverless application and the state of your serverless application. Use **sam list** before and after deployment to assist during local and cloud development.

Usage

```
$ sam list <options> <subcommand>
```

Options

`--help, -h`

　　Show this message and exit.

Subcommands

`endpoints`

　　Displays a list of cloud and local endpoints from your AWS CloudFormation stack. For more information, see sam list endpoints.

`resources`

　　Displays the resources in your AWS Serverless Application Model (AWS SAM) template that are created in AWS CloudFormation at deployment. For more information, see sam list resources.

`stack-outputs`

　　Displays the outputs of your AWS CloudFormation stack from an AWS SAM or AWS CloudFormation template. For more information, see sam list stack-outputs.

sam list endpoints

This page provides reference information for the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam list endpoints` subcommand.

For an introduction to the AWS SAM CLI, see What is the AWS SAM CLI?.

The `sam list endpoints` subcommand displays a list of cloud and local endpoints from your AWS CloudFormation stack. You can interact with these resources through the **sam local** and **sam sync** commands.

AWS Lambda and Amazon API Gateway resource types are supported with this command.

> ⓘ **Note**
>
> Custom domains are supported when configured for your Amazon API Gateway resources. This command will output the custom domain instead of the default endpoint.

Usage

```
$ sam list endpoints <options>
```

Options

`--config-env` *TEXT*

The environment name specifying the default parameter values in the configuration file to use.

Default value: `default`

For more information about configuration files, see [AWS SAM CLI configuration file](#).

`--config-file` *TEXT*

The path and file name of the configuration file containing default parameter values to use.

Default value: `samconfig.toml` in current working directory.

For more information about configuration files, see [AWS SAM CLI configuration file](#).

`--debug`

Turn on debug logging to print debug messages generated by the AWS SAM CLI with timestamps.

`--help, -h`

Show this message and exit.

`--output [json|table]`

Specify the format to output results.

Default value: `table`

`--profile` *TEXT*

Select a specific profile from your credential file to get AWS credentials.

`--region` *TEXT*

Set the AWS region of the service. For example, `us-east-1`.

`--save-params`

Save the parameters that you provide at the command line to the AWS SAM configuration file.

`--stack-name` *TEXT*

Name of the deployed AWS CloudFormation stack. The stack name can be found in your application's `samconfig.toml` file or designated configuration file.

When this option is not specified, local resources defined in your template will display.

`--template-file, --template, -t` *PATH*

AWS SAM template file.

Default value: `template.[yaml|yml|json]`

Examples

Display an output, in json format, of deployed resource endpoints from your AWS CloudFormation stack named `test-stack`.

```
$ sam list endpoints --stack-name test-stack --output json

[
  {
    "LogicalResourceId": "HelloWorldFunction",
    "PhysicalResourceId": "sam-app-test-list-HelloWorldFunction-H85Y7yIV7ZLq",
    "CloudEndpoint": "https://zt55oi7kbljxjmcoahsj3cknwu0rposq.lambda-url.us-
east-1.on.aws/",
    "Methods": "-"
  },
  {
    "LogicalResourceId": "ServerlessRestApi",
    "PhysicalResourceId": "uj80uoe2o2",
    "CloudEndpoint": [
      "https://uj80uoe2o2.execute-api.us-east-1.amazonaws.com/Prod",
      "https://uj80uoe2o2.execute-api.us-east-1.amazonaws.com/Stage"
```

```
    ],
    "Methods": [
      "/hello['get']"
    ]
  }
]
```

sam list resources

This page provides reference information for the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam list resources` subcommand.

For an introduction to the AWS SAM CLI, see [What is the AWS SAM CLI?](#).

The `sam list resources` subcommand displays the resources in your AWS Serverless Application Model (AWS SAM) template that are created in AWS CloudFormation by the AWS SAM transform at deployment.

Use **sam list resources** with an AWS SAM template before deployment to see resources that will be created. Provide an AWS CloudFormation stack name to view a consolidated list that includes deployed resources.

> **ⓘ Note**
>
> To generate a list of resources from your AWS SAM template, a local transform of your template is performed. Resources that will be deployed with conditions, such as within a specific region, are included in this list.

Usage

```
$ sam list resources <options>
```

Options

`--config-env` *TEXT*

The environment name specifying the default parameter values in the configuration file to use.

Default value: `default`

For more information about configuration files, see [AWS SAM CLI configuration file](#).

`--config-file` *TEXT*

The path and file name of the configuration file containing default parameter values to use.

Default value: `samconfig.toml` in current working directory.

For more information about configuration files, see [AWS SAM CLI configuration file](#).

`--debug`

Turn on debug logging to print debug messages generated by the AWS SAM CLI with timestamps.

`--help, -h`

Show this message and exit.

`--output [json|table]`

Specify the format to output results.

Default value: `table`

`--profile` *TEXT*

Select a specific profile from your credential file to get AWS credentials.

`--region` *TEXT*

Set the AWS region of the service. For example, `us-east-1`.

`--save-params`

Save the parameters that you provide at the command line to the AWS SAM configuration file.

`--stack-name` *TEXT*

Name of the deployed AWS CloudFormation stack. The stack name can be found in your application's `samconfig.toml` file or designated configuration file.

When provided, resource logical IDs from your template will be mapped to their corresponding physical IDs in AWS CloudFormation. To learn more about physical IDs, see [Resource fields](#) in the *AWS CloudFormation User Guide*.

When this option is not specified, local resources defined in your template will display.

`--template-file, --template, -t` *PATH*

AWS SAM template file.

Default value: `template.[yaml|yml|json]`

Examples

Display an output, in table format, of local resources from your AWS SAM template and deployed resources from your AWS CloudFormation stack named `test-stack`. Run from the same directory as your local template.

```
$ sam list resources --stack-name test-stack --output table


----------------------------------------------------------------------------------------
Logical ID                                                    Physical ID
----------------------------------------------------------------------------------------
HelloWorldFunction                                            sam-app-test-list-
HelloWorldFunction-H85Y7yIV7ZLq
HelloWorldFunctionHelloWorldPermissionProd                    sam-app-test-list-

 HelloWorldFunctionHelloWorldPermissionProd-1QH7CPOCBL2IK
HelloWorldFunctionRole                                        sam-app-test-list-
HelloWorldFunctionRole-SRJDMJ6F7F41
ServerlessRestApi                                             uj80uoe2o2
ServerlessRestApiDeployment47fc2d5f9d                         pncw5f
ServerlessRestApiProdStage                                    Prod
ServerlessRestApiDeploymentf5716dc08b                         -
----------------------------------------------------------------------------------------
```

sam list stack-outputs

This page provides reference information for the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam list stack-outputs` subcommand.

For an introduction to the AWS SAM CLI, see What is the AWS SAM CLI?.

The `sam list stack-outputs` subcommand displays the outputs of your AWS CloudFormation stack from an AWS Serverless Application Model (AWS SAM) or AWS CloudFormation template. For more information on `Outputs`, see Outputs in the *AWS CloudFormation User Guide*.

Usage

```
$ sam list stack-outputs <options>
```

Options

`--config-env` *TEXT*

The environment name specifying the default parameter values in the configuration file to use.

Default value: `default`

For more information about configuration files, see [AWS SAM CLI configuration file](#).

`--config-file` *TEXT*

The path and file name of the configuration file containing default parameter values to use.

Default value: `samconfig.toml` in current working directory.

For more information about configuration files, see [AWS SAM CLI configuration file](#).

`--debug`

Turn on debug logging to print debug messages generated by the AWS SAM CLI with timestamps.

`--help, -h`

Show this message and exit.

`--output [json|table]`

Specify the format to output results.

Default value: `table`

`--profile` *TEXT*

Select a specific profile from your credential file to get AWS credentials.

`--region` *TEXT*

Set the AWS region of the service. For example, `us-east-1`.

`--save-params`

Save the parameters that you provide at the command line to the AWS SAM configuration file.

`--stack-name` *TEXT*

Name of the deployed AWS CloudFormation stack. The stack name can be found in your application's `samconfig.toml` file or designated configuration file.

This option is required.

Examples

Displays the outputs, in table format, of resources in your AWS CloudFormation stack named
`test-stack`.

```
$ sam list stack-outputs --stack-name test-stack --output table


-------------------------------------------------------------------------------------
OutputKey                                OutputValue
 Description
-------------------------------------------------------------------------------------
HelloWorldFunctionIamRole                arn:aws:iam::account-number:role/sam-
 Implicit IAM Role created for Hello
                                         app-test-list-HelloWorldFunctionRole-    World
 function
                                         SRJDMJ6F7F41
HelloWorldApi                            https://uj80uoe2o2.execute-api.us-       API
 Gateway endpoint URL for Prod
                                         east-1.amazonaws.com/Prod/hello/         stage
 for Hello World function
HelloWorldFunction                       arn:aws:lambda:us-                       Hello
 World Lambda Function ARN
                                         east-1:account-number:function:sam-app-
                                         test-list-
                                         HelloWorldFunction-H85Y7yIV7ZLq
-------------------------------------------------------------------------------------
```

sam local generate-event

This page provides reference information for the AWS Serverless Application Model Command Line
Interface (AWS SAM CLI) `sam local generate-event` subcommand.

- For an introduction to the AWS SAM CLI, see [What is the AWS SAM CLI?](#).

- For documentation on using the AWS SAM CLI `sam local generate-event` command, see
  [Using sam local generate-event](#).

The `sam local generate-event` subcommand generates event payload samples for supported
AWS services.

Usage

```
$ sam local generate-event <options> <service> <event> <event-options>
```

Options

`--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.

`--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.

`--help`

Shows this message and exits.

Service

To see a list of supported services, run the following:

```
$ sam local generate-event
```

Event

To see a list of supported events that can be generated for each service, run the following:

```
$ sam local generate-event <service>
```

Event options

To see a list of supported event options that you can modify, run the following:

```
$ sam local generate-event <service> <event> --help
```

sam local invoke

This page provides reference information for the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam local invoke` subcommand.

- For an introduction to the AWS SAM CLI, see [What is the AWS SAM CLI?](#).

- For documentation on using the AWS SAM CLI `sam local invoke` subcommand, see [Using sam local invoke](#).

The `sam local invoke` subcommand initiates a one-time invocation of an AWS Lambda function locally.

Usage

```
$ sam local invoke <arguments> <options>
```

> ⓘ **Note**
>
> If you have more than one function defined in your AWS SAM template, provide the function logical ID that you want to invoke.

Arguments

Resource ID

The ID of the Lambda function to invoke.

This argument is optional. If your application contains a single Lambda function, the AWS SAM CLI will invoke it. If your application contains multiple functions, provide the ID of the function to invoke.

Valid values: The resource's logical ID or resource ARN.

Options

`--add-host` *LIST*

Passes a hostname to IP address mapping to the Docker container's host file. This parameter can be passed multiple times.

Example

Example: `--add-host` *example.com:127.0.0.1*

`--beta-features | --no-beta-features`

Allow or deny beta features.

`--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](#).

`--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](#).

`--container-env-vars`

(Optional) Pass environment variables to the Lambda function image container when debugging locally.

`--container-host` *TEXT*

Host of locally emulated Lambda container. The default value is `localhost`. If you want to run AWS SAM CLI in a Docker container on macOS, you can specify `host.docker.internal`. If you want to run the container on a different host than AWS SAM CLI, you can specify the IP address of the remote host.

`--container-host-interface` *TEXT*

The IP address of the host network interface that container ports should bind to. The default value is `127.0.0.1`. Use `0.0.0.0` to bind to all interfaces.

`--debug`

Turns on debug logging to print debug messages that the AWS SAM CLI generates, and to display timestamps.

`--debug-args` *TEXT*

Additional arguments to pass to the debugger.

`--debug-port, -d` *TEXT*

  When specified, starts the Lambda function container in debug mode and exposes this port on the local host.

`--debugger-path` *TEXT*

  The host path to a debugger that's mounted into the Lambda container.

`--docker-network` *TEXT*

  The name or ID of an existing Docker network that Lambda Docker containers should connect to, along with the default bridge network. If this isn't specified, the Lambda containers connect only to the default bridge Docker network.

`--docker-volume-basedir, -v` *TEXT*

  The location of the base directory where the AWS SAM file exists. If Docker is running on a remote machine, you must mount the path where the AWS SAM file exists on the Docker machine and modify this value to match the remote machine.

`--env-vars, -n` *PATH*

  The JSON file that contains values for the Lambda function's environment variables. For more information about environment variable files, see [Environment variable file](#).

`--event, -e` *PATH*

  The JSON file that contains event data that's passed to the Lambda function when it's invoked. If you don't specify this option, no event is assumed. To input JSON from `stdin`, you must pass in the value '-'. For details about event message formats from different AWS services, see [Working with other services](#) in the *AWS Lambda Developer Guide*.

`--force-image-build`

  Specifies whether the AWS SAM CLI should rebuild the image used for invoking Lambda functions with layers.

`--help`

  Shows this message and exits.

`--hook-name` `TEXT`

  The name of the hook that is used to extend AWS SAM CLI functionality.

  Accepted values: `terraform`.

`--invoke-image` *TEXT*

> The URI of the container image that you want to use for the local function invocation. By default, AWS SAM pulls the container image from Amazon ECR Public. Use this option to pull the image from another location.

> For example, `sam local invoke MyFunction --invoke-image amazon/aws-sam-cli-emulation-image-python3.8`.

`--layer-cache-basedir` *DIRECTORY*

> Specifies the location of the base directory where the layers that your template uses are downloaded to.

`--log-file, -l` *TEXT*

> The log file to send runtime logs to.

`--no-event`

> Invokes the function with an empty event.

`--parameter-overrides`

> (Optional) A string that contains AWS CloudFormation parameter overrides encoded as key-value pairs. Uses the same format as the AWS Command Line Interface (AWS CLI). For example: `'ParameterKey=KeyPairName, ParameterValue=MyKey ParameterKey=InstanceType, ParameterValue=t1.micro'`.

> This option is not compatible with `--hook-name`.

`--profile` *TEXT*

> The specific profile from your credential file that gets AWS credentials.

`--region` *TEXT*

> The AWS Region to deploy to. For example, us-east-1.

`--save-params`

> Save the parameters that you provide at the command line to the AWS SAM configuration file.

`--shutdown`

> Emulates a shutdown event after the invoke completes, in order to test extension handling of shutdown behavior.

`--skip-prepare-infra`

> Skips the preparation stage if no infrastructure changes have been made. Use with the `--hook-name` option.

`--skip-pull-image`

> By default, the AWS SAM CLI checks Lambda's latest remote runtime environment and updates your local image automatically to keep in sync.

> Specify this option to skip pulling down the latest Docker image for your Lambda runtime environment.

`--template, -t` *PATH*

> The AWS SAM template file.

> This option is not compatible with `--hook-name`.

> > ⓘ **Note**
> >
> > If you specify this option, AWS SAM loads only the template and the local resources that it points to.

`--terraform-plan-file`

> The relative or absolute path to your local Terraform plan file when using the AWS SAM CLI with Terraform Cloud. This option requires that `--hook-name` be set to `terraform`.

sam local start-api

This page provides reference information for the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam local start-api` subcommand.

- For an introduction to the AWS SAM CLI, see [What is the AWS SAM CLI?](#).
- For documentation on using the AWS SAM CLI `sam local start-api` subcommand, see [Using sam local start-api](#).

The `sam local start-api` subcommand runs your AWS Lambda functions locally to test through a local HTTP server host.

Usage

```
$ sam local start-api <options>
```

Options

`--add-host` *LIST*

Passes a hostname to IP address mapping to the Docker container's host file. This parameter can be passed multiple times.

Example

Example: `--add-host` *example.com:127.0.0.1*

`--beta-features | --no-beta-features`

Allow or deny beta features.

`--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](#).

`--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](#).

`--container-env-vars`

Optional. Pass environment variables to image container when locally debugging.

`--container-host` *TEXT*

Host of locally emulated Lambda container. The default value is `localhost`. If you want to run AWS SAM CLI in a Docker container on macOS, you can specify `host.docker.internal`. If you want to run the container on a different host than AWS SAM CLI, you can specify the IP address of the remote host.

`--container-host-interface` *TEXT*

The IP address of the host network interface that container ports should bind to. The default value is `127.0.0.1`. Use `0.0.0.0` to bind to all interfaces.

`--debug`

Turns on debug logging to print debug message generated by the AWS SAM CLI and display timestamps.

`--debug-args` *TEXT*

Additional arguments to be passed to the debugger.

`--debug-function`

Optional. Specifies the Lambda function to apply debug options to when `--warm-containers` is specified. This parameter applies to `--debug-port`, `--debugger-path`, and `--debug-args`.

`--debug-port, -d` *TEXT*

When specified, starts the Lambda function container in debug mode and exposes this port on the local host.

`--debugger-path` *TEXT*

The host path to a debugger that will be mounted into the Lambda container.

`--docker-network` *TEXT*

The name or ID of an existing Docker network that the Lambda Docker containers should connect to, along with the default bridge network. If this isn't specified, the Lambda containers only connect to the default bridge Docker network.

`--docker-volume-basedir, -v` *TEXT*

The location of the base directory where the AWS SAM file exists. If Docker is running on a remote machine, you must mount the path where the AWS SAM file exists on the Docker machine, and modify this value to match the remote machine.

`--env-vars, -n` *PATH*

The JSON file that contains values for the Lambda function's environment variables.

`--force-image-build`

Specifies whether AWS SAM CLI should rebuild the image used for invoking functions with layers.

```
--help
```

Shows this message and exits.

```
--hook-name TEXT
```

The name of the hook that is used to extend AWS SAM CLI functionality.

Accepted values: `terraform`.

```
--host TEXT
```

The local hostname or IP address to bind to (default: '127.0.0.1').

```
--invoke-image TEXT
```

The URI of the container image that you want to use for your Lambda functions. By default, AWS SAM pulls the container image from Amazon ECR Public. Use this option to pull the image from another location.

You can specify this option multiple times. Each instance of this option can take either a string or a key-value pair. If you specify a string, it is the URI of the container image to use for all functions in your application. For example, `sam local start-api --invoke-image public.ecr.aws/sam/emu-python3.8`. If you specify a key-value pair, the key is the resource name, and the value is the URI of the container image to use for that resource. For example `sam local start-api --invoke-image public.ecr.aws/sam/emu-python3.8 --invoke-image Function1=amazon/aws-sam-cli-emulation-image-python3.8` . With key-value pairs, you can specify different container images for different resources.

```
--layer-cache-basedir DIRECTORY
```

Specifies the location basedir where the Layers your template uses are downloaded to.

```
--log-file, -l TEXT
```

The log file to send runtime logs to.

```
--parameter-overrides
```

Optional. A string that contains AWS CloudFormation parameter overrides encoded as key-value pairs. Use the same format as the AWS CLI—for example, 'ParameterKey=KeyPairName, ParameterValue=MyKey ParameterKey=InstanceType,ParameterValue=t1.micro'.

`--port, -p` *INTEGER*

　　The local port number to listen on (default: '3000').

`--profile` *TEXT*

　　The specific profile from your credential file that gets AWS credentials.

`--region` *TEXT*

　　The AWS Region to deploy to. For example, us-east-1.

`--save-params`

　　Save the parameters that you provide at the command line to the AWS SAM configuration file.

`--shutdown`

　　Emulates a shutdown event after the invoke completes, in order to test extension handling of
　　shutdown behavior.

`--skip-prepare-infra`

　　Skips the preparation stage if no infrastructure changes have been made. Use with the `--`
　　`hook-name` option.

`--skip-pull-image`

　　Specifies whether the CLI should skip pulling down the latest Docker image for the Lambda
　　runtime.

`--ssl-cert-file` *PATH*

　　Path to SSL certificate file (default: None). When using this option, the `--ssl-key-file`
　　option must also be used.

`--ssl-key-file` *PATH*

　　Path to SSL key file (default: None). When using this option, the `--ssl-cert-file` option
　　must also be used.

`--static-dir, -s` *TEXT*

　　Any static asset (for example, CSS/JavaScript/HTML) files located in this directory are presented
　　at /.

`--template, -t` *PATH*

The AWS SAM template file.

> ℹ️ **Note**
>
> If you specify this option, AWS SAM loads only the template and the local resources that it points to.

`--terraform-plan-file`

The relative or absolute path to your local Terraform plan file when using the AWS SAM CLI with Terraform Cloud. This option requires that `--hook-name` be set to `terraform`.

`--warm-containers` *[EAGER | LAZY]*

Optional. Specifies how AWS SAM CLI manages containers for each function.

Two options are available:

EAGER: Containers for all functions are loaded at startup and persist between invocations.

LAZY: Containers are only loaded when each function is first invoked. Those containers persist for additional invocations.

sam local start-lambda

This page provides reference information for the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam local start-lambda` subcommand.

- For an introduction to the AWS SAM CLI, see [What is the AWS SAM CLI?](#).
- For documentation on using the AWS SAM CLI `sam local start-lambda` subcommand, see [Using sam local start-lambda](#).

The `sam local start-lambda` subcommand starts a local endpoint to emulate AWS Lambda.

Usage

```
$ sam local start-lambda <options>
```

Options

`--add-host` *LIST*

Passes a hostname to IP address mapping to the Docker container's host file. This parameter can be passed multiple times.

Example

Example: `--add-host` *example.com:127.0.0.1*

`--beta-features | --no-beta-features`

Allow or deny beta features.

`--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](#).

`--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](#).

`--container-env-vars`

Optional. Pass environment variables to image container when locally debugging.

`--container-host` *TEXT*

Host of locally emulated Lambda container. The default value is `localhost`. If you want to run AWS SAM CLI in a Docker container on macOS, you can specify `host.docker.internal`. If you want to run the container on a different host than AWS SAM CLI, you can specify the IP address of the remote host.

`--container-host-interface` *TEXT*

The IP address of the host network interface that container ports should bind to. The default value is `127.0.0.1`. Use `0.0.0.0` to bind to all interfaces.

`--debug`

Turns on debug logging to print debug message generated by the AWS SAM CLI and display timestamps.

`--debug-args` *TEXT*

Additional arguments to be passed to the debugger.

`--debug-function`

Optional. Specifies the Lambda function to apply debug options to when `--warm-containers` is specified. This parameter applies to `--debug-port`, `--debugger-path`, and `--debug-args`.

`--debug-port, -d` *TEXT*

When specified, starts the Lambda function container in debug mode, and exposes this port on the local host.

`--debugger-path` *TEXT*

The host path to a debugger to be mounted into the Lambda container.

`--docker-network` *TEXT*

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.

`--docker-volume-basedir, -v` *TEXT*

The location of the base directory where the AWS SAM file exists. If Docker is running on a remote machine, you must mount the path where the AWS SAM file exists on the Docker machine, and modify this value to match the remote machine.

`--env-vars, -n` *PATH*

The JSON file that contains values for the Lambda function's environment variables.

`--force-image-build`

Specify whether the CLI should rebuild the image used for invoking functions with layers.

`--help`

Shows this message and exits.

`--hook-name` *TEXT*

The name of the hook that is used to extend AWS SAM CLI functionality.

Accepted values: `terraform`.

`--host` *TEXT*

    The local hostname or IP address to bind to (default: '127.0.0.1').

`--invoke-image` *TEXT*

    The URI of the container image that you want to use for the local function invocation. By default, AWS SAM pulls the container image from Amazon ECR Public. Use this option to pull the image from another location.

    For example, `sam local start-lambda MyFunction --invoke-image amazon/aws-sam-cli-emulation-image-python3.8`.

`--layer-cache-basedir` *DIRECTORY*

    Specifies the location basedir where the layers your template uses are downloaded to.

`--log-file, -l` *TEXT*

    The log file to send runtime logs to.

`--parameter-overrides`

    Optional. A string that contains AWS CloudFormation parameter overrides encoded as key-value pairs. Use the same format as the AWS CLI—for example, 'ParameterKey=KeyPairName, ParameterValue=MyKey ParameterKey=InstanceType,ParameterValue=t1.micro'. This option is not compatible with `--hook-name`.

`--port, -p` *INTEGER*

    The local port number to listen on (default: '3001').

`--profile` *TEXT*

    The specific profile from your credential file that gets AWS credentials.

`--region` *TEXT*

    The AWS Region to deploy to. For example, us-east-1.

`--save-params`

    Save the parameters that you provide at the command line to the AWS SAM configuration file.

`--shutdown`

    Emulates a shutdown event after the invoke completes, in order to test extension handling of shutdown behavior.

`--skip-prepare-infra`

Skips the preparation stage if no infrastructure changes have been made. Use with the `--hook-name` option.

`--skip-pull-image`

Specifies whether the CLI should skip pulling down the latest Docker image for the Lambda runtime.

`--template, -t `*`PATH`*

The AWS SAM template file.

> **ⓘ Note**
>
> If you specify this option, AWS SAM loads only the template and the local resources that it points to. This option is not compatible with `--hook-name`.

`--terraform-plan-file`

The relative or absolute path to your local Terraform plan file when using the AWS SAM CLI with Terraform Cloud. This option requires that `--hook-name` be set to `terraform`.

`--warm-containers `*`[EAGER | LAZY]`*

Optional. Specifies how AWS SAM CLI manages containers for each function.

Two options are available:

- EAGER: Containers for all functions are loaded at startup and persist between invocations.
- LAZY: Containers are only loaded when each function is first invoked. Those containers persist for additional invocations.

sam logs

This page provides reference information for the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam logs` command.

For an introduction to the AWS SAM CLI, see [What is the AWS SAM CLI?](#).

The `sam logs` command fetches logs that are generated by your AWS Lambda functions.

Usage

```
$ sam logs <options>
```

Options

`--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](#).

`--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](#).

`--cw-log-group` *LIST*

Includes logs from the CloudWatch Logs log groups that you specify. If you specify this option along with `name`, AWS SAM includes logs from the specified log groups in addition to logs from the named resources.

`--debug`

Turns on debug logging to print debug message generated by the AWS SAM CLI and display timestamps.

`---end-time, e` *TEXT*

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'.

`--filter` *TEXT*

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, "error") or a pattern that's supported by Amazon CloudWatch Logs. For the syntax, see the [Amazon CloudWatch Logs documentation](#).

`--help`

Shows this message and exits.

`--include-traces`

Includes X-Ray traces in the log output.

`--name, -n` *TEXT*

The name of the resource for which to fetch logs. If this resource is part of an AWS CloudFormation stack, this can be the logical ID of the function resource in the AWS CloudFormation/AWS SAM template. Multiple names can be provided by repeating the parameter again. If resource is in a nested stack, the name can be prepended by the name of the nested stack name to pull logs from that resource (NestedStackLogicalId/ResourceLogicalId). If the resource name isn't given, the given stack will be scanned and log information will be pulled for all supported resources. If you don't specify this option, AWS SAM fetches logs for all resources in the stack that you specify. The following resource types are supported:

- `AWS::Serverless::Function`
- `AWS::Lambda::Function`
- `AWS::Serverless::Api`
- `AWS::ApiGateway::RestApi`
- `AWS::Serverless::HttpApi`
- `AWS::ApiGatewayV2::Api`
- `AWS::Serverless::StateMachine`
- `AWS::StepFunctions::StateMachine`

`--output` *TEXT*

Specifies the output format for logs. To print formatted logs, specify `text`. To print the logs as JSON, specify `json`.

`--profile` *TEXT*

The specific profile from your credential file that gets AWS credentials.

`--region` *TEXT*

The AWS Region to deploy to. For example, us-east-1.

`--save-params`

Save the parameters that you provide at the command line to the AWS SAM configuration file.

`--stack-name` *TEXT*

The name of the AWS CloudFormation stack that the resource is a part of.

`--start-time, -s` *TEXT*

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'.

`--tail, -t`

Tails the log output. This ignores the end time argument and continues to fetch logs as they become available.

Examples

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.

```
$ sam logs -n HelloWorldFunction --stack-name myStack
```

View logs for a specific time range using the -s (--start-time) and -e (--end-time) options.

```
$ sam logs -n HelloWorldFunction --stack-name myStack -s '10min ago' -e '2min ago'
```

You can also add the `--tail` option to wait for new logs and see them as they arrive.

```
$ sam logs -n HelloWorldFunction --stack-name myStack --tail
```

Use the `--filter` option to quickly find logs that match terms, phrases or values in your log events.

```
$ sam logs -n HelloWorldFunction --stack-name myStack --filter "error"
```

View the logs for a resource in a child stack.

```
$ sam logs --stack-name myStack -n childStack/HelloWorldFunction
```

Tail logs for all supported resources in your application.

```
$ sam logs --stack-name sam-app --tail
```

Fetch logs for a specific Lambda function and API Gateway API in your application.

```
$ sam logs --stack-name sam-app --name HelloWorldFunction --name HelloWorldRestApi
```

Fetch logs for all supported resources in your application, and additionally from the specified log groups.

```
$ sam logs --cw-log-group /aws/lambda/myfunction-123 --cw-log-group /aws/lambda/
myfunction-456
```

sam package

The AWS Serverless Application Model Command Line Interface (AWS SAM CLI) packages an AWS SAM application.

This command creates a `.zip` file of your code and dependencies, and uploads the file to Amazon Simple Storage Service (Amazon S3). AWS SAM enables encryption for all files stored in 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.

By default when you use this command, the AWS SAM CLI assumes that your current working directory is your project's root directory. The AWS SAM CLI first tries to locate a template file built using the sam build command, located in the `.aws-sam` subfolder, and named `template.yaml`. Next, the AWS SAM CLI tries to locate a template file named `template.yaml` or `template.yml` in the current working directory. If you specify the `--template` option, AWS SAM CLI's default behavior is overridden, and will package just that AWS SAM template and the local resources it points to.

> ⓘ **Note**
>
> sam deploy now implicitly performs the functionality of `sam package`. You can use the sam deploy command directly to package and deploy your application.

Usage

```
$ sam package <arguments> <options>
```

Arguments

Resource ID

The ID of the Lambda function to package.

This argument is optional. If your application contains a single Lambda function, the AWS SAM CLI will package it. If your application contains multiple functions, provide the ID of the function to package a single function.

Valid values: The resource's logical ID or resource ARN.

Options

`--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](#).

`--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](#).

`--debug`

Turns on debug logging to print debug message generated by the AWS SAM CLI and display timestamps.

`--force-upload`

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.

`--help`

Shows this message and exits.

`--image-repository` *TEXT*

The URI of the Amazon Elastic Container Registry (Amazon ECR) repository where this command uploads your function's image. Required for functions declared with the `Image` package type.

`--kms-key-id` *TEXT*

The ID of an AWS Key Management Service (AWS KMS) key used to encrypt artifacts that are at rest in the Amazon S3 bucket. If this option is not specified, then AWS SAM uses Amazon S3-managed encryption keys.

`--metadata`

(Optional) A map of metadata to attach to all artifacts that are referenced in your template.

`--no-progressbar`

Do not display a progress bar when uploading artifacts to Amazon S3.

`--output-template-file` *PATH*

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.

`--profile` *TEXT*

The specific profile from your credential file that gets AWS credentials.

`--region` *TEXT*

The AWS Region to deploy to. For example, us-east-1.

`--resolve-s3`

Automatically create an Amazon S3 bucket to use for packaging. If you specify both the `--s3-bucket` and `--resolve-s3` options, then an error will result.

`--s3-bucket` *TEXT*

The name of the Amazon S3 bucket where this command uploads your artifact. If your artifact is larger than 51,200 bytes, then either the `--s3-bucket` or the `--resolve-s3` option is required. If you specify both the `--s3-bucket` and `--resolve-s3` options, then an error will result.

`--s3-prefix` *TEXT*

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. This only applies for functions declared with `Zip` package type.

`--save-params`

Save the parameters that you provide at the command line to the AWS SAM configuration file.

`--signing-profiles` *LIST*

(Optional) The list of signing profiles to sign your deployment packages with. This parameter takes a list of key-value pairs, where the key is the name of the function or layer to sign, and the value is the signing profile, with an optional profile owner delimited with `:`. For example, `FunctionNameToSign=SigningProfileName1 LayerNameToSign=SigningProfileName2:SigningProfileOwner`.

`--template-file, --template, -t` *PATH*

The path and file name where your AWS SAM template is located.

> **ⓘ Note**
>
> If you specify this option, AWS SAM packages only the template and the local resources that it points to.

`--use-json`

Output JSON for the AWS CloudFormation template. YAML is used by default.

sam pipeline bootstrap

This page provides reference information for the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam local pipeline bootstrap` subcommand.

For an introduction to the AWS SAM CLI, see [What is the AWS SAM CLI?](#).

The `sam pipeline bootstrap` subcommand generates the required AWS infrastructure resources to connect to your CI/CD system. This step must be run for each deployment stage in your pipeline prior to running the **sam pipeline init** command.

This subcommand sets up the following AWS infrastructure resources:

- Option of configuring pipeline permissions through:
  - A pipeline IAM user with access key ID and secret key access credentials to be shared with the CI/CD system.

> **ⓘ Note**
>
> We recommend rotating access keys regularly. For more information, see Rotate access keys regularly for use cases that require long-term credentials in the *IAM User Guide*.

- Supported CI/CD platforms through OIDC. For an introduction on using OIDC with AWS SAM pipeline, go to Using OIDC authentication with AWS SAM pipeline.

- An AWS CloudFormation execution IAM role assumed by AWS CloudFormation to deploy the AWS SAM application.

- An Amazon S3 bucket to hold the AWS SAM artifacts.

- Optionally, an Amazon ECR image repository to hold container image Lambda deployment packages (if you have a resource that is of package type `Image`).

Usage

```
$ sam pipeline bootstrap <options>
```

Options

`--bitbucket-repo-uuid` *TEXT*

The UUID of the Bitbucket repository. This option is specific to using Bitbucket OIDC for permissions.

> **ⓘ Note**
>
> This value can be found at https://bitbucket.org/*workspace*/*repository*/admin/addon/admin/pipelines/openid-connect

`--bucket` *TEXT*

The ARN of the Amazon S3 bucket that holds the AWS SAM artifacts.

`--cicd-provider` *TEXT*

The CI/CD platform for the AWS SAM pipeline.

`--cloudformation-execution-role` *TEXT*

The ARN of the IAM role to be assumed by AWS CloudFormation while deploying the application's stack. Provide only if you want to use your own role. Otherwise, the command will create a new role.

`--config-env` *TEXT*

The environment name that specifies 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.

`--config-file` *PATH*

The path and file name of the configuration file containing the 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.

`--confirm-changeset | --no-confirm-changeset`

Prompt to confirm the deployment of your resources.

`--create-image-repository | --no-create-image-repository`

Specify whether to create an Amazon ECR image repository if none is provided. The Amazon ECR repository holds the container images of Lambda functions, or layers having a package type of `Image`. The default is `--no-create-image-repository`.

`--debug`

Turns on debug logging and prints debug messages that the AWS SAM CLI generates, and to display timestamps.

`--deployment-branch` *TEXT*

Name of the branch that deployments will occur from. This option is specific to using GitHub Actions OIDC for permissions.

`--github-org` *TEXT*

The GitHub organization that the repository belongs to. If no organization exists, enter the user name of the repository owner. This option is specific to using GitHub Actions OIDC for permissions.

`--github-repo` *TEXT*

    Name of the GitHub repository that deployments will occur from. This option is specific to using GitHub Actions OIDC for permissions.

`--gitlab-group` *TEXT*

    The GitLab group that the repository belongs to. This option is specific to using GitLab OIDC for permissions.

`--gitlab-project` *TEXT*

    The GitLab project name. This option is specific to using GitLab OIDC for permissions.

`--help, -h`

    Shows this message and exits.

`--image-repository` *TEXT*

    The ARN of an Amazon ECR image repository that holds the container images of Lambda functions, or layers that have a package type of `Image`. If provided, the `--create-image-repository` options is ignored. If not provided and `--create-image-repository` is specified, the command creates one.

`--interactive | --no-interactive`

    Disable interactive prompting for bootstrap parameters and fail if any required parameters are missing. The default value is `--interactive`. For this command, `--stage` is the only required parameter.

> ### ⓘ Note
>
>   If `--no-interactive` is specified along with `--use-oidc-provider`, all required parameters for your OIDC provider must be included.

`--oidc-client-id` *TEXT*

    The client ID configured for use with your OIDC provider.

`--oidc-provider` *[github-actions | gitlab | bitbucket-pipelines]*

    Name of the CI/CD provider that will be used for OIDC permissions. GitLab, GitHub, and Bitbucket are supported.

`--oidc-provider-url` *TEXT*

    The URL for the OIDC provider. Value must begin with **https://**.

`--permissions-provider` *[oidc | iam]*

    Choose a permissions provider to assume the pipeline execution role. The default value is **iam**.

`--pipeline-execution-role` *TEXT*

    The ARN of the IAM role to be assumed by the pipeline user to operate on this stage. Provide only if you want to use your own role. If not provided, this command will create a new role.

`--pipeline-user` *TEXT*

    The Amazon Resource Name (ARN) of the IAM user having its access key ID and secret access key shared with the CI/CD system. It is used to grant this IAM user permission to access the corresponding AWS account. If not provided, the command will create an IAM user along with the access key ID and secret access key credentials.

`--profile` *TEXT*

    The specific profile from your credential file that gets AWS credentials.

`--region` *TEXT*

    The AWS Region to deploy to. For example, `us-east-1`.

`--save-params`

    Save the parameters that you provide at the command line to the AWS SAM configuration file.

`--stage` *TEXT*

    The name of the corresponding deployment stage. It is used as a suffix for the created AWS infrastructure resources.

Troubleshooting

Error: Missing required parameter

When `--no-interactive` is specified along with `--use-oidc-provider` and any of the required parameters are not provided, this error message will be displayed along with a description of the missing parameters.

sam pipeline init

This page provides reference information for the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam local pipeline init` subcommand.

For an introduction to the AWS SAM CLI, see [What is the AWS SAM CLI?](#).

The `sam pipeline init` subcommand generates a pipeline configuration file that your CI/CD system can use to deploy serverless applications using AWS SAM.

Before using **sam pipeline init**, you must bootstrap the necessary resources for each stage in your pipeline. You can do this by running **sam pipeline init --bootstrap** to be guided through the setup and configuration file generation process, or refer to resources you have previously created with the **sam pipeline bootstrap** command.

Usage

```
$ sam pipeline init <options>
```

Options

`--bootstrap`

   Enable interactive mode that walks the user through creating necessary AWS infrastructure resources.

`--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](#).

`--config-file` *TEXT*

   The path and file name of the configuration file containing default parameter values to use. The default value is `samconfig.toml` in the project root directory. For more information about configuration files, see [AWS SAM CLI configuration file](#).

`--debug`

   Turns on debug logging to print debug messages that the AWS SAM CLI generates, and to display timestamps.

```
--help, -h
```

   Shows this message and exits.

```
--save-params
```

   Save the parameters that you provide at the command line to the AWS SAM configuration file.

sam publish

This page provides reference information for the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam publish` command.

For an introduction to the AWS SAM CLI, see [What is the AWS SAM CLI?](#).

The `sam publish` command publishes 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 AWS Region.

The `sam publish` command expects the AWS SAM template to include a `Metadata` section that contains application metadata required for publishing. In the `Metadata` section, the `LicenseUrl` and `ReadmeUrl` properties must refer to Amazon Simple Storage Service (Amazon S3) buckets, not local files. For more information about the `Metadata` section of the AWS SAM template, see [Publishing serverless applications using the AWS SAM CLI](#).

By default, `sam publish` creates the application as private. Before other AWS accounts are allowed to view and deploy your application, you must share it. For information on sharing applications, see [AWS Serverless Application Repository Resource-Based Policy Examples](#) in the *AWS Serverless Application Repository Developer Guide*.

> ⓘ **Note**
>
>   Currently `sam publish` doesn't support publishing nested applications that are specified locally. If your application contains nested applications, you must publish them separately to the AWS Serverless Application Repository before publishing your parent application.

Usage

```
$ sam publish <options>
```

Options

`--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](#).

`--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](#).

`--debug`

Turns on debug logging to print debug messages that the AWS SAM CLI generates, and to display timestamps.

`--help`

Shows this message and exits.

`--profile` *TEXT*

The specific profile from your credential file that gets AWS credentials.

`--region` *TEXT*

The AWS Region to deploy to. For example, us-east-1.

`--save-params`

Save the parameters that you provide at the command line to the AWS SAM configuration file.

`--semantic-version` *TEXT*

(Optional) Use this option to provide a semantic version of your application that overrides the `SemanticVersion` property in the `Metadata` section of the template file. For more information about semantic versioning, see the [Semantic Versioning specification](#).

`--template, -t` *PATH*

The path of AWS SAM template file [`default: template.[yaml|yml]`].

Examples

To publish an application:

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

sam remote invoke

This page provides reference information for the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam remote invoke` command.

- For an introduction to the AWS SAM CLI, see [What is the AWS SAM CLI?](#).

- For documentation on using the AWS SAM CLI `sam remote invoke` command, see [Using sam remote invoke](#).

The `sam remote invoke` command invokes supported resources in the AWS Cloud.

Usage

```
$ sam remote invoke <arguments> <options>
```

Arguments

Resource ID

The ID of the suppored resource to invoke.

This argument accepts the following values:

- **Amazon Resource Name (ARN)** – The ARN of the resource.

  > **ⓘ Tip**
  >
  > Use `sam list stack-outputs --stack-name` *<stack-name>* to obtain the ARN of your resources.

- **Logical ID** – The logical ID of the resource. You must also provide the AWS CloudFormation stack name using the `--stack-name` option.

- **Physical ID** – The physical ID of the resource. This ID gets created when you deploy a resource using AWS CloudFormation.

> **ⓘ Tip**
>
> Use `sam list resources --stack-name` `<stack-name>` to obtain the physical
> ID of your resources.

When you provide an ARN or physical ID:

If you provide an ARN or physical ID, do not provide a stack name. When the stack name
is provided using the `--stack-name` option, or when the stack name is defined in your
configuration file, the AWS SAM CLI will automatically process your resource ID as a logical
ID value from the AWS CloudFormation stack.

When you don't provide a resource ID:

If you don't provide a resource ID, but do provide a stack name with the `--stack-name`
option, the AWS SAM CLI will attempt to automatically invoke a resource in your AWS
CloudFormation stack using the following logic:

1. The AWS SAM CLI will identify resource types in the following order and move to the next
   step once the resource type is found in your stack:

   a. Lambda

   b. Step Functions

   c. Amazon SQS

   d. Kinesis Data Streams

2. If the resource type has a single resource in your stack, the AWS SAM CLI will invoke it. If
   multiple resources of the resource type exists in your stack, the AWS SAM CLI will return
   an error.

The following are examples of what the AWS SAM CLI will do:

- **Stack that contains two Lambda functions and an Amazon SQS queue** – The AWS SAM
  CLI will locate the Lambda resource type and return and error since the stack contains
  more than one Lambda function.

- **Stack that contains a Lambda function and two Amazon Kinesis Data Streams
  applications** – The AWS SAM CLI will locate the Lambda function and invoke it since the
  stack contains a single Lambda resource.

- **Stack that contains a single Amazon SQS queue and two Kinesis Data Streams applications** – The AWS SAM CLI will locate the Amazon SQS queue and invoke it since the stack contains a single Amazon SQS queue.

Options

`--beta-features | --no-beta-features`

Allow or deny beta features.

`--config-env` *TEXT*

Specify the environment to use from your AWS SAM CLI configuration file.

Default: `default`

`--config-file` *FILENAME*

Specify the path and file name of your configuration file.

For more information about configuration files, see [Configuring the AWS SAM CLI](#).

Default: `samconfig.toml` at the root of your project directory.

`--debug`

Activate debug logging. This prints debug messages and timestamps generated by the AWS SAM CLI.

`--event, -e` *TEXT*

The event to send to the target resource.

`--event-file` *FILENAME*

The path to a file that contains the event to send to the target resource.

`--help, -h`

Show the help message and exit.

`--output` *[text | json]*

Output the results of your invocation in a specific output format.

json – The request metadata and resource response are returned in JSON structure. The response contains the full SDK output.

text – The request metadata is returned in text structure. The resource response is returned in the output format of the invoked resource.

```
--parameter
```

Additional [Boto3](#) parameters that you can pass to the resource being invoked.

Amazon Kinesis Data Streams

The following additional parameters can be used to put a record in the Kinesis data stream:

- `ExplicitHashKey='`*`string`*`'`

- `PartitionKey='`*`string`*`'`

- `SequenceNumberForOrdering='`*`string`*`'`

- `StreamARN='`*`string`*`'`

For a description of each parameter, see [Kinesis.Client.put_record](#).

AWS Lambda

The following additional parameters can be used to invoke a Lambda resource and receive a buffered response:

- `ClientContext='`*`base64-encoded string`*`'`

- `InvocationType='`*`[ DryRun | Event | RequestResponse ]`*`'`

- `LogType='`*`[ None | Tail ]`*`'`

- `Qualifier='`*`string`*`'`

The following additional parameters can be used to invoke a Lambda resource with response streaming:

- `ClientContext='`*`base64-encoded string`*`'`

- `InvocationType='`*`[ DryRun | RequestResponse ]`*`'`

- `LogType='`*`[ None | Tail ]`*`'`

- `Qualifier='`*`string`*`'`

For a description of each parameter, see the following:

- Lambda with buffered response – [Lambda.Client.invoke](#)

- Lambda with response streaming – [Lambda.Client.invoke_with_response_stream](#)

Amazon Simple Queue Service (Amazon SQS)

The following additional parameters can be used to send a message to an Amazon SQS queue:

- `DelaySeconds=`*`integer`*
- `MessageAttributes='`*`json string`*`'`
- `MessageDeduplicationId='`*`string`*`'`
- `MessageGroupId='`*`string`*`'`
- `MessageSystemAttributes='`*`json string`*`'`

For a description of each parameter, see [SQS.Client.send_message](#).

AWS Step Functions

The following additional parameters can be used to start a state machine execution:

- `name='`*`string`*`'`
- `traceHeader='`*`string`*`'`

For a description of each parameter, see [SFN.Client.start_execution](#).

`--profile` *`TEXT`*

The specific profile from your credential file to get AWS credentials.

`--region` *`TEXT`*

The AWS Region of the resource. For example, `us-east-1`.

`--stack-name` *`TEXT`*

The name of the AWS CloudFormation stack that the resource belongs to.

`--test-event-name` *`NAME`*

The name of the shareable test event to pass to your Lambda function.

> ⓘ **Note**
>
> This option only supports Lambda functions.

sam remote test-event

This page provides reference information for the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam remote test-event` command.

- For an introduction to the AWS SAM CLI, see [What is the AWS SAM CLI?](#).

- For documentation on using the AWS SAM CLI `sam remote test-event` command, see [Using sam remote test-event](#).

The `sam remote test-event` command interacts with shareable test events in the Amazon EventBridge schema registry.

Usage

```
$ sam remote test-event <options> <subcommand>
```

Options

`--help, -h`

Show the help message and exit.

Subcommands

`delete`

Delete a shareable test event from the EventBridge schema registry. For more reference information, see [sam remote test-event delete](#).

`get`

Get a shareable test event from the EventBridge schema registry. For more reference information, see [sam remote test-event get](#).

`list`

List existing shareable test events for an AWS Lambda function. For more reference information, see [sam remote test-event list](#).

put

Save an event from a local file to the EventBridge schema registry. For more reference information, see sam remote test-event put.

sam remote test-event delete

This page provides reference information for the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam remote test-event delete` subcommand.

- For an introduction to the AWS SAM CLI, see What is the AWS SAM CLI?.
- For documentation on using the AWS SAM CLI `sam remote test-event` command, see Using sam remote test-event.

The `sam remote test-event delete` subcommand deletes a shareable test event from the Amazon EventBridge schema registry.

Usage

```
$ sam remote test-event delete <arguments> <options>
```

Arguments

Resource ID

The ID of the AWS Lambda function associated with the shareable test event.

If you provide a logical ID, you must also provide a value for the AWS CloudFormation stack associated with the Lambda function using the `--stack-name` option.

Valid values: The resource's logical ID or resource ARN.

Options

`--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.

```
--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](#).

```
--help, -h
```

Show the help message and exit.

```
--name TEXT
```

The name of the shareable test event to delete.

```
--stack-name TEXT
```

The name of the AWS CloudFormation stack associated with the Lambda function.

This option is required if you are providing the Lambda function logical ID as an argument.

sam remote test-event get

This page provides reference information for the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam remote test-event get` subcommand.

- For an introduction to the AWS SAM CLI, see [What is the AWS SAM CLI?](#).

- For documentation on using the AWS SAM CLI `sam remote test-event` command, see [Using sam remote test-event](#).

The `sam remote test-event get` subcommand gets a shareable test event from the Amazon EventBridge schema registry.

Usage

```
$ sam remote test-event get <arguments> <options>
```

Arguments

Resource ID

The ID of the AWS Lambda function associated with the shareable test event to get.

If you provide a logical ID, you must also provide a value for the AWS CloudFormation stack associated with the Lambda function using the `--stack-name` option.

Valid values: The resource's logical ID or resource ARN.

Options

`--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.

`--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.

`--help, -h`

Show the help message and exit.

`--name` *TEXT*

The name of the shareable test event to get.

`--output-file` *FILENAME*

The file path and name to save the event to on your local machine.

If you don't provide this option, the AWS SAM CLI will output the contents of the shareable test event to your console.

`--stack-name` *TEXT*

The name of the AWS CloudFormation stack associated with the Lambda function.

This option is required if you are providing the Lambda function logical ID as an argument.

sam remote test-event list

This page provides reference information for the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam remote test-event list` subcommand.

- For an introduction to the AWS SAM CLI, see [What is the AWS SAM CLI?](#).

- For documentation on using the AWS SAM CLI `sam remote test-event` command, see [Using sam remote test-event](#).

The `sam remote test-event list` subcommand lists the existing shareable test events for a specific AWS Lambda function from the Amazon EventBridge schema registry.

Usage

```
$ sam remote test-event list <arguments> <options>
```

Arguments

Resource ID

The ID of the Lambda function associated with the shareable test events.

If you provide a logical ID, you must also provide a value for the AWS CloudFormation stack associated with the Lambda function using the `--stack-name` option.

Valid values: The resource's logical ID or resource ARN.

Options

`--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](#).

`--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](#).

`--help, -h`

Show the help message and exit.

`--stack-name` *TEXT*

The name of the AWS CloudFormation stack associated with the Lambda function.

This option is required if you are providing the Lambda function logical ID as an argument.

sam remote test-event put

This page provides reference information for the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam remote test-event put` subcommand.

- For an introduction to the AWS SAM CLI, see [What is the AWS SAM CLI?](#).

- For documentation on using the AWS SAM CLI `sam remote test-event` command, see [Using sam remote test-event](#).

The `sam remote test-event put` subcommand saves a shareable test event from your local machine to the Amazon EventBridge schema registry.

Usage

```
$ sam remote test-event put <arguments> <options>
```

Arguments

Resource ID

The ID of the AWS Lambda function associated with the shareable test event.

If you provide a logical ID, you must also provide a value for the AWS CloudFormation stack associated with the Lambda function using the `--stack-name` option.

Valid values: The resource's logical ID or resource ARN.

Options

`--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](#).

`--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](#).

`--file` *FILENAME*

> The file path and name to the event to on your local machine.
>
> Provide - as the file name value to read from `stdin`.
>
> This option is required.

`--force, -f`

> Overwrite a shareable test event with the same name.

`--help, -h`

> Show the help message and exit.

`--name` *TEXT*

> The name to save the shareable test event as.
>
> If a shareable test event with the same name exists in the EventBridge schema registry, the AWS SAM CLI will not overwrite it. To overwrite, add the `--force` option.

`--output-file` *FILENAME*

> The file path and name to save the event to on your local machine.
>
> If you don't provide this option, the AWS SAM CLI will output the contents of the shareable test event to your console.

`--stack-name` *TEXT*

> The name of the AWS CloudFormation stack associated with the Lambda function.
>
> This option is required if you are providing the Lambda function logical ID as an argument.

sam sync

This page provides reference information for the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam sync` command.

- For an introduction to the AWS SAM CLI, see [What is the AWS SAM CLI?](#).

- For documentation on using the AWS SAM CLI, see [Using the AWS SAM CLI](#).

The `sam sync` command syncs local application changes to the AWS Cloud.

Usage

```
$ sam sync <options>
```

Options

`--base-dir, -s DIRECTORY`

Resolves relative paths to the function's or layer's source code with respect to this directory. Use this option to change how relative paths to source code folders are resolved. By default, relative paths are resolved with respect to the AWS SAM template's location.

In addition to the resources in the root application or stack that you're building, this option also applies to nested applications or stacks. Additionally, this option applies to the following resource types and properties:

- Resource type: `AWS::Serverless::Function` Property: `CodeUri`

- Resource type: `AWS::Serverless::Function` Resource attribute: `Metadata` Entry: `DockerContext`

- Resource type: `AWS::Serverless::LayerVersion` Property: `ContentUri`

- Resource type: `AWS::Lambda::Function` Property: `Code`

- Resource type: `AWS::Lambda::LayerVersion` Property: `Content`

`--build-image TEXT`

The URI for the [container image](#) that you want to use when building your application. By default, AWS SAM uses the container image repository URI from [Amazon Elastic Container Registry (Amazon ECR) Public](#). Specify this option to use a different image.

You can use this option multiple times in a single command. Each option accepts a string or a key-value pair.

- **String** – Specify the URI of the container image that all resources in your application will use. The following is an example:

```
$ sam sync --build-image amazon/aws-sam-cli-build-image-python3.8
```

- **Key-value pair** – Specify the resource name as the key and the container image URI to be used with that resource as the value. Use this format to specify a different container image URI for each resource in your application. The following is an example:

```
$ sam sync --build-image Function1=amazon/aws-sam-cli-build-image-python3.8
```

This option only applies if the `--use-container` option is specified, otherwise an error will result.

`--build-in-source | --no-build-in-source`

Provides `--build-in-source` to build your project directly in the source folder.

The `--build-in-source` option supports the following runtimes and build methods:

- **Runtimes** – Any Node.js runtime supported by the <u>`sam init --runtime`</u> option.
- **Build methods** – `Makefile`, `esbuild`.

The `--build-in-source` option is not compatible with the following options:

- `--use-container`

Default: `--no-build-in-source`

`--capabilities LIST`

A list of capabilities that you 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. Specify this option to override the default values. Valid values include the following:

- CAPABILITY_IAM
- CAPABILITY_NAMED_IAM
- CAPABILITY_RESOURCE_POLICY
- CAPABILITY_AUTO_EXPAND

Default: CAPABILITY_NAMED_IAM and CAPABILITY_AUTO_EXPAND

`--code`

By default, AWS SAM syncs all resources in your application. Specify this option to sync only code resources, which include the following:

- `AWS::Serverless::Function`

- `AWS::Lambda::Function`

- `AWS::Serverless::LayerVersion`

- `AWS::Lambda::LayerVersion`

- `AWS::Serverless::Api`

- `AWS::ApiGateway::RestApi`

- `AWS::Serverless::HttpApi`

- `AWS::ApiGatewayV2::Api`

- `AWS::Serverless::StateMachine`

- `AWS::StepFunctions::StateMachine`

To sync code resources, AWS SAM uses AWS service APIs directly, instead of deploying through AWS CloudFormation. To update your AWS CloudFormation stack, run **sam sync --watch** or **sam deploy**.

`--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](#).

`--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](#).

`--dependency-layer | --no-dependency-layer`

Specifies whether to separate dependencies of individual functions into another layer to speed up the sync process.

Default: `--dependency-layer`

`--image-repository` *TEXT*

The name of the Amazon Elastic Container Registry (Amazon ECR) repository where this command uploads your function's image. Required for functions declared with the `Image` package type.

`--image-repositories` *TEXT*

A mapping of functions to their Amazon ECR repository URI. Reference functions by their logical ID. The following is an example:

```
$ sam sync --image-repositories Function1=123456789012.dkr.ecr.us-
east-1.amazonaws.com/my-repo
```

You can specify this option multiple times in a single command.

`--kms-key-id` *TEXT*

The ID of an AWS Key Management Service (AWS KMS) key used to encrypt artifacts that are at rest in the Amazon S3 bucket. If you don't specify this option, then AWS SAM uses Amazon S3-managed encryption keys.

`--metadata`

A map of metadata to attach to all artifacts that you reference in your template.

`--notification-arns` *LIST*

A list of Amazon Simple Notification Service (Amazon SNS) topic ARNs that AWS CloudFormation associates with the stack.

`--parameter-overrides`

A string that contains AWS CloudFormation parameter overrides encoded as key-value pairs. Use the same format as the AWS Command Line Interface (AWS CLI). For example, `ParameterKey=ParameterValue InstanceType=t1.micro`.

`--resource` *TEXT*

Specifies the resource type to sync. To sync multiple resources, you can specify this option multiple times. This option is supported with the `--code` option. The value must be one of the listed resources under `--code`. For example, `--resource AWS::Serverless::Function --resource AWS::Serverless::LayerVersion`.

`--resource-id` *TEXT*

Specifies the resource ID to sync. To sync multiple resources, you can specify this option multiple times. This option is supported with the `--code` option. For example, `--resource-id Function1 --resource-id Function2`.

`--role-arn` *TEXT*

The Amazon Resource Name (ARN) of an IAM role that AWS CloudFormation assumes when applying the changeset.

`--s3-bucket` *TEXT*

The name of the Amazon Simple Storage Service (Amazon S3) bucket where this command uploads your AWS CloudFormation template. If your template is larger than 51,200 bytes, then either the `--s3-bucket` or the `--resolve-s3` option is required. If you specify both the `--s3-bucket` and `--resolve-s3` options, then an error occurs.

`--s3-prefix` *TEXT*

The prefix added to the names of the artifacts that you upload to the Amazon S3 bucket. The prefix name is a path name (folder name) for the Amazon S3 bucket. This applies only to functions declared with the `Zip` package type.

`--save-params`

Saves the parameters that you provide at the command line to the AWS SAM configuration file.

`--skip-deploy-sync | --no-skip-deploy-sync`

Specifies `--skip-deploy-sync` to skip the initial infrastructure sync if it isn't required. The AWS SAM CLI will compare your local AWS SAM template with the deployed AWS CloudFormation template and perform a deployment only if a change is detected.

Specifies `--no-skip-deploy-sync` to perform an AWS CloudFormation deployment every time `sam sync` is run.

To learn more, see [Skip the initial AWS CloudFormation deployment](#).

Default: `--skip-deploy-sync`

`--stack-name` *TEXT*

The name of the AWS CloudFormation stack for your application.

This option is required.

`--tags` *LIST*

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 that support it.

`--template-file, --template, -t` *PATH*

The path and file name where your AWS SAM template is located.

> **ⓘ Note**
>
> If you specify this option, then AWS SAM deploys only the template and the local resources that it points to.

`--use-container, -u`

If your functions depend on packages that have natively compiled dependencies, use this option to build your function inside an AWS Lambda-like Docker container.

> **ⓘ Note**
>
> Currently, this option is not compatible with `--dependency-layer`. If you use `--use-container` with `--dependency-layer`, the AWS SAM CLI informs you and continues with `--no-dependency-layer`.

`--watch`

Starts a process that watches your local application for changes and automatically syncs them to the AWS Cloud. By default, when you specify this option, AWS SAM syncs all resources in your application as you update them. With this option, AWS SAM performs an initial AWS CloudFormation deployment. Then, AWS SAM uses AWS service APIs to update code resources. AWS SAM uses AWS CloudFormation to update infrastructure resources when you update your AWS SAM template.

`--watch-exclude` *TEXT*

Excludes a file or folder from being observed for file changes. To use this option, `--watch` must also be provided.

This option receives a key-value pair:

- **Key** – The logical ID of a Lambda function in your application.

- **Value** – The associated file name or folder to exclude.

When you update any files or folders specified with the `--watch-exclude` option, the AWS SAM CLI will not initiate a sync. However, when an update to other files or folders initiates a sync, these files or folders will be included in that sync.

You can provide this option multiple times in a single command.

sam traces

This page provides reference information for the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam traces` command.

For an introduction to the AWS SAM CLI, see [What is the AWS SAM CLI?](#).

The `sam traces` command fetches AWS X-Ray traces in your AWS account in the AWS Region.

Usage

```
$ sam traces <options>
```

Options

`--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](#).

`--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](#).

`--end-time` *TEXT*

Fetches traces 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'.

`--output` *TEXT*

> Specifies the output format for logs. To print formatted logs, specify `text`. To print the logs as JSON, specify `json`.

`--save-params`

> Save the parameters that you provide at the command line to the AWS SAM configuration file.

`--start-time` *TEXT*

> Fetches traces 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'.

`--tail`

> Tails the trace output. This ignores the end time argument and continues to display traces as they become available.

`--trace-id` *TEXT*

> The unique identifier for an X-Ray trace.

Examples

Run the following command to fetch X-Ray traces by ID.

```
$ sam traces --trace-id tracing-id-1 --trace-id tracing-id-2
```

Run the following command to tail X-Ray traces as they become available.

```
$ sam traces --tail
```

sam validate

This page provides reference information for the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam validate` command.

For an introduction to the AWS SAM CLI, see [What is the AWS SAM CLI?](#).

The `sam validate` command verifies whether an AWS SAM template file is valid.

Usage

```
$ sam validate <options>
```

Options

--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.

--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.

--debug

Turns on debug logging to print debug message generated by the AWS SAM CLI and display timestamps.

--lint

Run linting validation on template through **cfn-lint**. Create a `cfnlintrc` config file to specify additional parameters. For more information, see cfn-lint in the *AWS CloudFormation GitHub repository*.

--profile *TEXT*

The specific profile from your credential file that gets AWS credentials.

--region *TEXT*

The AWS Region to deploy to. For example, us-east-1.

--save-params

Save the parameters that you provide at the command line to the AWS SAM configuration file.

--template-file, --template, -t *PATH*

The AWS SAM template file. Default value is `template.[yaml|yml]`.

If your template is in your current working directory and is named `template.[yaml|yml|json]`, this option is not required.

If you just ran **sam build**, this option is not required.

AWS SAM CLI configuration file

The AWS Serverless Application Model Command Line Interface (AWS SAM CLI) supports a project-level configuration file that you can use to configure AWS SAM CLI command parameter values.

For documentation on creating and using configuration files, see Configuring the AWS SAM CLI.

Topics

- Default configuration file settings
- Supported configuration file formats
- Specify a configuration file
- Configuration file basics
- Parameter value rules
- Configuration precedence
- Creating and modifying configuration files

Default configuration file settings

AWS SAM uses the following default configuration file settings:

- **Name** – `samconfig`.
- **Location** – At the root of your project. This is the same location as your `template.yaml` file.
- **Format** – TOML. To learn more, see TOML in the *TOML documentation*.

The following is an example project structure that includes the default configuration file name and location:

```
sam-app
### README.md
### __init__.py
```

```
### events
### hello_world
### samconfig.toml
### template.yaml
### tests
```

The following is an example `samconfig.toml` file:

```
...
version = 0.1

[default]
[default.global]
[default.global.parameters]
stack_name = "sam-app"

[default.build.parameters]
cached = true
parallel = true

[default.deploy.parameters]
capabilities = "CAPABILITY_IAM"
confirm_changeset = true
resolve_s3 = true

[default.sync.parameters]
watch = true

[default.local_start_api.parameters]
warm_containers = "EAGER"

[prod]
[prod.sync]
[prod.sync.parameters]
watch = false
```

Supported configuration file formats

TOML and [YAML|YML] formats are supported. See the following basic syntax:

TOML

```
version = 0.1
```

```
[environment]
[environment.command]
[environment.command.parameters]
option = parameter value
```

YAML

```
version: 0.1
environment:
  command:
    parameters:
      option: parameter value
```

Specify a configuration file

By default, the AWS SAM CLI looks for a configuration file in the following order:

1. **Custom configuration file** – If you use the `--config-file` option to specify a file name and location, the AWS SAM CLI looks for this file first.

2. **Default `samconfig.toml` file** – This is the default configuration file name and format, located at the root of your project. If you don't specify a custom configuration file, the AWS SAM CLI looks for this file next.

3. **`samconfig.[yaml|yml]` file** – If the `samconfig.toml` does not exist at the root of your project, the AWS SAM CLI looks for this file.

The following is an example of specifying a custom configuration file using the `--config-file` option:

```
$ sam deploy --config-file myconfig.yaml
```

Configuration file basics

Environment

An **environment** is a named identifier that contains a unique set of configuration settings. You can have multiple environments in a single AWS SAM application.

The default environment name is `default`.

Use the AWS SAM CLI `--config-env` option to specify the environment to use.

Command

The **command** is the AWS SAM CLI command to specify parameter values for.

To specify parameter values for all commands, use the `global` identifier.

When referencing an AWS SAM CLI command, replace spaces ( ) and hyphens (–) with underscores (_). See the following examples:

- `build`
- `local_invoke`
- `local_start_api`

Parameters

Parameters are specified as key-value pairs.

- The **key** is the AWS SAM CLI command option name.
- The **value** is the value to specify.

When specifying keys, use the long-form command option name and replace hyphens (–) with underscores (_). The following are examples:

- `region`
- `stack_name`
- `template_file`

Parameter value rules

TOML

- Boolean values can be `true` or `false`. For example, `confirm_changeset = true`.
- For string values, use quotation marks (`""`). For example, `region = "us-west-2"`.
- For list values, use quotation marks (`""`) and separate each value using a space ( ). For example: `capabilities = "CAPABILITY_IAM CAPABILITY_NAMED_IAM"`.

- For values that contain a list of key-value pairs, the pairs are space-delimited ( ) and the value of each pair is surrounded by encoded quotation marks (\" \"). For example, `tags = "project= \"my-application\" stage=\"production\""`.

- For parameter values that can be specified multiple times, the value is an array of arguments. For example: `image_repositories = ["my-function-1=image-repo-1", "my-function-2=image-repo-2"]`.

YAML

- Boolean values can be `true` or `false`. For example, `confirm_changeset: true`.

- For entries that contain a single string value, quotation marks ("") are optional. For example, `region: us-west-2`. This includes entries that contain multiple key-value pairs that are provided as a single string. The following is an example:

```
$ sam deploy --tags "foo=bar hello=world"
```

```
default:
  deploy:
    parameters:
      tags: foo=bar hello=world
```

- For entries that contain a list of values, or entries that can be used multiple times in a single command, specify them as a list of strings.

  The following is an example:

```
$ sam remote invoke --parameter "InvocationType=Event" --parameter "LogType=None"
```

```
default:
  remote_invoke:
    parameter:
    - InvocationType=Event
    - LogType=None
```

Configuration precedence

When configuring values, the following precedence takes place:

- Parameter values that you provide at the command line take precedence over corresponding values in the configuration file and `Parameters` section of the template file.

- If the `--parameter-overrides` option is used at the command line or in your configuration file with the `parameter_overrides` key, its values take precedence over values in the `Parameters` section of the template file.

- In your configuration file, entries provided for a specific command take precedence over global entries. In the following example, the `sam deploy` command will use the stack name `my-app-stack`.

  TOML

  ```
  [default.global.parameters]
  stack_name = "common-stack"

  [default.deploy.parameters]
  stack_name = "my-app-stack"
  ```

  YAML

  ```
  default:
    global:
      parameters:
        stack_name: common-stack
    deploy:
      parameters:
        stack_name: my-app-stack
  ```

Creating and modifying configuration files

Creating configuration files

When you create an application using `sam init`, a default `samconfig.toml` file is created. You can also manually create your configuration file.

Modifying configuration files

You can manually modify your configuration files. Also, during any AWS SAM CLI interactive flow, configured values will be displayed in brackets ([ ]). If you modify these values, the AWS SAM CLI will update your configuration file.

The following is an example interactive flow using the `sam deploy --guided` command:

```
$ sam deploy --guided

Configuring SAM deploy
======================

    Looking for config file [samconfig.toml] :  Found
    Reading default arguments  :  Success

    Setting default arguments for 'sam deploy'
    =========================================
    Stack Name [sam-app]: ENTER
    AWS Region [us-west-2]: ENTER
    #Shows you resources changes to be deployed and require a 'Y' to initiate deploy
    Confirm changes before deploy [Y/n]: n
    #SAM needs permission to be able to create roles to connect to the resources in
 your template
    Allow SAM CLI IAM role creation [Y/n]: ENTER
    #Preserves the state of previously provisioned resources when an operation fails
    Disable rollback [y/N]: ENTER
    HelloWorldFunction may not have authorization defined, Is this okay? [y/N]: y
    Save arguments to configuration file [Y/n]: ENTER
    SAM configuration file [samconfig.toml]: ENTER
    SAM configuration environment [default]: ENTER
```

When modifying your configuration file, the AWS SAM CLI handles global values as follows:

- If the parameter value exists in the `global` section of your configuration file, the AWS SAM CLI doesn't write the value to the specific command section.

- If the parameter value exists in both the `global` and specific command sections, the AWS SAM CLI deletes the specific entry in favor of the global value.

Managing AWS SAM CLI versions

Manage your AWS Serverless Application Model Command Line Interface (AWS SAM CLI) versions through upgrading, downgrading, and uninstalling. Optionally, you can download and install the AWS SAM CLI nightly build.

Topics

- [Upgrading the AWS SAM CLI](#)

- [Uninstalling the AWS SAM CLI](#)

- [Switch from using Homebrew to manage the AWS SAM CLI](#)

- [Managing the AWS SAM CLI nightly build](#)

- [Installing the AWS SAM CLI into a virtual environment using pip](#)

- [Managing the AWS SAM CLI with Homebrew](#)

- [Troubleshooting](#)

Upgrading the AWS SAM CLI

Linux

To upgrade the AWS SAM CLI on Linux, follow the installation instructions in [Installing the AWS SAM CLI](#), but add the `--update` option to the install command, as follows:

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

macOS

The AWS SAM CLI must be upgraded through the same method used to install it. We recommend that you use the package installer to install and upgrade the AWS SAM CLI.

To upgrade the AWS SAM CLI using the package installer, install the latest package version. For instructions, see [Installing the AWS SAM CLI](#).

Windows

To upgrade the AWS SAM CLI, repeat the Windows installation steps in [Installing the AWS SAM CLI](#) again.

Uninstalling the AWS SAM CLI

Linux

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

1. Locate the symlink and install paths.

    - Find the symlink using the **which** command:

    ```
    which sam
    ```

    The output shows the path where the AWS SAM binaries are located, for example:

    ```
    /usr/local/bin/sam
    ```

    - Find the directory that the symlink points to using the **ls** command:

    ```
    ls -l /usr/local/bin/sam
    ```

    In the following example, the installation directory is `/usr/local/aws-sam-cli`.

    ```
    lrwxrwxrwx 1 ec2-user ec2-user 49 Oct 22 09:49 /usr/local/bin/sam -> /usr/local/
    aws-sam-cli/current/bin/sam
    ```

2. Delete the symlink.

    ```
    sudo rm /usr/local/bin/sam
    ```

3. Delete the installation directory.

    ```
    sudo rm -rf /usr/local/aws-sam-cli
    ```

macOS

Uninstall the AWS SAM CLI through the same method that was used to install it. We recommend that you use the package installer to install the AWS SAM CLI.

If you installed the AWS SAM CLI using the package installer, follow these steps to uninstall.

To uninstall the AWS SAM CLI

1. Remove the AWS SAM CLI program by modifying and running the following:

    ```
    $ sudo rm -rf /path-to/aws-sam-cli
    ```

a. *sudo* – If your user has write permissions to where the AWS SAM CLI program is installed, **sudo** is not required. Otherwise, **sudo** is required.

b. */path-to* – Path to where you installed the AWS SAM CLI program. The default location is `/usr/local`.

2. Remove the AWS SAM CLI $PATH by modifying and running the following:

```
$ sudo rm -rf /path-to-symlink-directory/sam
```

a. *sudo* – If your user has write permissions to $PATH, **sudo** is not required. Otherwise, **sudo** is required.

b. *path-to-symlink-directory* – Your $PATH environment variable. The default location is `/usr/local/bin`.

3. Verify that the AWS SAM CLI is uninstalled by running the following:

```
$ sam --version
command not found: sam
```

Windows

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

1. From the Start menu, search for "Add or remove programs".

2. Choose the result named **AWS SAM Command Line Interface**, and then choose **Uninstall** to launch the uninstaller.

3. Confirm that you want to uninstall the AWS SAM CLI.

Switch from using Homebrew to manage the AWS SAM CLI

If you use Homebrew to install and upgrade the AWS SAM CLI, we recommend using an AWS supported method. Follow these instructions to switch to a supported method.

To switch from using Homebrew

1. Follow instructions at Uninstalling a Homebrew installed AWS SAM CLI to uninstall the Homebrew managed version.

2.   Follow instructions at [Installing the AWS SAM CLI](#) to install the AWS SAM CLI using a supported method.

Managing the AWS SAM CLI nightly build

You can download and install the AWS SAM CLI nightly build. It contains a pre-release version of the AWS SAM CLI code that may be less stable than the production version. When installed, you can use the nightly build with the `sam-nightly` command. You can install and use both the production and nightly build versions of the AWS SAM CLI at the same time.

> ⓘ **Note**
>
> The nightly build does not contain a pre-release version of the build image. Because of that, building your serverless application with the **--use-container** option uses the latest production version of the build image.

Installing the AWS SAM CLI nightly build

To install the AWS SAM CLI nightly build, follow these instructions.

Linux

You can install the nightly build version of the AWS SAM CLI on the Linux x86_64 platform using the package installer.

To install the AWS SAM CLI nightly build

1.   Download the package installer from [sam-cli-nightly](#) in the *aws-sam-cli GitHub repository*.
2.   Follow the steps for [installing the AWS SAM CLI](#) to install the nightly build package.

macOS

You can install the nightly build version of the AWS SAM CLI on macOS, using the nightly build package installer.

To install the AWS SAM CLI nightly build

1.   Download the package installer for your platform from [sam-cli-nightly](#) in the *aws-sam-cli GitHub repository*.

2. Follow the steps for [installing the AWS SAM CLI](#) to install the nightly build package.

Windows

The nightly build version of the AWS SAM CLI is available with this download link: [AWS SAM CLI nightly build](#). To install the nightly build on Windows, perform the same steps as in the [Installing the AWS SAM CLI](#), but use the nightly build download link instead.

To verify that you have installed the nightly build version, run the **sam-nightly --version** command. The output of this command is in the form `1.X.Y.dev<YYYYMMDDHHmm>`, for example:

```
SAM CLI, version 1.20.0.dev202103151200
```

Switch from Homebrew to the package installer

If you are using Homebrew to install and upgrade the AWS SAM CLI nightly build and would like to switch to using the package installer, follow these steps.

To switch from Homebrew to the package installer

1. Uninstall the Homebrew installed AWS SAM CLI nightly build.

    ```
    $ brew uninstall aws-sam-cli-nightly
    ```

2. Verify that the AWS SAM CLI nightly build is uninstalled by running the following:

    ```
    $ sam-nightly --version
    zsh: command not found: sam-nightly
    ```

3. Follow steps in the previous section to install the AWS SAM CLI nightly build.

Installing the AWS SAM CLI into a virtual environment using pip

We recommend using the native package installer to install the AWS SAM CLI. If you must use pip, we recommend that you install the AWS SAM CLI into a virtual environment. This ensures a clean installation environment and an isolated environment if errors occur.

> ⓘ **Note**
>
> As of October 24, 2023, the AWS SAM CLI is discontinuing support for Python 3.7. To learn more, see AWS SAM CLI discontinuing support for Python 3.7.

To install the AWS SAM CLI into a virtual environment

1. From a starting directory of your choice, create a virtual environment and name it.

    Linux / macOS

    ```
    $ mkdir project
    $ cd project
    $ python3 -m venv venv
    ```

    Windows

    ```
    > mkdir project
    > cd project
    > py -3 -m venv venv
    ```

2. Activate the virtual environment

    Linux / macOS

    ```
    $ . venv/bin/activate
    ```

    The prompt changes to show you that your virtual environment is active.

    ```
    (venv) $
    ```

    Windows

    ```
    > venv\Scripts\activate
    ```

    The prompt changes to show you that your virtual environment is active.

    ```
    (venv) >
    ```

3. Install the AWS SAM CLI into your virtual environment.

```
(venv) $ pip install --upgrade aws-sam-cli
```

4. Verify that the AWS SAM CLI is installed correctly.

```
(venv) $ sam --version
SAM CLI, version 1.94.0
```

5. You can use the `deactivate` command to exit the virtual environment. Whenever you start a new session, you must reactivate the environment.

Managing the AWS SAM CLI with Homebrew

> **ⓘ Note**
>
> Starting September 2023, AWS will no longer maintain the AWS managed Homebrew installer for the AWS SAM CLI (`aws/tap/aws-sam-cli`). To continue using Homebrew, you can use the community managed installer (`aws-sam-cli`). From September 2023, any Homebrew command that references aws/tap/aws-sam-cli will redirect to `aws-sam-cli`.
> We recommend that you use our supported [installation](#) and [upgrade](#) methods.

Installing the AWS SAM CLI using Homebrew

> **ⓘ Note**
>
> These instructions use the community managed AWS SAM CLI Homebrew installer. For further support, see the [homebrew-core](#) repository.

To install the AWS SAM CLI

1. Run the following:

```
$ brew install aws-sam-cli
```

2. Verify the installation:

```
$ sam --version
```

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

```
SAM CLI, version 1.94.0
```

Upgrading the AWS SAM CLI using Homebrew

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

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

Uninstalling a Homebrew installed AWS SAM CLI

If the AWS SAM CLI was installed using Homebrew, follow these steps to uninstall it.

To uninstall the AWS SAM CLI

1. Run the following:

    ```
    $ brew uninstall aws-sam-cli
    ```

2. Verify that the AWS SAM CLI is uninstalled by running the following:

    ```
    $ sam --version
    command not found: sam
    ```

Switching to the community managed Homebrew installer

If you are using the AWS managed Homebrew installer (`aws/tap/aws-sam-cli`) and prefer to continue using Homebrew, we recommend switching to the community managed Homebrew installer (`aws-sam-cli`).

To switch in a single command, run the following:

```
$ brew uninstall aws-sam-cli && brew untap aws/tap && brew cleanup aws/tap && brew
  update && brew install aws-sam-cli
```

Follow these instructions to run each command individually.

To switch to the community managed Homebrew installer

1. Uninstall the AWS managed Homebrew version of the AWS SAM CLI:

```
$ brew uninstall aws-sam-cli
```

2. Verify that the AWS SAM CLI has been uninstalled:

```
$ which sam
sam not found
```

3. Remove the AWS managed AWS SAM CLI tap:

```
$ brew untap aws/tap
```

If you receive an error like the following, add the `--force` option and try again.

```
Error: Refusing to untap aws/tap because it contains the following installed
  formulae or casks:
aws-sam-cli-nightly
```

4. Remove cached files for the AWS managed installer:

```
$ brew cleanup aws/tap
```

5. Update Homebrew and all formulae:

```
$ brew update
```

6. Install the community managed version of the AWS SAM CLI:

```
$ brew install aws-sam-cli
```

7. Verify that the AWS SAM CLI is successfully installed:

```
$ sam --version
SAM CLI, version 1.94.0
```

Troubleshooting

If you come across errors when installing or using the AWS SAM CLI, see AWS SAM CLI troubleshooting.

Verify the integrity of the AWS SAM CLI installer

When installing the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) using a package installer, you can verify its integrity before installation. This is an optional, but highly recommended step.

The two options of verification available to you are:

- Verify the package installer signature file.

- Verify the package installer hash value.

When available for your platform, we recommend verifying the signature file option. This option offers an extra layer of security since the key values are published here and managed separately from our GitHub repository.

Topics

- Verify the installer signature file

- Verify the hash value

Verify the installer signature file

Linux

arm64 - command line installer

AWS SAM uses GnuPG to sign the AWS SAM CLI .zip installer. Verification is performed in the following steps:

1. Use the primary public key to verify the signer public key.

2. Use the signer public key to verify the AWS SAM CLI package installer.

To verify the integrity of the signer public key

1. Copy the primary public key and save it to your local machine as a `.txt` file. For example,
    primary-public-key.txt.

    ```
    -----BEGIN PGP PUBLIC KEY BLOCK-----
    Version: GnuPG v2.0.22 (GNU/Linux)

    mQINBGRuSzMBEADsqiwOy78w7F4+sshaMFRIwRGNRm94p5Qey2KMZBxekFtoryVD
    D9jEOnvupx4tvhfBHz5EcUHCEOdl4MTqdBy6vVAshozgxVb9RE8JpECn5lw7XC69
    4Y7Gy1TKKQMEWtDXElkGxIFdUWvWjSnPlzfnoXwQYGeE93CUS3h5dImP22Yk1Ct6
    eGGhlcbg1X4L8EpFMj7GvcsU8f7ziVI/PyC1Xwy39Q8/I67ip5eU5ddxO/xHqrbL
    YC7+8pJPbRMej2twT2LrcpWWYAbprMtRoa6WfE0/thoo3xhHpIMHdPfAA86ZNGIN
    kRLjGUg7jnPTRW4Oin3pCc8nT4Tfc1QERkHm641gTC/jUvpmQsM6h/FUVP2i5iE/
    JHpJcMuL2Mg6zDo3x+3gTCf+Wqz3rZzxB+wQT3yryZs6efcQy7nROiRxYBxCSXX0
    2cNYzsYLb/bYaW8yqWIHD5IqKhw269gp2E5Khs60zgS3CorMb5/xHgXjUCVgcu8a
    a8ncdf9fjl3WS5p0ohetPbO2ZjWv+MaqrZOmUIgKbA4RpWZ/fU97P5BW9ylwmIDB
    sWy0cMxg8MlvSdLytPieogaM0qMg3u5qXRGBr6Wmevkty0qgnmpGGc5zPiUbtOE8
    CnFFqyxBpj5IOnG0KZGVihvn+iRxrv6GO7WWO92+Dc6m94U0EEiBR7QiOwARAQAB
    tDRBV1MgU0FNIENMSSBQcmltYXJ5IDxhd3Mtc2FtLWNsaS1wcmltYXJ5QGFtYXpv
    bi5jb20+iQI/BBMBCQApBQJkbkszAhsvBQkHhM4ABwsJCAcDAgEGFQgCCQoLBBYC
    AwECHgECF4AACgkQQv1fenOtiFqTuhAAzi5+ju5UVOWqHKevOJSO08T4QB8HcqAE
    SVO3mY6/j29knkcL8ubZP/DbpV7QpHPI2PB5qSXsiDTP3IYPbeY78zHSDjljaIK3
    njJLMScFeGPyfPpwMsuY4nzrRIgAtXShPA8N/k4ZJcafnpNqKj7QnPxiC1KaIQWm
    pOtvb8msUF3/s0UTa5Ys/lNRhVC0eGg32ogXGdojZA2kHZWdm9udLo4CDrDcrQT7
    NtDcJASapXSQL63XfAS3snEc4e1941YxcjfYZ33rel8K9juyDZfi1slWR/L3AviI
    QFIaqSHzyOtP1oinUkoVwL8ThevKD3Ag9CZflZLzNCV7yqlF8RlhEZ4zcE/3s9El
    WzCFsozb5HfE1AZonmrDh3SyOEIBMcS6vG5dWnvJrAuSYv2rX38++K5Pr/MIAfOX
    DOI1rtA+XDsHNv9lSwSy0lt+iClawZANO9IXCiN1rOYcVQlwzDFwCNWDgkwdOqS0
    gOA2f8NF9lE5nBbeEuYquoOl1Vy8+ICbgOFs9LoWZlnVh7/RyY6ssowiU9vGUnHI
    L8f9jqRspIz/Fm3JD86ntZxLVGkeZUz62FqErdohYfkFIVcv7GONTEyrz5HLlnpv
    FJ0MR0HjrMrZrnOVZnwBKhpbLocTsH+3t5It4ReYEX0f1DIOL/KRwPvjMvBVkXY5
    hblRVDQoOWc=
    =d9oG
    -----END PGP PUBLIC KEY BLOCK-----
    ```

2. Import the primary public key to your keyring.

    ```
    $ gpg --import primary-public-key.txt

    gpg: directory `/home/.../.gnupg' created
    gpg: new configuration file `/home/.../.gnupg/gpg.conf' created
    ```

```
gpg: WARNING: options in `/home/.../.gnupg/gpg.conf' are not yet active during this
 run
gpg: keyring `/home/.../.gnupg/secring.gpg' created
gpg: keyring `/home/.../.gnupg/pubring.gpg' created
gpg: /home/.../.gnupg/trustdb.gpg: trustdb created
gpg: key 73AD885A: public key "AWS SAM CLI Primary <aws-sam-cli-
primary@amazon.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
```

3. Copy the signer public key and save it to your local machine as a `.txt` file. For example,
    signer-public-key.txt.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2.0.22 (GNU/Linux)

mQINBGRtS20BEAC7GjaAwverrB1zNEu2q3EGI6HC37WzwL5dy30f4LirZOWS3piK
oKfTqPjXPrLCf1GL2mMqUSgSnpEbPNXuvWTW1CfSnnjwuH8ZqbvvUQyHJwQyYpKm
KMwb+8V0bzzQkMzDVqolYQCi5XyGpAuo3wroxXSzG6r/mIhbiq3aRnL+2lo4XOYk
r7q9bhBqbJhzjkm7N62PhPWmi/+/EGdEBakA1pReE+cKjP2UAp5L6CPShQl2fRKL
9BumitNfFHHs1JZgZSCCruiWny3XkUaXUEMfyoE9nNbfqNvuqV2KjWguZCXASgz2
ZSPF4DTVIBMfP+xrZGQSWdGU/67QdysDQW81TbFOjK9ZsRwwGC4kbg/K98IsCNHT
ril5RZbyr8pw3fw7jYjjI2ElAacRWp53iRzvutm5AruPpLfoKDQ/tKzBUYItBwlu
Z/diKgcqtw7xDlyqNyTN8xFPFqMO2I8IsZ2Pdl131htdFiZMiin1RQG9pV9p2vHS
eQVY2uKCnvnA6vFCQYKXP7p0IwReuPNzDvECUsidw8VTakTqZsANT/bU17e4KuKn
+JgbNrKOasJX37sDb/9ruysozLvy78ozYKJDLmC3yoRQ8DhEjviT4cnjORgNmvnZ
0a5AA/DJPQW4buRrXdxu+fITzBxQn2+GO/iDNCxtJaq5SYVBKjTmTWPUJwARAQAB
tDBBV1MgU0FNIENMSSBUZWFtIDxhd3Mtc2FtLWNsaS1zaWduZXJAYW1hem9uLmNv
bT6JAj8EEwEJACkFAmRtS20CGy8FCQPCZwAHCwkIBwMCAQYVCAIJCgsEFgIDAQIe
AQIXgAAKCRDHoF9D/grd+lE4D/4kJW65He2LNsbLTta7lcGfsEXCf4zgIvkytS7U
3R36zMD8IEyWJjlZ+aPkIP8/jFJrFl4pVHbU7vX85Iut1vV7m+8BgWt25mJhnoJ9
KPjXGra9mYP+Cj8zFAcjvtl3NBAPodyfcfCTWsU3umF9ArOFICcrGCzHX2SS7wX5
h9n0vYRZxk5Qj5FsgskKAQLq33CKFAMlaqZnL5gWRvTeycSIxsyus+stX+8YBPCO
J64f7+y+MPIP1+m2njlVXg1xLEMMVa08oWccOMiakgzDev3LCrPy+wdwdn7Ut7oA
pna3DNy9aYNd2lh6vUCJeJ+Yi1Bl2jYpzLcCLKrHUmln9/rRSz7Orbg8P181kfPu
G/M7CD5FwhxP3p4+0XoGwxQefrV2jqpSnbLae7xbYJiJAhbpjWDQhuNGUbPcDmqk
aH0Q3XU8AonJ8YqaQ/q3VZ3JBiH3TbBrOXsvd59cwxYyf83aJ/WLCb2P8y75zDad
lnOP713ThF5J/Afj9HjO9waFV0Z2W2ZZe4rU2OJTAiXEtM8xsFMrc7TCUacJtJGs
u4kdBmXREcVpSz65h9ImSy2ner9qktnVVCW4mZPj63IhB37YtoLAMyz3a3R2RFNk
viEX8foOTUg1FmwHoftxZ9P91QwLoTajkDrh26ueIe45sG6Uxua2AP4Vo37cFfCj
ryV8OokCHAQQAQkABgUCZG5MWAAKCRBC/V96c62IWmglD/9idU43kW8Zy8Af1j8l
Am3lI4d9ksOleeKRZqxo/SZ5rovF32DO2nw7XRXq1+EbhgJaI3QwwOi0U0pfAMVT
4b9TdxdH+n+tqzCHh3jZqmo9sw+c9WFXYJN1hU9bLzcHXS8hOTbyoE2EuXx56ds9
```

```
L/BWCcd+LIvapw0lggFfavVx/QF4C7nBKjnJ66+xxwfgVIKR7oGlqDiHMfp9ZWh5
HhEqZo/nrNhdY0h3sczEdqC2N6eIa8mgHffHZdKudDMXIXHbgdhW9pcZXDIktVf7
j9wehsWOyYXiRgR0dz7DI26AUG4JLh5FTtx9XuSBdEsI69Jd4dJuibmgtImzbZjn
7un8DJWIyqi7Ckk96Tr4oXB9mYAXaWlR4C9j5XJhMNZgkOycuY2DADnbGmSb+1kA
ju77H4ff84+vMDwUzUt2Wwb+GjzXu2g6Wh+bWhGSirYlel+6xYrI6beu1BDCFLq+
VZFE8WggjJHpwcL7CiqadfVIQaw4HY0jQFTSdwzPWhJvYjXFOhMkyCcjssbtmB+z
/otfgySyQqThrD48RWS5GuyqCA+pK3UNmEJ11c1AXMdTn2VWInR1NOJNALQ2du3y
q8t1vMsErVOJ7pkZ50F4ef17PE6DKrXX8ilwGFyVuX5ddyt/t9J5pC3sRwHWXVZx
GXwoX75FwIEHA3n5Q7rZ69Ea6Q==
=ZIO7
-----END PGP PUBLIC KEY BLOCK-----
```

4. Import the signer public key to your keyring.

```
$ gpg --import signer-public-key.txt

gpg: key FE0ADDFA: public key "AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>"
 imported
gpg: Total number processed: 1
gpg:                 imported: 1  (RSA: 1)
gpg: no ultimately trusted keys found
```

Take note of the key value from the output. For example, *FE0ADDFA*.

5. Use the key value to obtain and verify the signer public key fingerprint.

```
$ gpg --fingerprint FE0ADDFA

pub   4096R/FE0ADDFA 2023-05-23 [expires: 2025-05-22]
      Key fingerprint = 37D8 BE16 0355 2DA7 BD6A  04D8 C7A0 5F43 FE0A DDFA
uid                   AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>
```

The fingerprint should match the following:

```
37D8 BE16 0355 2DA7 BD6A  04D8 C7A0 5F43 FE0A DDFA
```

If the fingerprint string doesn't match, do not use the AWS SAM CLI installer. Escalate to the AWS SAM team by [creating an issue](#) in the *aws-sam-cli GitHub repository*.

6. Verify the signatures of the signer public key:

```
$ gpg --check-sigs FE0ADDFA
```

```
pub    4096R/FE0ADDFA 2023-05-23 [expires: 2025-05-22]
uid                    AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>
sig!3         FE0ADDFA 2023-05-23   AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>
sig!          73AD885A 2023-05-24   AWS SAM CLI Primary <aws-sam-cli-
primary@amazon.com>
```

If you see `1 signature not checked due to a missing key`, repeat the previous steps
to import the primary and signer public keys to your keyring.

You should see the key values for both the primary public key and signer public key listed.

Now that you have verified the integrity of the signer public key, you can use the signer public key
to verify the AWS SAM CLI package installer.

To verify the integrity of the AWS SAM CLI package installer

1. **Obtain the AWS SAM CLI package signature file** – Download the signature file for the AWS
   SAM CLI package installer by using the following command:

   ```
   $ wget https://github.com/aws/aws-sam-cli/releases/latest/download/aws-sam-cli-
   linux-arm64.zip.sig
   ```

2. **Verify the signature file** – Pass both the downloaded `.sig` and `.zip` files as parameters to
   the gpg command. The following is an example:

   ```
   $ gpg --verify aws-sam-cli-linux-arm64.zip.sig aws-sam-cli-linux-arm64.zip
   ```

   The output should look similar to the following:

   ```
   gpg: Signature made Tue 30 May 2023 10:03:57 AM UTC using RSA key ID FE0ADDFA
   gpg: Good signature from "AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>"
   gpg: WARNING: This key is not certified with a trusted signature!
   gpg:           There is no indication that the signature belongs to the owner.
   Primary key fingerprint: 37D8 BE16 0355 2DA7 BD6A  04D8 C7A0 5F43 FE0A DDFA
   ```

   - The `WARNING: This key is not certified with a trusted signature!`
     message can be ignored. It occurs because there isn't a chain of trust between your personal

PGP key (if you have one) and the AWS SAM CLI PGP key. For more information, see [Web of trust](#).

- If the output contains the phrase BAD `signature`, check that you performed the procedure correctly. If you continue to get this response, escalate to the AWS SAM team by [creating an issue](#) in the *aws-sam-cli GitHub repository* and avoid using the downloaded file.

The `Good signature from "AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>"` message means that the signature is verified and you can move forward with installation.

x86_64 - command line installer

AWS SAM uses [GnuPG](#) to sign the AWS SAM CLI .zip installer. Verification is performed in the following steps:

1. Use the primary public key to verify the signer public key.
2. Use the signer public key to verify the AWS SAM CLI package installer.

To verify the integrity of the signer public key

1. Copy the primary public key and save it to your local machine as a `.txt` file. For example, *primary-public-key.txt*.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2.0.22 (GNU/Linux)

mQINBGRuSzMBEADsqiwOy78w7F4+sshaMFRIwRGNRm94p5Qey2KMZBxekFtoryVD
D9jEOnvupx4tvhfBHz5EcUHCEOdl4MTqdBy6vVAshozgxVb9RE8JpECn5lw7XC69
4Y7Gy1TKKQMEWtDXElkGxIFdUWvWjSnPlzfnoXwQYGeE93CUS3h5dImP22Yk1Ct6
eGGhlcbg1X4L8EpFMj7GvcsU8f7ziVI/PyC1Xwy39Q8/I67ip5eU5ddxO/xHqrbL
YC7+8pJPbRMej2twT2LrcpWWYAbprMtRoa6WfE0/thoo3xhHpIMHdPfAA86ZNGIN
kRLjGUg7jnPTRW4Oin3pCc8nT4Tfc1QERkHm641gTC/jUvpmQsM6h/FUVP2i5iE/
JHpJcMuL2Mg6zDo3x+3gTCf+Wqz3rZzxB+wQT3yryZs6efcQy7nROiRxYBxCSXX0
2cNYzsYLb/bYaW8yqWIHD5IqKhw269gp2E5Khs60zgS3CorMb5/xHgXjUCVgcu8a
a8ncdf9fjl3WS5p0ohetPbO2ZjWv+MaqrZOmUIgKbA4RpWZ/fU97P5BW9ylwmIDB
sWy0cMxg8MlvSdLytPieogaM0qMg3u5qXRGBr6Wmevkty0qgnmpGGc5zPiUbtOE8
CnFFqyxBpj5IOnG0KZGVihvn+iRxrv6GO7WWO92+Dc6m94U0EEiBR7QiOwARAQAB
tDRBV1MgU0FNIENMSSBQcmltYXJ5IDExhd3Mtc2FtLWNsaS1wcmltYXJ5QGFtYXpv
bi5jb20+iQI/BBMBCQApBQJkbkszAhsvBQkHhM4ABwsJCAcDAgEGFQgCCQoLBBYC
```

```
AwECHgECF4AACgkQQv1fenOtiFqTuhAAzi5+ju5UVOWqHKevOJSO08T4QB8HcqAE
SVO3mY6/j29knkcL8ubZP/DbpV7QpHPI2PB5qSXsiDTP3IYPbeY78zHSDjljaIK3
njJLMScFeGPyfPpwMsuY4nzrRIgAtXShPA8N/k4ZJcafnpNqKj7QnPxiC1KaIQWm
pOtvb8msUF3/s0UTa5Ys/lNRhVC0eGg32ogXGdojZA2kHZWdm9udLo4CDrDcrQT7
NtDcJASapXSQL63XfAS3snEc4e1941YxcjfYZ33rel8K9juyDZfi1slWR/L3AviI
QFIaqSHzyOtP1oinUkoVwL8ThevKD3Ag9CZflZLzNCV7yqlF8RlhEZ4zcE/3s9El
WzCFsozb5HfE1AZonmrDh3SyOEIBMcS6vG5dWnvJrAuSYv2rX38++K5Pr/MIAfOX
DOI1rtA+XDsHNv9lSwSy0lt+iClawZANO9IXCiN1rOYcVQlwzDFwCNWDgkwdOqS0
gOA2f8NF9lE5nBbeEuYquoOl1Vy8+ICbgOFs9LoWZlnVh7/RyY6ssowiU9vGUnHI
L8f9jqRspIz/Fm3JD86ntZxLVGkeZUz62FqErdohYfkFIVcv7GONTEyrz5HLlnpv
FJ0MR0HjrMrZrnOVZnwBKhpbLocTsH+3t5It4ReYEX0f1DIOL/KRwPvjMvBVkXY5
hblRVDQoOWc=
=d9oG
-----END PGP PUBLIC KEY BLOCK-----
```

2. Import the primary public key to your keyring.

```
$ gpg --import primary-public-key.txt

gpg: directory `/home/.../.gnupg' created
gpg: new configuration file `/home/.../.gnupg/gpg.conf' created
gpg: WARNING: options in `/home/.../.gnupg/gpg.conf' are not yet active during this
 run
gpg: keyring `/home/.../.gnupg/secring.gpg' created
gpg: keyring `/home/.../.gnupg/pubring.gpg' created
gpg: /home/.../.gnupg/trustdb.gpg: trustdb created
gpg: key 73AD885A: public key "AWS SAM CLI Primary <aws-sam-cli-
primary@amazon.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
```

3. Copy the signer public key and save it to your local machine as a `.txt` file. For example,
    signer-public-key.txt.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2.0.22 (GNU/Linux)

mQINBGRtS20BEAC7GjaAwverrB1zNEu2q3EGI6HC37WzwL5dy30f4LirZOWS3piK
oKfTqPjXPrLCf1GL2mMqUSgSnpEbPNXuvWTW1CfSnnjwuH8ZqbvvUQyHJwQyYpKm
KMwb+8V0bzzQkMzDVqolYQCi5XyGpAuo3wroxXSzG6r/mIhbiq3aRnL+2lo4XOYk
r7q9bhBqbJhzjkm7N62PhPWmi/+/EGdEBakA1pReE+cKjP2UAp5L6CPShQl2fRKL
9BumitNfFHHs1JZgZSCCruiWny3XkUaXUEMfyoE9nNbfqNvuqV2KjWguZCXASgz2
ZSPF4DTVIBMfP+xrZGQSWdGU/67QdysDQW81TbFOjK9ZsRwwGC4kbg/K98IsCNHT
```

```
ril5RZbyr8pw3fw7jYjjI2ElAacRWp53iRzvutm5AruPpLfoKDQ/tKzBUYItBwlu
Z/diKgcqtw7xDlyqNyTN8xFPFqMO2I8IsZ2Pdl131htdFiZMiin1RQG9pV9p2vHS
eQVY2uKCnvnA6vFCQYKXP7p0IwReuPNzDvECUsidw8VTakTqZsANT/bU17e4KuKn
+JgbNrKOasJX37sDb/9ruysozLvy78ozYKJDLmC3yoRQ8DhEjviT4cnjORgNmvnZ
0a5AA/DJPQW4buRrXdxu+fITzBxQn2+GO/iDNCxtJaq5SYVBKjTmTWPUJwARAQAB
tDBBV1MgU0FNIENMSSBUZWFtIDxhd3Mtc2FtLWNsaS1zaWduZXJAYW1hem9uLmNv
bT6JAj8EEwEJACkFAmRtS20CGy8FCQPCZwAHCwkIBwMCAQYVCAIJCgsEFgIDAQIe
AQIXgAAKCRDHoF9D/grd+lE4D/4kJW65He2LNsbLTta7lcGfsEXCf4zgIvkytS7U
3R36zMD8IEyWJjlZ+aPkIP8/jFJrFl4pVHbU7vX85Iut1vV7m+8BgWt25mJhnoJ9
KPjXGra9mYP+Cj8zFAcjvtl3NBAPodyfcfCTWsU3umF9ArOFICcrGCzHX2SS7wX5
h9n0vYRZxk5Qj5FsgskKAQLq33CKFAMlaqZnL5gWRvTeycSIxsyus+stX+8YBPCO
J64f7+y+MPIP1+m2njlVXg1xLEMMVa08oWccOMiakgzDev3LCrPy+wdwdn7Ut7oA
pna3DNy9aYNd2lh6vUCJeJ+Yi1Bl2jYpzLcCLKrHUmln9/rRSz7Orbg8P181kfPu
G/M7CD5FwhxP3p4+0XoGwxQefrV2jqpSnbLae7xbYJiJAhbpjWDQhuNGUbPcDmqk
aH0Q3XU8AonJ8YqaQ/q3VZ3JBiH3TbBrOXsvd59cwxYyf83aJ/WLCb2P8y75zDad
lnOP713ThF5J/Afj9HjO9waFV0Z2W2ZZe4rU2OJTAiXEtM8xsFMrc7TCUacJtJGs
u4kdBmXREcVpSz65h9ImSy2ner9qktnVVCW4mZPj63IhB37YtoLAMyz3a3R2RFNk
viEX8foOTUg1FmwHoftxZ9P91QwLoTajkDrh26ueIe45sG6Uxua2AP4Vo37cFfCj
ryV8OokCHAQQAQkABgUCZG5MWAAKCRBC/V96c62IWmglD/9idU43kW8Zy8Af1j8l
Am3lI4d9ksOleeKRZqxo/SZ5rovF32DO2nw7XRXq1+EbhgJaI3QwwOi0U0pfAMVT
4b9TdxdH+n+tqzCHh3jZqmo9sw+c9WFXYJN1hU9bLzcHXS8hOTbyoE2EuXx56ds9
L/BWCcd+LIvapw0lggFfavVx/QF4C7nBKjnJ66+xxwfgVIKR7oGlqDiHMfp9ZWh5
HhEqZo/nrNhdY0h3sczEdqC2N6eIa8mgHffHZdKudDMXIXHbgdhW9pcZXDIktVf7
j9wehsWOyYXiRgR0dz7DI26AUG4JLh5FTtx9XuSBdEsI69Jd4dJuibmgtImzbZjn
7un8DJWIyqi7Ckk96Tr4oXB9mYAXaWlR4C9j5XJhMNZgkOycuY2DADnbGmSb+1kA
ju77H4ff84+vMDwUzUt2Wwb+GjzXu2g6Wh+bWhGSirYlel+6xYrI6beu1BDCFLq+
VZFE8WggjJHpwcL7CiqadfVIQaw4HY0jQFTSdwzPWhJvYjXFOhMkyCcjssbtmB+z
/otfgySyQqThrD48RWS5GuyqCA+pK3UNmEJ11c1AXMdTn2VWInR1NOJNALQ2du3y
q8t1vMsErVOJ7pkZ50F4ef17PE6DKrXX8ilwGFyVuX5ddyt/t9J5pC3sRwHWXVZx
GXwoX75FwIEHA3n5Q7rZ69Ea6Q==
=ZIO7
-----END PGP PUBLIC KEY BLOCK-----
```

4. Import the signer public key to your keyring.

```
$ gpg --import signer-public-key.txt

gpg: key FE0ADDFA: public key "AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>"
 imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
gpg: no ultimately trusted keys found
```

Take note of the key value from the output. For example, *FE0ADDFA*.

5. Use the key value to obtain and verify the signer public key fingerprint.

```
$ gpg --fingerprint FE0ADDFA

pub    4096R/FE0ADDFA 2023-05-23 [expires: 2025-05-22]
        Key fingerprint = 37D8 BE16 0355 2DA7 BD6A  04D8 C7A0 5F43 FE0A DDFA
uid                      AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>
```

The fingerprint should match the following:

```
37D8 BE16 0355 2DA7 BD6A  04D8 C7A0 5F43 FE0A DDFA
```

If the fingerprint string doesn't match, do not use the AWS SAM CLI installer. Escalate to the AWS SAM team by [creating an issue](#) in the *aws-sam-cli GitHub repository*.

6. Verify the signatures of the signer public key:

```
$ gpg --check-sigs FE0ADDFA

pub    4096R/FE0ADDFA 2023-05-23 [expires: 2025-05-22]
uid                      AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>
sig!3        FE0ADDFA 2023-05-23  AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>
sig!         73AD885A 2023-05-24  AWS SAM CLI Primary <aws-sam-cli-
primary@amazon.com>
```

If you see `1 signature not checked due to a missing key`, repeat the previous steps to import the primary and signer public keys to your keyring.

You should see the key values for both the primary public key and signer public key listed.

Now that you have verified the integrity of the signer public key, you can use the signer public key to verify the AWS SAM CLI package installer.

To verify the integrity of the AWS SAM CLI package installer

1. **Obtain the AWS SAM CLI package signature file** – Download the signature file for the AWS SAM CLI package installer by using the following command:

```
$ wget https://github.com/aws/aws-sam-cli/releases/latest/download/aws-sam-cli-
linux-x86_64.zip.sig
```

2. **Verify the signature file** – Pass both the downloaded `.sig` and `.zip` files as parameters to the gpg command. The following is an example:

```
$ gpg --verify aws-sam-cli-linux-x86_64.zip.sig aws-sam-cli-linux-x86_64.zip
```

The output should look similar to the following:

```
gpg: Signature made Tue 30 May 2023 10:03:57 AM UTC using RSA key ID FE0ADDFA
gpg: Good signature from "AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:            There is no indication that the signature belongs to the owner.
Primary key fingerprint: 37D8 BE16 0355 2DA7 BD6A  04D8 C7A0 5F43 FE0A DDFA
```

- The `WARNING: This key is not certified with a trusted signature!` message can be ignored. It occurs because there isn't a chain of trust between your personal PGP key (if you have one) and the AWS SAM CLI PGP key. For more information, see [Web of trust](#).

- If the output contains the phrase `BAD signature`, check that you performed the procedure correctly. If you continue to get this response, escalate to the AWS SAM team by [creating an issue](#) in the *aws-sam-cli GitHub repository* and avoid using the downloaded file.

The `Good signature from "AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>"` message means that the signature is verified and you can move forward with installation.

macOS

GUI and command line installer

You can verify the integrity of the AWS SAM CLI package installer signature file by using the `pkgutil` tool or manually.

To verify using pkgutil

1. Run the following command, providing the path to the downloaded installer on your local machine:

```
$ pkgutil --check-signature /path/to/aws-sam-cli-installer.pkg
```

The following is an example:

```
$ pkgutil --check-signature /Users/user/Downloads/aws-sam-cli-macos-arm64.pkg
```

2. From the output, locate the **SHA256 fingerprint** for **Developer ID Installer: AMZN Mobile LLC**. The following is an example:

```
Package "aws-sam-cli-macos-arm64.pkg":
   Status: signed by a developer certificate issued by Apple for distribution
   Notarization: trusted by the Apple notary service
   Signed with a trusted timestamp on: 2023-05-16 20:29:29 +0000
   Certificate Chain:
    1. Developer ID Installer: AMZN Mobile LLC (94KV3E626L)
       Expires: 2027-06-28 22:57:06 +0000
       SHA256 Fingerprint:
           49 68 39 4A BA 83 3B F0 CC 5E 98 3B E7 C1 72 AC 85 97 65 18 B9 4C
           BA 34 62 BF E9 23 76 98 C5 DA
       ------------------------------------------------------------------------
    2. Developer ID Certification Authority
       Expires: 2031-09-17 00:00:00 +0000
       SHA256 Fingerprint:
           F1 6C D3 C5 4C 7F 83 CE A4 BF 1A 3E 6A 08 19 C8 AA A8 E4 A1 52 8F
           D1 44 71 5F 35 06 43 D2 DF 3A
       ------------------------------------------------------------------------
    3. Apple Root CA
       Expires: 2035-02-09 21:40:36 +0000
       SHA256 Fingerprint:
           B0 B1 73 0E CB C7 FF 45 05 14 2C 49 F1 29 5E 6E DA 6B CA ED 7E 2C
           68 C5 BE 91 B5 A1 10 01 F0 24
```

3. The **Developer ID Installer: AMZN Mobile LLC SHA256 fingerprint** should match the following value:

```
49 68 39 4A BA 83 3B F0 CC 5E 98 3B E7 C1 72 AC 85 97 65 18 B9 4C BA 34 62 BF E9 23
76 98 C5 DA
```

If the fingerprint string doesn't match, do not use the AWS SAM CLI installer. Escalate to the AWS SAM team by creating an issue in the *aws-sam-cli GitHub repository*. If the fingerprint string does match, you can move forward with using the package installer.

To verify the package installer manually

- See How to verify the authenticity of manually downloaded Apple software updates at the *Apple support website*.

Windows

The AWS SAM CLI installer is packaged as MSI files for the Windows operating system.

To verify the integrity of the installer

1. Right-click on the installer and open the **Properties** window.

2. Choose the **Digital Signatures** tab.

3. From the **Signature List**, choose **Amazon Web Services, Inc.**, and then choose **Details**.

4. Choose the **General** tab, if not already selected, and then choose **View Certificate**.

5. Choose the **Details** tab, and then choose **All** in the **Show** dropdown list, if not already selected.

6. Scroll down until you see the **Thumbprint** field and then choose **Thumbprint**. This displays the entire thumbprint value in the lower window.

7. Match the thumbprint value to the following value. If the value matches, move forward with installation. If not, escalate to the AWS SAM team by creating an issue in the *aws-sam-cli GitHub repository*.

```
c011d416e99a1142c0e0235118ef64c2681f3db9
```

Verify the hash value

Linux

x86_64 - command line installer

Verify the integrity and authenticity of the downloaded installer files by generating a hash value using the following command:

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

The output should look like the following example:

```
<64-character SHA256 hash value> aws-sam-cli-linux-x86_64.zip
```

Compare the 64-character SHA-256 hash value with the one for your desired AWS SAM CLI version in the AWS SAM CLI release notes on GitHub.

macOS

GUI and command line installer

Verify the integrity and authenticity of the downloaded installer by generating a hash value using the following command:

```
$ shasum -a 256 path-to-pkg-installer/name-of-pkg-installer

# Examples
$ shasum -a 256 ~/Downloads/aws-sam-cli-macos-arm64.pkg
$ shasum -a 256 ~/Downloads/aws-sam-cli-macos-x86_64.pkg
```

Compare your 64-character SHA-256 hash value with the corresponding value in the AWS SAM CLI release notes GitHub repository.

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:

```
$ 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:

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

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

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

AWS SAM CLI troubleshooting

Troubleshoot error messages when using, installing, and managing the AWS Serverless Application Model Command Line Interface (AWS SAM CLI).

Topics

- [Installation errors](#)
- [Error messages](#)
- [Warning messages](#)

Installation errors

Linux

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

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

Shell error: "command not found"

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

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

If you receive this error, you're using an unsupported version of Linux, and the built-in glibc version is out of date. Try either of the following:

- Upgrade your Linux host to the 64-bit version of a recent distribution of CentOS, Fedora, Ubuntu, or Amazon Linux 2.

- Follow the instructions for [Installing the AWS SAM CLI](#).

macOS

The installation failed
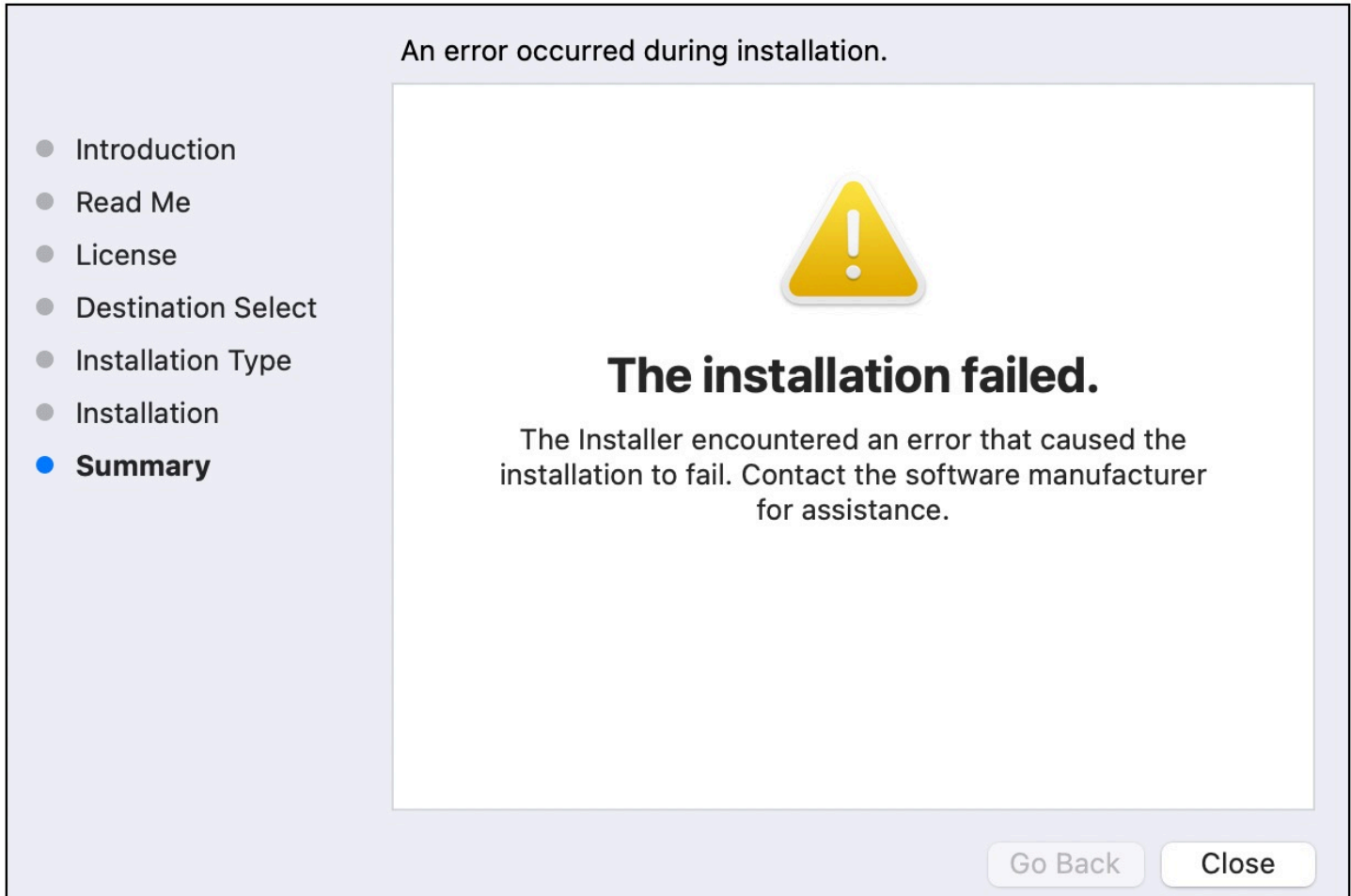
If you are installing the AWS SAM CLI for your user and selected an installation directory that you don't have write permissions for, this error could occur. Try either of the following:

1. Select a different installation directory that you have write permissions for.

2. Delete the installer. Then, download and run it again.

Error messages

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 a domain that's not valid. This can happen if your serverless application failed to deploy successfully, or if you have a typo in your **curl** command. Verify that the application deployed successfully by using the AWS CloudFormation console or the AWS CLI, and verify that your **curl** command is correct.

Error: Can't find exact resource information with given stack name

When running the `sam remote invoke` command on an application that contains a single Lambda function resource, you see the following error:

```
Error: Can't find exact resource information with given <stack-name>. Please provide
  full resource ARN or --stack-name to resolve the ambiguity.
```

Possible cause: You didn't provide the `--stack-name` option.

If a function ARN is not provided as an argument, the `sam remote invoke` command requires that the `--stack-name` option is provided.

Solution: Provide the `--stack-name` option.

The following is an example:

```
$ sam remote invoke --stack-name sam-app

Invoking Lambda Function HelloWorldFunction



START RequestId: 40593abb-e1ad-4d99-87bd-ac032e364e82 Version: $LATEST
END RequestId: 40593abb-e1ad-4d99-87bd-ac032e364e82
REPORT RequestId: 40593abb-e1ad-4d99-87bd-ac032e364e82  Duration: 11.31 ms
  Billed Duration: 12 ms  Memory Size: 128 MB     Max Memory Used: 67 MB  Init
 Duration: 171.71 ms
{"statusCode":200,"body":"{\"message\":\"hello world\"}"}%
```

Error: Can't find resource information from stack name

When running the `sam remote invoke` command and passing a Lambda function ARN as an argument, you see the following error:

```
Error: Can't find resource information from stack name (<stack-name>) and resource id
  (<function-id>)
```

Possible cause: You have the stack name value defined in your `samconfig.toml` file.

The AWS SAM CLI first checks your `samconfig.toml` file for a stack name. If specified, the argument is passed as a logical ID value.

Solution: Pass the function's logical ID instead.

You can pass the function's logical ID as an argument instead of the function's ARN.

Solution: Remove the stack name value from your configuration file.

You can remove the stack name value from your configuration file. This prevents the AWS SAM CLI from passing your function ARN as a logical ID value.

Run `sam build` after modifying your configuration file.

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

When running the **sam deploy** command, 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.

Error: FileNotFoundError in Windows

When running commands in AWS SAM CLI on Windows, you may see the following error:

```
Error: FileNotFoundError
```

Possible cause: The AWS SAM CLI might interact with filepaths that exceed the Windows max path limitation.

Solution: To resolve this issue, the new long paths behavior must enabled. To do this, see Enable Long Paths in Windows 10, Version 1607, and Later in the *Microsoft Windows App Development Documentation*.

Error: pip's dependency resolver …

Example error text:

```
ERROR: pip's dependency resolver does not currently take into account all the packages
 that are installed. This behaviour is the source of the following dependency
 conflicts.
aws-sam-cli 1.58.0 requires aws-sam-translator==1.51.0, but you have aws-sam-translator
 1.58.0 which is incompatible.
aws-sam-cli 1.58.0 requires typing-extensions==3.10.0.0, but you have typing-extensions
 4.4.0 which is incompatible.
```

Possible cause: If you use pip to install packages, dependencies between packages may conflict.

Each version of the `aws-sam-cli` package depends on a version of the `aws-sam-translator` package. For example, `aws-sam-cli` v1.58.0 may depend on `aws-sam-translator` v1.51.0.

If you install the AWS SAM CLI using pip, then install another package which depends on a newer version of `aws-sam-translator`, the following will occur:

- The newer version of `aws-sam-translator` will install.

- The current version of `aws-sam-cli` and the newer version of `aws-sam-translator` may not be compatible.

- When you use the AWS SAM CLI, the dependency resolver error will occur.

Solutions:

1. Use the AWS SAM CLI native package installer.

   a. Uninstall the AWS SAM CLI using pip. For instructions, see Uninstalling the AWS SAM CLI.

   b. Install the AWS SAM CLI using the native package installer. For instructions, see Installing the AWS SAM CLI.

   c. When necessary, upgrade the AWS SAM CLI using the native package installer. For instructions, see Upgrading the AWS SAM CLI.

2. If you must use pip, we recommend that you install the AWS SAM CLI into a virtual environment. This ensures a clean installation environment and an isolated environment if errors occur. For instructions, see Installing the AWS SAM CLI into a virtual environment using pip.

Error: No such command 'remote'

When running the `sam remote invoke` command, you see the following error:

```
$ sam remote invoke ...
2023-06-20 08:15:07 Command remote not available
Usage: sam [OPTIONS] COMMAND [ARGS]...
Try 'sam -h' for help.

Error: No such command 'remote'.
```

Possible cause: Your version of the AWS SAM CLI is out of date.

The AWS SAM CLI `sam remote invoke` command was released with AWS SAM CLI version 1.88.0. You can check your version by running the `sam --version` command.

Solution: Upgrade your AWS SAM CLI to the latest version.

For instructions, see Upgrading the AWS SAM CLI.

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

When running the **sam local start-api** command, 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 more information, see Installing Docker.

Error: Security Constraints Not Satisfied

When running **sam deploy --guided**, you're prompted with the question `Function 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 might have a publicly accessible Amazon API Gateway API configured without authorization. By responding **N** to this prompt, you're saying that this is not OK.

To fix this, you have the following options:

- Configure your application with authorization. For information about configuring authorization, see Controlling access to API Gateway APIs.

- If your intention is to have a publicly accessible API endpoint without authorization, restart your deployment and respond to this question with **Y** to indicate that you're OK with deploying.

message: 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.

Warning messages

Warning: ... AWS will no longer maintain the Homebrew installer for AWS SAM ...

When installing the AWS SAM CLI using Homebrew, you see the following warning message:

```
Warning: ... AWS will no longer maintain the Homebrew installer for AWS SAM (aws/tap/
aws-sam-cli).
    For AWS supported installations, use the first party installers ...
```

Potential cause: AWS no longer maintaining Homebrew support.

Starting September 2023, AWS will no longer maintain the Homebrew installer for the AWS SAM CLI.

Solution: Use an AWS supported installation method.

- You can find AWS supported installation methods at Installing the AWS SAM CLI.

Solution: To continue using Homebrew, use the community managed installer.

- You can use the community managed Homebrew installer at your discretion. For instructions, see Managing the AWS SAM CLI with Homebrew.

AWS SAM connector reference

This section contains reference information for the AWS Serverless Application Model (AWS SAM) connector resource type. For an introduction to connectors, see [Managing resource permissions](#) [with AWS SAM connectors](#).

Supported source and destination resource types for connectors

The `AWS::Serverless::Connector` resource type supports a select number of connections between source and destination resources. When configuring connectors in your AWS SAM template, use the following table to reference supported connections and the properties that need to be defined for each source and destination resource type. For more information about configuring connectors in your template, see [AWS::Serverless::Connector](#).

For both source and destination resources, when defined within the same template, use the `Id` property. Optionally, a `Qualifier` can be added to narrow the scope of your defined resource. When the resource is not within the same template, use a combination of supported properties.

To request new connections, [submit a new issue](#) at the *serverless-application-model AWS GitHub repository*.

Source type	Destination type	Permissions	Source properties	Destination properties
`AWS::ApiGateway::RestApi`	`AWS::Lambda::Function`	`Write`	`Id` or `Qualifier`, `ResourceId`, and `Type`	`Id` or `Arn` and `Type`
`AWS::ApiGateway::RestApi`	`AWS::Serverless::Function`	`Write`	`Id` or `Qualifier`, `ResourceId`, and `Type`	`Id` or `Arn` and `Type`
`AWS::ApiGatewayV2::Api`	`AWS::Lambda::Function`	`Write`	`Id` or `Qualifier`, `ResourceId`, and `Type`	`Id` or `Arn` and `Type`

Source type	Destination type	Permissions	Source properties	Destination properties
`AWS::ApiGatewayV2::Api`	`AWS::Serverless::Function`	`Write`	`Id` or `Qualifier`, `ResourceId`, and `Type`	`Id` or `Arn` and `Type`
`AWS::AppSync::DataSource`	`AWS::DynamoDB::Table`	`Read`	`Id` or `RoleName` and `Type`	`Id` or `Arn` and `Type`
`AWS::AppSync::DataSource`	`AWS::DynamoDB::Table`	`Write`	`Id` or `RoleName` and `Type`	`Id` or `Arn` and `Type`
`AWS::AppSync::DataSource`	`AWS::Events::EventBus`	`Write`	`Id` or `RoleName` and `Type`	`Id` or `Arn` and `Type`
`AWS::AppSync::DataSource`	`AWS::Lambda::Function`	`Write`	`Id` or `RoleName` and `Type`	`Id` or `Arn` and `Type`
`AWS::AppSync::DataSource`	`AWS::Serverless::Function`	`Write`	`Id` or `RoleName` and `Type`	`Id` or `Arn` and `Type`
`AWS::AppSync::DataSource`	`AWS::Serverless::SimpleTable`	`Read`	`Id` or `RoleName` and `Type`	`Id` or `Arn` and `Type`
`AWS::AppSync::DataSource`	`AWS::Serverless::SimpleTable`	`Write`	`Id` or `RoleName` and `Type`	`Id` or `Arn` and `Type`

Source type	Destination type	Permissions	Source properties	Destination properties
`AWS::AppSync::GraphQLApi`	`AWS::Lambda::Function`	`Write`	`Id` or `ResourceId` and `Type`	`Id` or `Arn` and `Type`
`AWS::AppSync::GraphQLApi`	`AWS::Serverless::Function`	`Write`	`Id` or `ResourceId` and `Type`	`Id` or `Arn` and `Type`
`AWS::DynamoDB::Table`	`AWS::Lambda::Function`	`Read`	`Id` or `Arn` and `Type`	`Id` or `RoleName` and `Type`
`AWS::DynamoDB::Table`	`AWS::Serverless::Function`	`Read`	`Id` or `Arn` and `Type`	`Id` or `RoleName` and `Type`
`AWS::Events::Rule`	`AWS::Events::EventBus`	`Write`	`Id` or `RoleName` and `Type`	`Id` or `Arn` and `Type`
`AWS::Events::Rule`	`AWS::Lambda::Function`	`Write`	`Id` or `Arn` and `Type`	`Id` or `Arn` and `Type`
`AWS::Events::Rule`	`AWS::Serverless::Function`	`Write`	`Id` or `Arn` and `Type`	`Id` or `Arn` and `Type`
`AWS::Events::Rule`	`AWS::Serverless::StateMachine`	`Write`	`Id` or `RoleName` and `Type`	`Id` or `Arn` and `Type`
`AWS::Events::Rule`	`AWS::SNS::Topic`	`Write`	`Id` or `Arn` and `Type`	`Id` or `Arn` and `Type`

Source type	Destination type	Permissions	Source properties	Destination properties
`AWS::Events::Rule`	`AWS::SQS::Queue`	`Write`	`Id` or `Arn` and `Type`	`Id` or `Arn`, `QueueUrl`, and `Type`
`AWS::Events::Rule`	`AWS::StepFunctions::StateMachine`	`Write`	`Id` or `RoleName` and `Type`	`Id` or `Arn` and `Type`
`AWS::Lambda::Function`	`AWS::DynamoDB::Table`	`Read`, `Write`	`Id` or `RoleName` and `Type`	`Id` or `Arn` and `Type`
`AWS::Lambda::Function`	`AWS::Events::EventBus`	`Write`	`Id` or `RoleName` and `Type`	`Id` or `Arn` and `Type`
`AWS::Lambda::Function`	`AWS::Lambda::Function`	`Write`	`Id` or `RoleName` and `Type`	`Id` or `Arn` and `Type`
`AWS::Lambda::Function`	`AWS::Location::PlaceIndex`	`Read`	`Id` or `RoleName` and `Type`	`Id` or `Arn` and `Type`
`AWS::Lambda::Function`	`AWS::S3::Bucket`	`Read`, `Write`	`Id` or `RoleName` and `Type`	`Id` or `Arn` and `Type`
`AWS::Lambda::Function`	`AWS::Serverless::Function`	`Write`	`Id` or `RoleName` and `Type`	`Id` or `Arn` and `Type`

Source type	Destination type	Permissions	Source properties	Destination properties
`AWS::Lambda::Function`	`AWS::Serverless::SimpleTable`	Read, Write	Id or RoleName and Type	Id or Arn and Type
`AWS::Lambda::Function`	`AWS::Serverless::StateMachine`	Read, Write	Id or RoleName and Type	Id or Arn, Name, and Type
`AWS::Lambda::Function`	`AWS::SNS::Topic`	Write	Id or RoleName and Type	Id or Arn and Type
`AWS::Lambda::Function`	`AWS::SQS::Queue`	Read, Write	Id or RoleName and Type	Id or Arn and Type
`AWS::Lambda::Function`	`AWS::StepFunctions::StateMachine`	Read, Write	Id or RoleName and Type	Id or Arn, Name, and Type
`AWS::S3::Bucket`	`AWS::Lambda::Function`	Write	Id or Arn and Type	Id or Arn and Type
`AWS::S3::Bucket`	`AWS::Serverless::Function`	Write	Id or Arn and Type	Id or Arn and Type
`AWS::Serverless::Api`	`AWS::Lambda::Function`	Write	Id or Qualifier , ResourceId , and Type	Id or Arn and Type

Source type	Destination type	Permissions	Source properties	Destination properties
`AWS::Serverless::Api`	`AWS::Serverless::Function`	`Write`	`Id` or `Qualifier`, `ResourceId`, and `Type`	`Id` or `Arn` and `Type`
`AWS::Serverless::Function`	`AWS::DynamoDB::Table`	`Read, Write`	`Id` or `RoleName` and `Type`	`Id` or `Arn` and `Type`
`AWS::Serverless::Function`	`AWS::Events::EventBus`	`Write`	`Id` or `RoleName` and `Type`	`Id` or `Arn` and `Type`
`AWS::Serverless::Function`	`AWS::Lambda::Function`	`Write`	`Id` or `RoleName` and `Type`	`Id` or `Arn` and `Type`
`AWS::Serverless::Function`	`AWS::S3::Bucket`	`Read, Write`	`Id` or `RoleName` and `Type`	`Id` or `Arn` and `Type`
`AWS::Serverless::Function`	`AWS::Serverless::Function`	`Write`	`Id` or `RoleName` and `Type`	`Id` or `Arn` and `Type`
`AWS::Serverless::Function`	`AWS::Serverless::SimpleTable`	`Read, Write`	`Id` or `RoleName` and `Type`	`Id` or `Arn` and `Type`
`AWS::Serverless::Function`	`AWS::Serverless::StateMachine`	`Read, Write`	`Id` or `RoleName` and `Type`	`Id` or `Arn`, `Name`, and `Type`

Source type	Destination type	Permissions	Source properties	Destination properties
`AWS::Serverless::Function`	`AWS::SNS::Topic`	`Write`	Id or RoleName and Type	Id or Arn and Type
`AWS::Serverless::Function`	`AWS::SQS::Queue`	`Read, Write`	Id or RoleName and Type	Id or Arn and Type
`AWS::Serverless::Function`	`AWS::StepFunctions::StateMachine`	`Read, Write`	Id or RoleName and Type	Id or Arn, Name, and Type
`AWS::Serverless::HttpApi`	`AWS::Lambda::Function`	`Write`	Id or Qualifier , ResourceId , and Type	Id or Arn and Type
`AWS::Serverless::HttpApi`	`AWS::Serverless::Function`	`Write`	Id or Qualifier , ResourceId , and Type	Id or Arn and Type
`AWS::Serverless::SimpleTable`	`AWS::Lambda::Function`	`Read`	Id or Arn and Type	Id or RoleName and Type
`AWS::Serverless::SimpleTable`	`AWS::Serverless::Function`	`Read`	Id or Arn and Type	Id or RoleName and Type
`AWS::Serverless::StateMachine`	`AWS::DynamoDB::Table`	`Read, Write`	Id or RoleName and Type	Id or Arn and Type

Source type	Destination type	Permissions	Source properties	Destination properties
`AWS::Serverless::StateMachine`	`AWS::Events::EventBus`	`Write`	Id or RoleName and Type	Id or Arn and Type
`AWS::Serverless::StateMachine`	`AWS::Lambda::Function`	`Write`	Id or RoleName and Type	Id or Arn and Type
`AWS::Serverless::StateMachine`	`AWS::S3::Bucket`	`Read, Write`	Id or RoleName and Type	Id or Arn and Type
`AWS::Serverless::StateMachine`	`AWS::Serverless::Function`	`Write`	Id or RoleName and Type	Id or Arn and Type
`AWS::Serverless::StateMachine`	`AWS::Serverless::SimpleTable`	`Read, Write`	Id or RoleName and Type	Id or Arn and Type
`AWS::Serverless::StateMachine`	`AWS::Serverless::StateMachine`	`Read, Write`	Id or RoleName and Type	Id or Arn, Name, and Type
`AWS::Serverless::StateMachine`	`AWS::SNS::Topic`	`Write`	Id or RoleName and Type	Id or Arn and Type

Source type	Destination type	Permissions	Source properties	Destination properties
`AWS::Serverless::StateMachine`	`AWS::SQS::Queue`	`Write`	Id or RoleName and Type	Id or Arn and Type
`AWS::Serverless::StateMachine`	`AWS::StepFunctions::StateMachine`	`Read, Write`	Id or RoleName and Type	Id or Arn, Name, and Type
`AWS::SNS::Topic`	`AWS::Lambda::Function`	`Write`	Id or Arn and Type	Id or Arn and Type
`AWS::SNS::Topic`	`AWS::Serverless::Function`	`Write`	Id or Arn and Type	Id or Arn and Type
`AWS::SNS::Topic`	`AWS::SQS::Queue`	`Write`	Id or Arn and Type	Id or Arn, QueueUrl, and Type
`AWS::SQS::Queue`	`AWS::Lambda::Function`	`Read, Write`	Id or Arn and Type	Id or RoleName and Type
`AWS::SQS::Queue`	`AWS::Serverless::Function`	`Read, Write`	Id or Arn and Type	Id or RoleName and Type
`AWS::StepFunctions::StateMachine`	`AWS::DynamoDB::Table`	`Read, Write`	Id or RoleName and Type	Id or Arn and Type

Source type	Destination type	Permissions	Source properties	Destination properties
`AWS::Step Functions ::StateMa chine`	`AWS::Even ts::Event Bus`	`Write`	`Id` or `RoleName` and `Type`	`Id` or `Arn` and `Type`
`AWS::Step Functions ::StateMa chine`	`AWS::Lamb da::Funct ion`	`Write`	`Id` or `RoleName` and `Type`	`Id` or `Arn` and `Type`
`AWS::Step Functions ::StateMa chine`	`AWS::S3:: Bucket`	`Read, Write`	`Id` or `RoleName` and `Type`	`Id` or `Arn` and `Type`
`AWS::Step Functions ::StateMa chine`	`AWS::Serv erless::F unction`	`Write`	`Id` or `RoleName` and `Type`	`Id` or `Arn` and `Type`
`AWS::Step Functions ::StateMa chine`	`AWS::Serv erless::S impleTable`	`Read, Write`	`Id` or `RoleName` and `Type`	`Id` or `Arn` and `Type`
`AWS::Step Functions ::StateMa chine`	`AWS::Serv erless::S tateMachi ne`	`Read, Write`	`Id` or `RoleName` and `Type`	`Id` or `Arn`, `Name`, and `Type`
`AWS::Step Functions ::StateMa chine`	`AWS::SNS: :Topic`	`Write`	`Id` or `RoleName` and `Type`	`Id` or `Arn` and `Type`

Source type	Destination type	Permissions	Source properties	Destination properties
`AWS::Step Functions ::StateMa chine`	`AWS::SQS: :Queue`	`Write`	`Id` or `RoleName` and `Type`	`Id` or `Arn` and `Type`
`AWS::Step Functions ::StateMa chine`	`AWS::Step Functions ::StateMa chine`	`Read,Write`	`Id` or `RoleName` and `Type`	`Id` or `Arn`, `Name`, and `Type`

IAM policies created by connectors

This section documents the AWS Identity and Access Management (IAM) policies that are created by AWS SAM when using connectors.

`AWS::DynamoDB::Table` to `AWS::Lambda::Function`

Policy type

[Customer managed policy](#) attached to the `AWS::Lambda::Function` role.

Access categories

`Read`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:DescribeStream",
        "dynamodb:GetRecords",
        "dynamodb:GetShardIterator",
        "dynamodb:ListStreams"
      ],
      "Resource": [
        "%{Source.Arn}/stream/*"
```

```
      ]
    }
  ]
}
```

AWS::Events::Rule to AWS::SNS::Topic

Policy type

[AWS::SNS::TopicPolicy](#) attached to the AWS::SNS::Topic.

Access categories

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "events.amazonaws.com"
      },
      "Resource": "%{Destination.Arn}",
      "Action": "sns:Publish",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "%{Source.Arn}"
        }
      }
    }
  ]
}
```

AWS::Events::Rule to AWS::Events::EventBus

Policy type

[Customer managed policy](#) attached to the AWS::Events::Rule role.

Access categories

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:PutEvents"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

AWS::Events::Rule to AWS::StepFunctions::StateMachine

Policy type

[Customer managed policy](#) attached to the AWS::Events::Rule role.

Access categories

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:StartExecution"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

AWS::Events::Rule to AWS::Lambda::Function

Policy type

[AWS::Lambda::Permission](#) attached to the AWS::Lambda::Function.

Access categories

Write

```
{
  "Action": "lambda:InvokeFunction",
  "Principal": "events.amazonaws.com",
  "SourceArn": "%{Source.Arn}"
}
```

AWS::Events::Rule to AWS::SQS::Queue

Policy type

AWS::SQS::QueuePolicy attached to the AWS::SQS::Queue.

Access categories

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "events.amazonaws.com"
      },
      "Resource": "%{Destination.Arn}",
      "Action": "sqs:SendMessage",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "%{Source.Arn}"
        }
      }
    }
  ]
}
```

AWS::Lambda::Function to AWS::Lambda::Function

Policy type

Customer managed policy attached to the AWS::Lambda::Function role.

Access categories

```
Write
```

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeAsync",
        "lambda:InvokeFunction"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

`AWS::Lambda::Function` to `AWS::S3::Bucket`

Policy type

[Customer managed policy](#) attached to the `AWS::Lambda::Function` role.

Access categories

```
Read
```

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectAcl",
        "s3:GetObjectLegalHold",
        "s3:GetObjectRetention",
        "s3:GetObjectTorrent",
        "s3:GetObjectVersion",
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionForReplication",
        "s3:GetObjectVersionTorrent",
```

```
        "s3:ListBucket",
        "s3:ListBucketMultipartUploads",
        "s3:ListBucketVersions",
        "s3:ListMultipartUploadParts"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}/*"
      ]
    }
  ]
}
```

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:AbortMultipartUpload",
        "s3:DeleteObject",
        "s3:DeleteObjectVersion",
        "s3:PutObject",
        "s3:PutObjectLegalHold",
        "s3:PutObjectRetention",
        "s3:RestoreObject"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}/*"
      ]
    }
  ]
}
```

AWS::Lambda::Function to AWS::DynamoDB::Table

Policy type

[Customer managed policy](#) attached to the AWS::Lambda::Function role.

Access categories

Read

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:Query",
        "dynamodb:Scan",
        "dynamodb:BatchGetItem",
        "dynamodb:ConditionCheckItem",
        "dynamodb:PartiQLSelect"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}/index/*"
      ]
    }
  ]
}
```

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:PutItem",
        "dynamodb:UpdateItem",
        "dynamodb:DeleteItem",
        "dynamodb:BatchWriteItem",
        "dynamodb:PartiQLDelete",
        "dynamodb:PartiQLInsert",
        "dynamodb:PartiQLUpdate"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}/index/*"
      ]
    }
  ]
```

```
  }
```

AWS::Lambda::Function to AWS::SQS::Queue

Policy type

[Customer managed policy](#) attached to the AWS::Lambda::Function role.

Access categories

Read

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sqs:ReceiveMessage",
        "sqs:GetQueueAttributes"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sqs:DeleteMessage",
        "sqs:SendMessage",
        "sqs:ChangeMessageVisibility",
        "sqs:PurgeQueue"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
```

```
    ]
}
```

AWS::Lambda::Function to AWS::SNS::Topic

Policy type

[Customer managed policy](#) attached to the `AWS::Lambda::Function` role.

Access categories

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

AWS::Lambda::Function to AWS::StepFunctions::StateMachine

Policy type

[Customer managed policy](#) attached to the `AWS::Lambda::Function` role.

Access categories

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:StartExecution",
```

```
        "states:StartSyncExecution"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "states:StopExecution"
      ],
      "Resource": [
        "arn:${AWS::Partition}:states:${AWS::Region}:${AWS::AccountId}:execution:
%{Destination.Name}:*"
      ]
    }
  ]
}
```

Read

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:DescribeStateMachine",
        "states:ListExecutions"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "states:DescribeExecution",
        "states:DescribeStateMachineForExecution",
        "states:GetExecutionHistory"
      ],
      "Resource": [
        "arn:${AWS::Partition}:states:${AWS::Region}:${AWS::AccountId}:execution:
%{Destination.Name}:*"
```

```
            ]
        }
    ]
}
```

AWS::Lambda::Function to AWS::Events::EventBus

Policy type

[Customer managed policy](#) attached to the AWS::Lambda::Function role.

Access categories

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:PutEvents"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

AWS::Lambda::Function to AWS::Location::PlaceIndex

Policy type

[Customer managed policy](#) attached to the AWS::Lambda::Function role.

Access categories

Read

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

```
      "Effect": "Allow",
      "Action": [
        "geo:DescribePlaceIndex",
        "geo:GetPlace",
        "geo:SearchPlaceIndexForPosition",
        "geo:SearchPlaceIndexForSuggestions",
        "geo:SearchPlaceIndexForText"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

AWS::ApiGatewayV2::Api to AWS::Lambda::Function

Policy type

AWS::Lambda::Permission attached to the AWS::Lambda::Function.

Access categories

Write

```
{
  "Action": "lambda:InvokeFunction",
  "Principal": "apigateway.amazonaws.com",
  "SourceArn": "arn:${AWS::Partition}:execute-api:${AWS::Region}:${AWS::AccountId}:
%{Source.ResourceId}/%{Source.Qualifier}"
}
```

AWS::ApiGateway::RestApi to AWS::Lambda::Function

Policy type

AWS::Lambda::Permission attached to the AWS::Lambda::Function.

Access categories

Write

```
{
```

```
    "Action": "lambda:InvokeFunction",
    "Principal": "apigateway.amazonaws.com",
    "SourceArn": "arn:${AWS::Partition}:execute-api:${AWS::Region}:${AWS::AccountId}:
%{Source.ResourceId}/%{Source.Qualifier}"
  }
```

AWS::SNS::Topic to AWS::SQS::Queue

Policy type

AWS::SQS::QueuePolicy attached to the AWS::SQS::Queue.

Access categories

Write

```json
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.amazonaws.com"
      },
      "Resource": "%{Destination.Arn}",
      "Action": "sqs:SendMessage",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "%{Source.Arn}"
        }
      }
    }
  ]
}
```

AWS::SNS::Topic to AWS::Lambda::Function

Policy type

AWS::Lambda::Permission attached to the AWS::Lambda::Function.

Access categories

Write

```
{
  "Action": "lambda:InvokeFunction",
  "Principal": "sns.amazonaws.com",
  "SourceArn": "%{Source.Arn}"
}
```

AWS::SQS::Queue to AWS::Lambda::Function

Policy type

[Customer managed policy](#) attached to the AWS::Lambda::Function role.

Access categories

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sqs:DeleteMessage"
      ],
      "Resource": [
        "%{Source.Arn}"
      ]
    }
  ]
}
```

Read

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sqs:ReceiveMessage",
        "sqs:GetQueueAttributes"
      ],
      "Resource": [
        "%{Source.Arn}"
```

```
        ]
      }
    ]
}
```

AWS::S3::Bucket to AWS::Lambda::Function

Policy type

[AWS::Lambda::Permission](#) attached to the `AWS::Lambda::Function`.

Access categories

Write

```
{
  "Action": "lambda:InvokeFunction",
  "Principal": "s3.amazonaws.com",
  "SourceArn": "%{Source.Arn}",
  "SourceAccount": "${AWS::AccountId}"
}
```

AWS::StepFunctions::StateMachine to AWS::Lambda::Function

Policy type

[Customer managed policy](#) attached to the `AWS::StepFunctions::StateMachine` role.

Access categories

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeAsync",
        "lambda:InvokeFunction"
      ],
      "Resource": [
        "%{Destination.Arn}"
```

```
        ]
      }
    ]
  }
```

AWS::StepFunctions::StateMachine to AWS::SNS::Topic

Policy type

[Customer managed policy](#) attached to the AWS::StepFunctions::StateMachine role.

Access categories

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

AWS::StepFunctions::StateMachine to AWS::SQS::Queue

Policy type

[Customer managed policy](#) attached to the AWS::StepFunctions::StateMachine role.

Access categories

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Action": [
        "sqs:SendMessage"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

AWS::StepFunctions::StateMachine to AWS::S3::Bucket

Policy type

[Customer managed policy](#) attached to the AWS::StepFunctions::StateMachine role.

Access categories

Read

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectAcl",
        "s3:GetObjectLegalHold",
        "s3:GetObjectRetention",
        "s3:GetObjectTorrent",
        "s3:GetObjectVersion",
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionForReplication",
        "s3:GetObjectVersionTorrent",
        "s3:ListBucket",
        "s3:ListBucketMultipartUploads",
        "s3:ListBucketVersions",
        "s3:ListMultipartUploadParts"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}/*"
      ]
    }
```

```
    ]
}
```

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:AbortMultipartUpload",
        "s3:DeleteObject",
        "s3:DeleteObjectVersion",
        "s3:PutObject",
        "s3:PutObjectLegalHold",
        "s3:PutObjectRetention",
        "s3:RestoreObject"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}/*"
      ]
    }
  ]
}
```

AWS::StepFunctions::StateMachine to AWS::DynamoDB::Table

Policy type

[Customer managed policy](#) attached to the AWS::StepFunctions::StateMachine role.

Access categories

Read

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:Query",
```

```
        "dynamodb:Scan",
        "dynamodb:BatchGetItem",
        "dynamodb:ConditionCheckItem",
        "dynamodb:PartiQLSelect"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}/index/*"
      ]
    }
  ]
}
```

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:PutItem",
        "dynamodb:UpdateItem",
        "dynamodb:DeleteItem",
        "dynamodb:BatchWriteItem",
        "dynamodb:PartiQLDelete",
        "dynamodb:PartiQLInsert",
        "dynamodb:PartiQLUpdate"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}/index/*"
      ]
    }
  ]
}
```

AWS::StepFunctions::StateMachine to AWS::StepFunctions::StateMachine

Policy type

[Customer managed policy](#) attached to the AWS::StepFunctions::StateMachine role.

Access categories

Read

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:DescribeExecution"
      ],
      "Resource": [
        "arn:${AWS::Partition}:states:${AWS::Region}:${AWS::AccountId}:execution:
%{Destination.Name}:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "events:DescribeRule"
      ],
      "Resource": [
        "arn:${AWS::Partition}:events:${AWS::Region}:${AWS::AccountId}:rule/
StepFunctionsGetEventsForStepFunctionsExecutionRule"
      ]
    }
  ]
}
```

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:StartExecution"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
```

```
      "states:StopExecution"
      ],
      "Resource": [
        "arn:${AWS::Partition}:states:${AWS::Region}:${AWS::AccountId}:execution:
%{Destination.Name}:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "events:PutTargets",
        "events:PutRule"
      ],
      "Resource": [
        "arn:${AWS::Partition}:events:${AWS::Region}:${AWS::AccountId}:rule/
StepFunctionsGetEventsForStepFunctionsExecutionRule"
      ]
    }
  ]
}
```

AWS::StepFunctions::StateMachine to AWS::Events::EventBus

Policy type

[Customer managed policy](#) attached to the AWS::StepFunctions::StateMachine role.

Access categories

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:PutEvents"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
```

```
    }
```

AWS::AppSync::DataSource to AWS::DynamoDB::Table

Policy type

[Customer managed policy](#) attached to the AWS::AppSync::DataSource role.

Access categories

Read

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:Query",
        "dynamodb:Scan",
        "dynamodb:BatchGetItem",
        "dynamodb:ConditionCheckItem",
        "dynamodb:PartiQLSelect"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}/index/*"
      ]
    }
  ]
}
```

Write

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

```
        "dynamodb:PartiQLDelete",
        "dynamodb:PartiQLInsert",
        "dynamodb:PartiQLUpdate"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}/index/*"
      ]
    }
  ]
}
```

AWS::AppSync::DataSource to AWS::Lambda::Function

Policy type

[Customer managed policy](#) attached to the `AWS::AppSync::DataSource` role.

Access categories

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeAsync",
        "lambda:InvokeFunction"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}:*"
      ]
    }
  ]
}
```

AWS::AppSync::DataSource to AWS::Events::EventBus

Policy type

[Customer managed policy](#) attached to the `AWS::AppSync::DataSource` role.

Access categories

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:PutEvents"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

`AWS::AppSync::GraphQLApi` to `AWS::Lambda::Function`

Policy type

[AWS::Lambda::Permission](#) attached to the `AWS::Lambda::Function`.

Access categories

Write

```
{
  "Action": "lambda:InvokeFunction",
  "Principal": "appsync.amazonaws.com",
  "SourceArn": "arn:${AWS::Partition}:appsync:${AWS::Region}:${AWS::AccountId}:apis/
%{Source.ResourceId}"
}
```

Installing Docker to use with the AWS SAM CLI

Docker is an application that runs containers on your machine. With Docker, AWS SAM can provide a local environment similar to AWS Lambda as a container to build, test, and debug your serverless applications.

> ⓘ **Note**
>
> Docker is required only for testing your applications locally and for building deployment
> packages using the `--use-container` option.

Topics

- [Installing Docker](#)
- [Next steps](#)

Installing Docker

Follow these instructions to install Docker on your operating system.

Linux

Docker is available on many different operating systems, including most modern Linux
distributions, such as CentOS, Debian, and Ubuntu. For information about installing Docker on your
particular operating system, see [Get Docker](#) on the Docker Docs website.

To install Docker on Amazon Linux 2 or Amazon Linux 2023

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

   ```
   $ sudo yum update -y
   ```

2. Install the most recent Docker Community Edition package.

   - For Amazon Linux 2, run the following:

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

   - For Amazon Linux 2023, run the following:

     ```
     $ sudo yum install -y docker
     ```

3. Start the Docker service.

   ```
   $ sudo service docker start
   ```

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

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

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

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

    ```
    $ docker ps
    ```

    You should see the following output, confirming that Docker is installed and running:

    ```
    CONTAINER ID        IMAGE           COMMAND             CREATED
    STATUS              PORTS           NAMES
    ```

> **ⓘ Note**
>
> On Linux, to build and run Lambda functions with a different instruction set architecture
> than your host machine, there are additional steps to configure Docker. For example, to run
> `arm64` functions on an `x86_64` machine, you can run the following command to configure
> the Docker daemon: `docker run --rm --privileged multiarch/qemu-user-static --reset -p yes`.

If you run into issues installing Docker, see Installation errors. Or, see the Troubleshooting section
of **Post-installation steps for Linux** on the Docker Docs website.

macOS

> **ⓘ Note**
>
> Docker Desktop is officially supported, but starting with AWS SAM CLI version 1.47.0, you
> can use alternatives as long as they use the Docker runtime.

1. Install Docker

    The AWS SAM CLI supports Docker running on macOS Sierra 10.12 or later. For how to install Docker, see Install Docker Desktop for Mac on the Docker Docs website.

2. Configure your shared drives

    The AWS SAM CLI requires that the project directory, or any parent directory, is listed in a shared drive. This may require file sharing. For more information, see the Volume mounting requires file sharing troubleshooting topic at *Docker docs*.

3. Verify the installation

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

If you run into issues installing Docker, for more troubleshooting tips, see the Troubleshoot and diagnose section of the Docker Docs website.

Windows

> ℹ️ **Note**
>
> AWS SAM officially supports Docker Desktop. However, starting with AWS SAM CLI version 1.47.0, you can use alternatives as long as they use the Docker runtime.

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 on the Docker Docs website.

    *   To install Docker for earlier versions of Windows, see The Docker Toolbox on the Docker Toolbox GitHub repository.

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 for Docker to function properly.

3. Verify the installation.

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

If you run into issues installing Docker, for more troubleshooting tips, see the [Troubleshoot and diagnose](#) section of the Docker Docs website.

Next steps

For how to install the AWS SAM CLI, see [Installing the AWS SAM CLI](#).

Image repositories

AWS SAM simplifies continuous integration and continuous delivery (CI/CD) tasks for serverless applications with the help of build container images. The images that AWS SAM provides include the AWS SAM command line interface (CLI) and build tools for a number of supported AWS Lambda runtimes. This make it easier to build and package serverless applications using the AWS SAM CLI. You can use these images with CI/CD systems to automate the building and deployment of AWS SAM applications. For examples, see [Deploying using CI/CD systems](#).

AWS SAM build container image URIs are tagged with the version of the AWS SAM CLI included in that image. If you specify the untagged URI, then the latest version is used. For example, `public.ecr.aws/sam/build-nodejs20.x` uses the latest image. However, `public.ecr.aws/sam/build-nodejs20.x:1.24.1` uses the the image containing AWS SAM CLI version 1.24.1.

Starting with version 1.33.0 of the AWS SAM CLI, both `x86_64` and `arm64` container images are available for supported runtimes. For more information, see [Lambda runtimes](#) in the *AWS Lambda Developer Guide*.

> ⓘ **Note**
>
> Prior to version 1.22.0 of the AWS SAM CLI, DockerHub was the default repository that the AWS SAM CLI pulled the container image from. Starting with version 1.22.0, the default

repository changed to Amazon Elastic Container Registry Public (Amazon ECR Public). To pull a container image from a repository other than the current default, you can use the **sam build** command with the **--build-image** option. The examples at the end of this topic show how to build applications using DockerHub repository images.

Image repository URIs

The following table lists the URIs of Amazon ECR Public build container images that you can use to build and package serverless applications with AWS SAM.

> ⓘ **Note**
>
> Amazon ECR Public replaced DockerHub starting with the AWS SAM CLI version 1.22.0. If you are using an earlier version of the AWS SAM CLI, we recommend that you upgrade.

Runtime	Amazon ECR Public
Custom runtime (AL2023)	public.ecr.aws/sam/build-provided.al2023
Custom runtime (AL2)	public.ecr.aws/sam/build-provided.al2
Custom runtime	public.ecr.aws/sam/build-provided
Go 1.x	public.ecr.aws/sam/build-go1.x
Java 21	public.ecr.aws/sam/build-java21
Java 17	public.ecr.aws/sam/build-java17
Java 11	public.ecr.aws/sam/build-java11
Java 8 (AL2)	public.ecr.aws/sam/build-java8.al2
Java 8	public.ecr.aws/sam/build-java8
.NET 8	public.ecr.aws/sam/build-dotnet8
.NET 7	public.ecr.aws/sam/build-dotnet7

Runtime	Amazon ECR Public
.NET 6	public.ecr.aws/sam/build-dotnet6
Node.js 20	public.ecr.aws/sam/build-nodejs20.x
Node.js 18	public.ecr.aws/sam/build-nodejs18.x
Node.js 16	public.ecr.aws/sam/build-nodejs16.x
Python 3.12	public.ecr.aws/sam/build-python3.12
Python 3.11	public.ecr.aws/sam/build-python3.11
Python 3.10	public.ecr.aws/sam/build-python3.10
Python 3.9	public.ecr.aws/sam/build-python3.9
Python 3.8	public.ecr.aws/sam/build-python3.8
Ruby 3.3	public.ecr.aws/sam/build-ruby3.3
Ruby 3.2	public.ecr.aws/sam/build-ruby3.2

Examples

The following two example commands build applications using container images from the DockerHub repository:

Build a Node.js 20 application using a container image pulled from DockerHub:

```
$ sam build --use-container --build-image amazon/aws-sam-cli-build-image-nodejs20.x
```

Build a function resource using the Python 3.12 container image pulled from DockerHub:

```
$ sam build --use-container --build-image Function1=amazon/aws-sam-cli-build-image-python3.12
```

Deploying serverless applications gradually

AWS Serverless Application Model (AWS SAM) comes built-in with [CodeDeploy](#) to provide gradual AWS 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. If an update doesn't work correctly, you can roll back the changes.

- Defines pre-traffic and post-traffic test functions to verify that the newly deployed code is configured correctly and that your application operates as expected.

- Automatically 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 the use of CodeDeploy to gradually shift customers to your newly deployed version of Lambda function:

```
Resources:
MyLambdaFunction:
  Type: AWS::Serverless::Function
  Properties:
    Handler: index.handler
    Runtime: nodejs12.x
    CodeUri: s3://bucket/code.zip

    AutoPublishAlias: live

    DeploymentPreference:
      Type: Canary10Percent10Minutes
```

```
    Alarms:
      # A list of alarms that you want to monitor
      - !Ref AliasErrorMetricGreaterThanZeroAlarm
      - !Ref LatestVersionErrorMetricGreaterThanZeroAlarm
    Hooks:
      # Validation Lambda functions that are run before & after traffic shifting
      PreTraffic: !Ref PreTrafficLambdaFunction
      PostTraffic: !Ref PostTrafficLambdaFunction
```

These revisions to the AWS SAM template do the following:

- `AutoPublishAlias`: By adding this property and specifying an alias name, AWS SAM:

  - Detects when new code is being deployed, based on changes to the Lambda function's Amazon S3 URI.

  - Creates and publishes an updated version of that function with the latest code.

  - Creates an alias with a name that you provide (unless an alias already exists), and points to the updated version of the Lambda function. Function invocations should use the alias qualifier to take advantage of this. If you aren't familiar with Lambda function versioning and aliases, see [AWS Lambda Function Versioning and Aliases](#).

- `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-traffic or post-traffic tests fail, or if a CloudWatch alarm is triggered, CodeDeploy rolls back your deployment. You can specify how traffic should be shifted between versions in the following ways:

  - `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.

  - `AllAtOnce`: All traffic is shifted from the original Lambda function to the updated Lambda function version at once.

  The following table outlines other traffic-shifting options that are available beyond the one used in the example.

Deployment Preference Type
Canary10Percent30Minutes
Canary10Percent5Minutes
Canary10Percent10Minutes
Canary10Percent15Minutes
Linear10PercentEvery10Minutes
Linear10PercentEvery1Minute
Linear10PercentEvery2Minutes
Linear10PercentEvery3Minutes
AllAtOnce

- `Alarms`: These are CloudWatch alarms that are triggered by any errors raised by the deployment. When encountered, they automatically roll back your deployment. For example, if the updated code you're deploying causes 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 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.

Gradually deploying a Lambda function for the first time

When deploying a Lambda function gradually, CodeDeploy requires a previously deployed function version to shift traffic from. Therefore, your first deployment should be accomplished in two steps:

- **Step 1**: Deploy your Lambda function and automatically create aliases with `AutoPublishAlias`.

- **Step 2**: Perform your gradual deployment with `DeploymentPreference`.

Performing your first gradual deployment in two steps gives CodeDeploy a previous Lambda function version to shift traffic from.

Step 1: Deploy your Lambda function

```
Resources:
MyLambdaFunction:
  Type: AWS::Serverless::Function
  Properties:
    Handler: index.handler
    Runtime: nodejs12.x
    CodeUri: s3://bucket/code.zip

    AutoPublishAlias: live
```

Step 2: Perform your gradual deployment

```
Resources:
MyLambdaFunction:
  Type: AWS::Serverless::Function
  Properties:
    Handler: index.handler
    Runtime: nodejs12.x
    CodeUri: s3://bucket/code.zip

    AutoPublishAlias: live

    DeploymentPreference:
      Type: Canary10Percent10Minutes
      Alarms:
        # A list of alarms that you want to monitor
```

```
        - !Ref AliasErrorMetricGreaterThanZeroAlarm
        - !Ref LatestVersionErrorMetricGreaterThanZeroAlarm
    Hooks:
      # Validation Lambda functions that are run before and after traffic shifting
      PreTraffic: !Ref PreTrafficLambdaFunction
      PostTraffic: !Ref PostTrafficLambdaFunction
```

Learn more

For a hands-on example of configuring a gradual deployment, see Module 5 - Canary Deployments
in *The Complete AWS SAM Workshop*.

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 command line interface (CLI) collects telemetry, such as generic usage metrics,
system and environment information, and errors. For details about the types of telemetry
collected, see Types of information collected.

The AWS SAM CLI does **not** collect personal information, such as user names or email addresses. It
also does not extract sensitive project-level information.

Customers control whether telemetry is turned on, and they can change their settings at any point
of time. If telemetry remains on, the AWS SAM CLI sends telemetry data in the background without
requiring any additional customer interaction.

Turn off telemetry for a session

In macOS and Linux operating systems, you can turn off telemetry for a single session. To turn off
telemetry for your current session, run the following command to set the environment variable
SAM_CLI_TELEMETRY to `false`. Repeat the command for each new terminal or session.

```
export SAM_CLI_TELEMETRY=0
```

Turn off telemetry for your profile in all sessions

Run the following commands to turn off telemetry for all sessions when you're running the AWS SAM CLI on your operating system.

To turn off telemetry in Linux

1. Run:

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

2. Run:

```
source ~/.profile
```

To turn off telemetry in macOS

1. Run:

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

2. Run:

```
source ~/.profile
```

To turn off telemetry in Windows

You can set the environment variable temporarily for the lifetime of the terminal window with the following command:

If using Command Prompt:

```
set SAM_CLI_TELEMETRY 0
```

If using PowerShell:

```
$env:SAM_CLI_TELEMETRY=0
```

To set the environment variable permanently in either the Command Prompt or PowerShell, use the following command:

```
setx SAM_CLI_TELEMETRY 0
```

> ℹ️ **Note**
>
> Changes will not go into effect until the terminal has been closed and reopened.

Types of information collected

- **Usage information** – The generic commands and subcommands that customers run.
- **Errors and diagnostic information** – The status and duration of commands that customers 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), environment in which the AWS SAM CLI runs (for example, AWS CodeBuild, an AWS IDE toolkit, or a terminal), and hash values of usage attributes.

Learn more

The telemetry data that the AWS SAM CLI collects adheres to the AWS data privacy policies. For more information, see the following:

- AWS Service Terms
- Data Privacy FAQ

Important notes

This section contains important notes and announcements for AWS Serverless Application Model (AWS SAM).

Topics

- Important notes for 2023
- Important notes for 2020

Important notes for 2023

October 2023

AWS SAM CLI discontinuing support for Python 3.7

Published on 2023-10-20

Python 3.7 received end-of-life status in June of 2023. The AWS SAM CLI will discontinue support for Python 3.7 on October 24, 2023. For more information, see the [announcement](#) at the *aws-sam-cli GitHub repository*.

This change impacts the following users:

- If you use Python 3.7 and install the AWS SAM CLI through `pip`.
- If you use the `aws-sam-cli` as a library and build your application with Python 3.7.

If you install and manage the AWS SAM CLI through another method, you are not affected.

For impacted users, we recommend that you upgrade your development environment to Python 3.8 or newer.

This change does not affect support for the Python 3.7 AWS Lambda runtime environment. For more information, see [Runtime deprecation policy](#) in the *AWS Lambda Developer Guide*.

Important notes for 2020

June 2020

Installing the 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](#).

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:** December 6, 2023

Change	Description	Date
[Add AWS SAM CLI arm64 package installer for Linux](#)	For instructions, see [Installing the AWS SAM CLI](#).	December 6, 2023
[Add --watch-exclude option for the AWS SAM CLI sam sync command](#)	Exclude files and folders from initiating a sync. To learn more, see [Specify files and folders that won't initiate a sync](#).	December 6, 2023
[Add --build-in-source option for the AWS SAM CLI sam sync command](#)	Build your project in your source folder to speed up the build process. To learn more, see [Speed up build times by building your project in the source folder](#).	December 6, 2023
[Add --build-in-source option for the AWS SAM CLI sam build command](#)	Build your project in your source folder to speed up the build process. To learn more, see [Speed up build times by building your project in the source folder](#).	December 6, 2023
[Add new resource support for AWS SAM CLI remote invoke command](#)	Use `sam remote invoke` with Kinesis Data Streams applications, Amazon SQS queues, and Step Functions	November 15, 2023

state machines. To learn
more, see Using sam remote
invoke.

Add new AWS SAM CLI remote test-event command for shareable test events	Use the AWS SAM CLI to access and manage shareable test events from the EventBridge schema registry to test your Lambda functions in the AWS Cloud. To learn more, see Using sam remote test-event.	October 3, 2023
Add YAML file format support for the AWS SAM CLI configuration file	The AWS SAM CLI now supports the [.yaml|.yml] file format. Configuring the AWS SAM CLI and AWS SAM CLI configuration file pages have been updated.	July 18, 2023
AWS SAM CLI support for Terraform is now generally available	To learn more about AWS SAM CLI support for Terraform, see AWS SAM CLITerraform support.	September 5, 2023
Add AWS SAM CLI support for Terraform Cloud	The AWS SAM CLI now supports local testing for Terraform Cloud. To learn more, see Set up for Terraform Cloud.	September 5, 2023
Add AWS SAM CLIsam local start-api command support for Terraform	The What is AWS SAM CLI support for Terraform? section has been updated to include AWS SAM CLI sam local start-api command support for Terraform.	July 6, 2023

Add new AWS SAM CLI remote invoke command	To start using `sam remote invoke`, see Using sam remote invoke.	June 22, 2023
Add AWS AppSyncGraphQL API serverless resource type	Create new `AWS::Serverless::GraphQLApi` section that describes how to define a GraphQL API resource with AWS SAM.	June 22, 2023
Add AWS SAM CLI support for Ruby 3.2	Update sam init page to include new base image and runtime values. Update Image repositories page with Ruby 3.2 Amazon ECR URI.	June 6, 2023
Add optional steps for integrity verification of the AWS SAM CLI package installer	Update Installing the AWS SAM CLI page to reflect optional step. Create Verify the integrity of the AWS SAM CLI installer page to document steps.	May 31, 2023
Add sam sync option to skip infrastructure sync	Customize whether an AWS CloudFormation deployment is required each time `sam sync` is run. To learn more, see Skip the initial AWS CloudFormation deployment.	March 23, 2023
Add support for DocumentDB event source type	The AWS SAM template specification now supports DocumentDB event source type for the `AWS::Serverless::Function` resource. To learn more, see DocumentDB.	March 10, 2023

[Build Rust Lambda functions with Cargo Lambda](#)	Use the AWS SAM CLI to build your Rust Lambda functions using Cargo Lambda. To learn more, see [Building Rust Lambda functions with Cargo Lambda.](#)	February 23, 2023
[Build function resources outside of AWS SAM](#)	Added guidance on skipping functions when using the **sam build** command. To learn more, see [Building functions outside of AWS SAM](#).	February 14, 2023
[New embedded connectors syntax](#)	Use the new embedded connectors syntax to define your `AWS::Serverless::Connector` resources. To learn more, see [Managing resource permissions with AWS SAM connectors](#).	February 8, 2023
[Added new sam list command for the AWS SAM CLI](#)	Use **sam list** to view important information about the resources in your serverless application. To learn more, see [sam list](#).	February 2, 2023
[Added Format and OutExtension build properties for esbuild](#)	Building Node.js Lambda functions with esbuild now supports `Format` and `OutExtension` build properties. To learn more, see [Building Node.js Lambda functions with esbuild](#).	February 2, 2023

[Added runtime managemen t options to the AWS SAM template specification](#)	Configure runtime management options for your Lambda functions. To learn more, see `RuntimeMa nagementConfig` .	January 24, 2023
[Target property added to EventSource for AWS::Serv erless::StateMachine resource.](#)	`AWS::Serverless::S tateMachine` resource type supports the `Target` property for [EventBrid geRule](#) and [Schedule](#) event sources.	January 13, 2023
[Configure scaling of SQS pollers for Lambda functions](#)	Configure scaling of SQS pollers with the `ScalingCo nfig` property for `AWS::Serverless::F unction` . To learn more, see [ScalingConfig](#) .	January 12, 2023
[Validate AWS SAM applicati ons with cfn-lint](#)	You can use cfn-lint to validate your AWS SAM templates through the AWS SAM CLI. To learn more, see [Validate with cfn-lint](#).	January 11, 2023
[Monitor your serverless applications with CloudWatch Application Insights](#)	Configure Amazon CloudWatc h Application Insights to monitor your AWS SAM applications. To learn more, see [Monitor your serverless applications with CloudWatch Application Insights](#).	December 19, 2022

Added AWS SAM CLI package installer for macOS	Install the AWS SAM CLI using the new macOS package installer. To learn more, see Installing the AWS SAM CLI.	December 6, 2022
Added support for Lambda SnapStart	Configure SnapStart for your Lambda functions to create snapshots, which are cached states of your initialized functions. To learn more, see AWS::Serverless::F unction .	November 28, 2022
Added AWS SAM CLI support for nodejs18.x	AWS SAM CLI now supports nodejs18.x runtime. To learn more, see sam init.	November 17, 2022
Added guidance on configuri ng access and permissions	AWS SAM provides two options that simplify management of access and permissions for your serverle ss applications.To learn more, see Managing resource access and permissions.	November 17, 2022
Added support for building .NET 7 Lambda functions with Native AOT compilation	Build and package your .NET 7 Lambda functions with AWS SAM, utilizing Native Ahead- of-Time (AOT) compilation to improve Lambda cold-star t times. To learn more, see Building .NET 7 Lambda functions with Native AOT compilation.	November 15, 2022

[Added AWS SAM CLITerraf orm support for local debugging and testing](#)	Use the AWS SAM CLI within your Terraform projects to perform local debugging and testing of your Lambda functions and layers. To learn more, see [AWS SAM CLI Terraform support](#).	November 14, 2022
[Added AWS SAM support for EventBridge Scheduler](#)	The AWS Serverless Applicati on Model (AWS SAM) template specification provides a simple, short-hand syntax that you can use to schedule events with EventBridge Scheduler for AWS Lambda and AWS Step Functions. For more informat ion, see [Scheduling events with EventBridge Scheduler](#).	November 10, 2022
[Simplified the AWS SAM CLI installation instructions](#)	AWS SAM CLI prerequisites and optional steps have been moved to separate pages. Installation steps for supported operating systems can be found at [Installing the AWS SAM CLI](#).	November 4, 2022
[Added fix to allow long paths for Windows 10 users](#)	The AWS SAM CLI app templates repository contains some long file paths which may cause errors when running `sam init` due to Windows 10 **MAX_PATH** lim itations. For more informati on, see [Installing the AWS SAM CLI](#)	November 4, 2022

Updated gradual deployment process for first time deployments	Gradually deploying a Lambda function with AWS CodeDeploy requires two steps. To learn more, see Gradually deploying a Lambda function for the first time.	October 13, 2022
Additional Lambda event filtering support for more types of events	`FilterCritera` property added to MSK, MQ, and `SelfManagedKafka` event source types.	October 13, 2022
Added OpenID Connect (OIDC) support for AWS SAM pipeline	AWS SAM supports OpenID Connect (OIDC) user authentication for Bitbucket, GitHub Actions, and GitLab continuous integration and continuous delivery (CI/CD) platforms. To learn more, see Using OIDC User Accounts with AWS SAM pipeline.	October 13, 2022
Note on JwtConfiguration properties	Added note on defining `issuer` and `audience` properties under `JwtConfiguration` for `OAuth2Authorizer`.	October 7, 2022

New properties for Function and StateMachine EventSource	Enabled and State properties added to CloudWatchEvent event source for AWS::Serverless::Function . State property added to Schedule event source for AWS::Serverless::Function and AWS::Serverless::StateMachine .	October 6, 2022
AWS SAM connectors now generally available	Connectors are an AWS SAM abstract resource type, identified as AWS::Serverless::Connector , that provides a simple and secure method of provisioning permissions between your serverless application resources. To learn more, see Managing resource permissions with AWS Serverless Application Model connectors.	October 6, 2022
Added new sam sync options to the AWS SAM CLI	--dependency-layer and --use-container options added to **sam sync**.	September 20, 2022
Added new sam deploy options to the AWS SAM CLI	--on-failure option added to **sam deploy**.	September 9, 2022
esbuild support now generally available	To build and package Node.js Lambda functions, you can use the AWS SAM CLI with the esbuild JavaScript bundler.	September 1, 2022

Updated AWS SAM CLI telemetry	Description of **system and environment information** collected has been updated to include hash values of usage attributes.	September 1, 2022
Added local environment variable support to AWS SAM CLI	Use environment variables with AWS SAM CLI when invoking Lambda functions locally and when running API Gateway locally.	September 1, 2022
Support for Lambda instruction set architectures	Use the AWS SAM CLI to build Lambda functions and Lambda layers for x86_64 or arm64 instruction set architectures. For more information, see the Architectures property of the AWS::Serverless::Function resource type and the CompatibleArchitectures property of the AWS::Serverless::LayerVersion resource type.	October 1, 2021
Generating example pipeline configurations	Use the AWS SAM CLI to generate example pipelines for multiple CI/CD systems, using the new sam pipeline bootstrap and sam pipeline init commands. For more information, see Generating example CI/CD pipelines.	July 21, 2021

[AWS SAM CLIAWS CDK integration (preview, phase 2)](#)	With phase 2 of the public preview release, you can now use the AWS SAM CLI to package and deploy AWS CDK applications. You can also download a sample AWS CDK application directly using the AWS SAM CLI. For more information, see [AWS Cloud Development Kit (AWS CDK) (Preview)](#).	July 13, 2021
[Support for RabbitMQ as an event source for functions](#)	Added support for RabbitMQ as an event source for serverless functions. For more information, see the [`SourceAccessConfig urations `](#) property of the MQ event source of the [`AWS::Serverless::F unction `](#) resource type.	July 7, 2021
[Deploying serverless applicati ons using Amazon ECR build container images](#)	Use Amazon ECR build container images to deploy serverless applications with common CI/CD systems such as AWS CodePipeline, Jenkins, GitLab CI/CD, and GitHub Actions. For more information, see [Deploying serverless applications](#).	June 24, 2021

[Debugging AWS SAM applications with AWS Toolkits](#)	AWS Toolkits now supports step-through debugging with more combinations of integrated developme nt environments (IDEs) and runtimes. For more informati on, see [Using AWS Toolkits](#).	May 20, 2021
[AWS SAM CLIAWS CDK integration (preview)](#)	You can now use the AWS SAM CLI to locally test and build AWS CDK applicati ons. This is a public preview release. For more information, see [AWS Cloud Development Kit (AWS CDK) (Preview)](#).	April 29, 2021
[Default container image repository changed to Amazon ECR Public](#)	The default container image repository changed from DockerHub to [Amazon ECR Public](#). For more information, see [Image repositories](#).	April 6, 2021
[Nightly AWS SAM CLI builds](#)	You can now install a pre-release version of the AWS SAM CLI, which is built nightly. For more informati on, see the **Nightly build** section of the OS subtopic of your choice under [Installing t he AWS SAM CLI](#).	March 25, 2021

Build container environment variables support	You can now pass environme nt variables to build container s. For more information, see the `--container-env-va r` and `--container-env- var-file` options in [sam build]().	March 4, 2021
New Linux installation process	You can now install the AWS SAM CLI using a native Linux installer. For more informat ion, see [Installing the AWS SAM CLI on Linux]().	February 10, 2021
Support for dead-letter queues for EventBridge	Added support for dead-lett er queues for EventBridge and `Schedule` event sources for serverless functions and state machines. For more informati on, see the `DeadLette rConfig` property of the `EventBridgeRule` and `Schedule` event sources, for both the [AWS::Serv erless::Function]() and [AWS::Serverless::S tateMachine]() resource types.	January 29, 2021

[Support for custom checkpoin ts](#)	Added support for custom checkpoints for DynamoDB and Kinesis event sources for serverless functions. For more information, see the `FunctionResponseTy pes` property of the [Kinesis](#) and [DynamoDB](#) data types of the [AWS::Serv erless::Function](#) resource type.	January 29, 2021
[Support for tumbling windows](#)	Added support for tumbling windows for DynamoDB and Kinesis event sources for serverless functions. For more information, see the `TumblingWindowInSe conds` property of the [Kinesis](#) and [DynamoDB](#) data types of the [AWS::Serv erless::Function](#) resource type.	December 17, 2020
[Support for warm containers](#)	Added support for warm containers when testing locally using the AWS SAM CLI commands [sam local start-api](#) and [sam local start-lambda](#). For more information, see the `-- warm-containers` option for those commands.	December 16, 2020

[Support for Lambda container images](#)	Added support for Lambda container images. For more information, see [Building applications](#).	December 1, 2020
[Support for code signing](#)	Added support for code signing and trusted deployments of serverless application code. For more information, see [Configuring code signing for AWS SAM applications](#).	November 23, 2020
[Support for parallel and cached builds](#)	Improved performance of serverless application builds by adding two options to the `sam build` command: `--parallel`, which builds functions and layers in parallel rather than sequentially, and `--cached`, which uses build artifacts from previous builds when no changes have been made that requires a rebuild.	November 10, 2020

Support for Amazon MQ, and mutual TLS authentication	Added support for Amazon MQ as an event source for serverless functions . For more information, see the `EventSource` and `MQ` data types of the `AWS::Serverless::Function` resource type. Also added support for mutual Transport Layer Security (TLS) authentication for API Gateway APIs and HTTP APIs. For more information, see the `DomainConfiguration` data type of the `AWS::Serverless::Api` resource type, or the `HttpApiDomainConfiguration` data type of the `AWS::Serverless::HttpApi` resource type.	November 5, 2020
Support for Lambda authorizers for HTTP APIs	Added support for Lambda authorizers for the `AWS::Serverless::HttpApi` resource type. For more information, see Lambda authorizer example (`AWS::Serverless::HttpApi`).	October 27, 2020

[Support for multiple configuration files and environments](#)	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](#).	September 24, 2020
[Support for X-Ray with Step Functions, and references when controlling access to APIs](#)	Added support for X-Ray as an event source for serverles s state machines. For more information, see the `Tracing` property of the `AWS::Serverless::StateMachine` resource type. Also added support for references when controlling access to APIs. For more information, see the `ResourcePolicyStatement` data type.	September 17, 2020
[Support for Amazon MSK](#)	Added support for Amazon MSK as an event source for serverless functions. This allows records in an Amazon MSK topic to trigger your Lambda function. For more information, see the `EventSource` and `MSK` data types of the `AWS::Serverless::Function` resource type.	August 13, 2020

Support for Amazon EFS	Added support for mounting Amazon EFS file systems to local directories. This allows your Lambda function code to access and modify shared resources. For more information, see the `FileSystemConfigs` property of the `AWS::Serverless::Function` resource type.	June 16, 2020
Orchestrating serverless applications	Added support for orchestrating applications by creating Step Functions state machines using AWS SAM. For more information, see Orchestrating AWS resources with AWS Step Functions and the `AWS::Serverless::StateMachine` resource type.	May 27, 2020
Building custom runtimes	Added the ability to build custom runtimes. For more information, see Building custom runtimes.	May 21, 2020
Building layers	Added the ability to build individual `LayerVersion` resources. For more information, see Building layers.	May 19, 2020

[Generated AWS CloudForm ation resources](#)	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
[Setting up AWS credentials](#)	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
[AWS SAM specification and AWS SAM CLI updates](#)	Migrated the AWS SAM specification from GitHub. For more information, see [AWS SAM specification](#). Also updated the deployment workflow with changes to the `sam deploy` command.	November 25, 2019

New options for controlling access to API Gateway APIs and policy template updates	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
Getting started updates	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.	July 25, 2019
Controlling access to API Gateway APIs	Added support for controlling access to API Gateway APIs. For more information, see Controlling access to API Gateway APIs.	March 21, 2019
Added sam publish to the AWS SAM CLI	The new sam publish 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.	December 21, 2018

[Nested applications and layers support](#)	Added support for nested applications and layers. For more information, see [Using nested applications](#) and [Working with layers](#).	November 29, 2018
[Added `sam build` to the AWS SAM CLI](#)	The new `sam build` 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](#).	November 19, 2018
[Added new installation options for the AWS SAM CLI](#)	Added Linuxbrew (Linux), MSI (Windows), and Homebrew (macOS) installation options for the AWS SAM CLI. For more information, see [Installing the AWS SAM CLI](#).	November 7, 2018
[New guide](#)	This is the first release of the *AWS Serverless Application Model Developer Guide*.	October 17, 2018