Amazon Serverless Application Model

开发人员指南

Amazon Serverless Application Model: 开发人员指南

Table of Contents

Amazon Serverless Application Model(Amazon SAM) 是什么？

Amazon Serverless Application Model(Amazon SAM) 是一个工具包，可改善开发人员在上构建和运行无服务器应用程序的体验Amazon。 Amazon SAM由两个主要部分组成：

1. Amazon SAM模板规范 — 一个开源框架，可用于在其上定义无服务器应用程序基础架构Amazon。
2. Amazon SAM命令行界面 (Amazon SAMCLI) — 命令行工具，您可以将其与Amazon SAM模板和支持的第三方集成一起使用，以构建和运行无服务器应用程序。

您是无服务器的新手吗？

我们建议您对以下主题有基本的了解：

- Event-driven architecture (p. 12)
- Infrastructure as Code (IaC) (p. 12)
- Serverless technologies (p. 12)

要了解更多信息，请参阅 无服务器概念 (p. 12)。

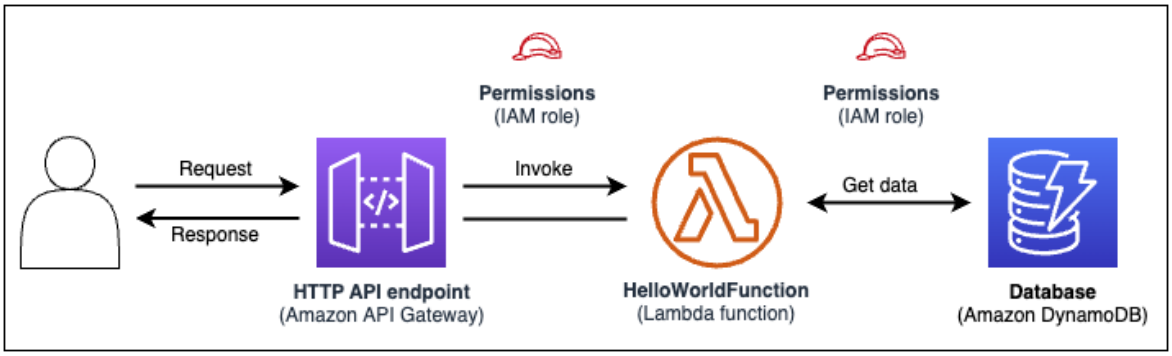
Amazon SAM模板规格是什么？

Amazon SAM模板规范是一个开源框架，您可以使用它来定义和管理您的无服务器应用程序基础架构代码。Amazon SAM模板规范是：

- 内@@ 置Amazon CloudFormation — 利用其对资源和属性配置的广泛支持，直接在Amazon SAM模板中使用Amazon CloudFormation语法。如果您已经熟悉Amazon CloudFormation，则无需学习新服务即可管理您的应用程序基础架构代码。
- Amazon CloudFormation— 的扩展Amazon SAM提供了自己独特的语法，专门用于加快无服务器开发。可以在同一个模板中同时使用Amazon CloudFormation和Amazon SAM语法。
- 抽象的简短语法 — 使用该Amazon SAM语法，您可以快速定义基础架构，减少代码行数，降低出错几率。它的语法经过精心设计，旨在抽象化定义无服务器应用程序基础架构的复杂性。
- 转换Amazon SAM — 完成将模板转换为配置基础架构所需的代码的复杂工作Amazon CloudFormation。

下面是一个基本的无服务器应用程序示例。此应用程序处理通过 HTTP 请求从数据库获取所有项目的请求。它由以下部分组成：

1. 一个包含处理请求的逻辑的函数。
2. 一个 HTTP API，用作客户端（请求者）和应用程序之间的通信。
3. 用于存储物品的数据库。
4. 应用程序安全运行的权限。



可以在以下Amazon SAM模板中定义此应用程序的基础架构代码：

```
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
```

在 23 行代码中，定义了以下基础架构：

- 使用Amazon Lambda服务的函数。
- 使用Amazon API Gateway 服务的 HTTP API。
- 使用Amazon DynamoDB 服务的数据库。
- 这些服务相互交互所需的Amazon Identity and Access Management (IAM) 权限。

要配置此基础架构，将模板部署到Amazon CloudFormation。在部署期间，Amazon SAM将 23 行代码转换为生成这些资源所需的Amazon CloudFormation语法Amazon。转换后的Amazon CloudFormation模板包含 200 多行代码！

转换后的Amazon CloudFormation模板

```
{
  "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": [
          "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"
          }
        ]
      }
    }
  }
}
```

通过使用Amazon SAM，您可以定义 23 行基础架构代码。 Amazon SAM将您的代码转换为预置应用程序所需的 200 多行Amazon CloudFormation代码。

什么是Amazon SAM CLI？

Amazon SAMCLI 是一种命令行工具，您可以将其与Amazon SAM模板和支持的第三方集成一起使用，以构建和运行您的无服务器应用程序。使用Amazon SAM CLI 可以：

- 快速初始化新的应用程序项目。
- 生成要部署的应用程序。
- 执行本地调试和测试。
- 部署您的应用程序。
- 配置 CI/CD 部署管道。
- 监控云中的应用程序并对其进行故障排除。
- 在开发时将本地更改同步到云端。
- 还有更多！

CAmazon SAM LI 与Amazon SAM和Amazon CloudFormation模板一起使用时效果最佳。它也可以与第三方产品一起使用，例如Terraform.

初始化新项目

从起始模板中选择或选择自定义模板位置以开始新项目。

在这里，我们使用该sam init命令来初始化一个新的应用程序项目。我们选择 Hello World 示例项目作为开始。CAmazon SAM LI 下载入门模板并创建我们的项目文件夹目录结构。

```
→  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]:
```

生成要部署的应用程序

Package 函数依赖关系并组织项目代码和文件夹结构，为部署做准备。

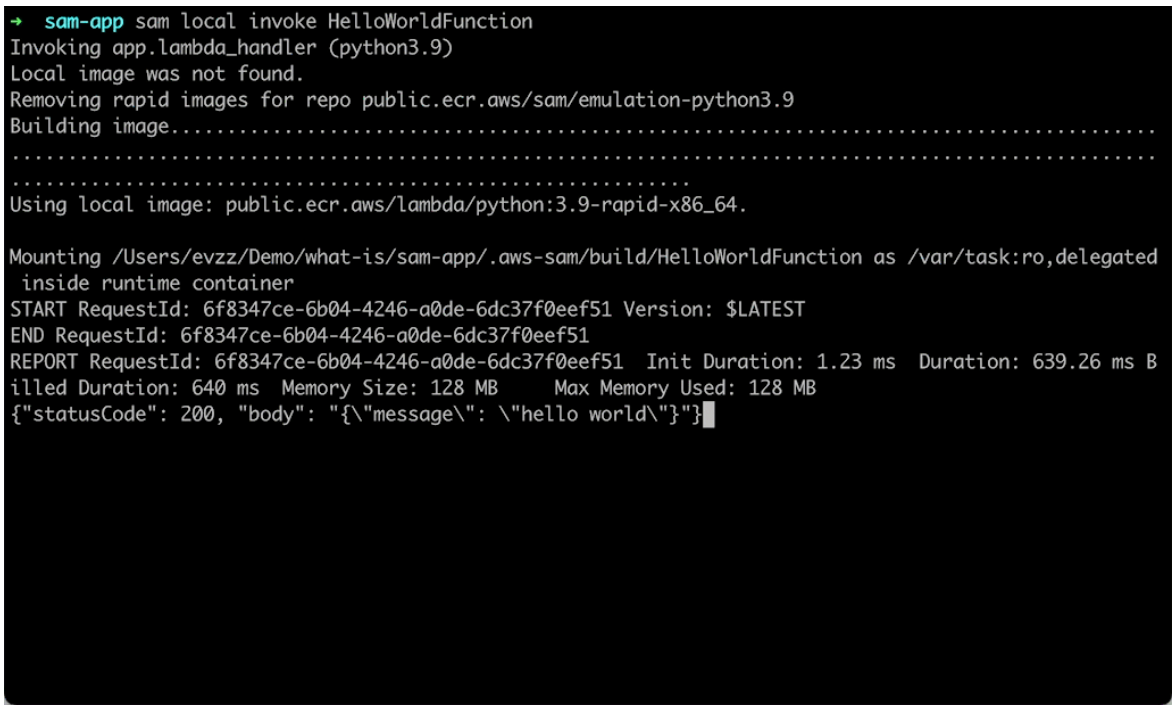
在这里，我们使用sam build命令来准备应用程序以进行部署。CAmazon SAM LI 创建一个`.aws-sam`目录并在其中组织我们的应用程序依赖关系和文件以进行部署。

```
→  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
```

执行本地调试和测试

在本地计算机上，模拟事件、测试 API、调用函数等，以调试和测试应用程序。

在这里，我们使用sam local invoke命令`HelloWorldFunction`在本地调用。为此，Amazon SAMCLI 创建了一个本地容器，构建我们的函数，调用它并输出结果。



部署您的应用程序

配置应用程序的部署设置并部署到Amazon云端以配置您的资源。

在这里，我们使用sam deploy --guided命令通过交互式流程部署我们的应用程序。CAmazon SAM LI 指导我们配置应用程序的部署设置，将模板转换为Amazon CloudFormation，然后部署Amazon CloudFormation到创建资源。

```
→  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
```

配置 CI/CD 部署管道

使用支持的 C I/CD 系统创建安全的持续集成和交付 (CI/CD) 管道。

在这里，我们使用sam pipeline init --bootstrap命令为我们的应用程序配置 CI/CD 部署管道。CAmazon SAM LI 指导我们了解我们的选项，并生成用于我们的 CI/CD 系统的Amazon资源和配置文件。

```
[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]:
```

监控云中的应用程序并对其进行故障排除

查看有关已部署资源的重要信息，收集日志，并使用内置监控工具，例如Amazon X-Ray。

在这里，我们使用sam list命令来查看我们部署的资源。我们获取我们的 API 端点并调用它，这会触发我们的函数。然后，我们sam logs用来查看函数的日志。

```
→  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
```

在开发时将本地更改同步到云端

在本地计算机上开发时，自动将更改同步到云端。快速查看您的更改并在云端执行测试和验证。

在这里，我们使用sam sync --watch命令让Amazon SAM CLI 监视本地更改。我们会修改我们的`HelloWorldFunction`代码，Amazon SAMCLI 会自动检测更改并将我们的更新部署到云端。

使用 Amazon SAM 的好处

以下是一些您可以使用以下方法完成的示例Amazon SAM：

使用更少的代码快速定义应用程序基础架构代码

编写Amazon SAM模板来定义您的无服务器应用程序基础架构代码。将您的模板直接部署Amazon CloudFormation到以配置您的资源。

在无服务器应用程序的整个开发生命周期中对其进行管理

使用Amazon SAM CLI 在开发生命周期的创作、构建、部署、测试和监控阶段管理您的无服务器应用程序。有关更多信息，请参阅使用 Amazon SAM CLI (p. 37)：

使用Amazon SAM连接器在资源之间快速配置权限

在Amazon SAM模板中使用Amazon SAM连接器来定义Amazon资源之间的权限。 Amazon SAM将您的代码转换为促进您的意图所需的 IAM 权限。有关更多信息，请参阅使用Amazon SAM连接器管理资源权限 (p. 273)：

在开发时持续将本地更改同步到云端

使用Amazon SAM CLIsam sync 命令自动将本地更改同步到云端，从而加快开发和云测试工作流程。有关更多信息，请参阅使用 sam sync (p. 68)：

管理您的Terraform无服务器应用程序

使用Amazon SAM CLI 对您的 Lambda 函数和层进行本地调试和测试。有关更多信息，请参阅Amazon SAMCLI Terraform 支持 (p. 388)：

了解更多信息

要继续了解有关的更多信息Amazon SAM，请参阅以下资源：

- 完整Amazon SAM研讨会 — 该研讨会旨在向您介绍所Amazon SAM提供的许多主要功能。
- 与 SAM 的会话 — 我们的Amazon无服务器开发者倡导者团队制作的关于使用的视频系列Amazon SAM。
- Serverless Land — 汇集了Amazon无服务器的最新信息、博客、视频、代码和学习资源的站点。

后续步骤

如果这是您第一次使用Amazon SAM，请参阅Amazon SAM 入门 (p. 13)。

无服务器概念

在使用Amazon Serverless Application Model (Amazon SAM) 之前，请先了解基本的无服务器概念。

无服务器概念

事件驱动型架构

无服务器应用程序由单独的服务组成，例如Amazon Lambda用于计算的Amazon服务以及用于数据库管理的 Amazon DynamoDB，每项服务都发挥专门的作用。然后，这些服务通过事件驱动的架构相互松散地集成。要了解有关事件驱动型架构，请参阅什么是事件驱动型架构？ 。

基础设施即代码 (iaC)

基础设施即代码 (IaC) 是一种以开发人员对待代码的相同方式处理基础架构的方法，将同样严格的应用程序代码开发应用于基础架构配置。您可以在模板文件中定义基础架构，将其部署到模板文件中Amazon，然后为您Amazon创建资源。使用 IAC，您可以在代码中定义Amazon要预置的内容。有关更多信息，请参阅AmazonAmazon白皮书简介中的基础架构即代码。 DevOps

无服务器技术

借助Amazon无服务器技术，您无需管理自己的服务器即可构建和运行应用程序。所有服务器管理均由此完成Amazon，提供了许多好处，例如自动扩展和内置高可用性，使您可以快速将想法付诸生产。使用无服务器技术，您可以专注于产品的核心，而不必担心管理和操作服务器。要了解有关Serverlessss Serverlessss

- 无服务器开启Amazon
- 无服务器开发人员指南 — 提供Amazon云端无服务器开发的概念性概述。

有关核心Amazon无服务器服务的基本介绍，请参阅 Server less 101：了解 Serverless L and 上的无服务器服务。

后续步骤

有关简介Amazon SAM，请参见Amazon Serverless Application Model(Amazon SAM) 是什么？ (p. 1)

Amazon SAM 入门

通过安装Amazon SAM命令行界面Amazon Serverless Application Model (Amazon SAMAmazon SAMCLI) 开始使用 ()。

主题

Amazon SAM先决条件

在安装和使用Amazon Serverless Application Model命令行界面 (Amazon SAMCLI) 之前，请完成以下先决条件。

要使用 CAmazon SAM LI，您需要满足以下条件：

- Amazon账户、Amazon Identity and Access Management (IAM) 凭证和 IAM 访问key pair。
- Amazon Command Line Interface(Amazon CLI) 用于配置Amazon证书。

主题

步骤 1：注册 Amazon 账户

如果您还没有 Amazon Web Services 账户，请完成以下步骤来创建一个。

注册 Amazon Web Services 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，您将接到一通电话，要求您使用电话键盘输入一个验证码。

   当您注册 Amazon Web Services 账户 时，系统将会创建一个 Amazon Web Services 账户根用户。根用户有权访问该账户中的所有 Amazon Web Services和资源。作为安全最佳实践，请 为管理用户分配管理访问权限，并且只使用根用户执行 需要根用户访问权限的任务。

步骤 2：创建 IAM 用户账户

保护 IAM 用户

注册 Amazon Web Services 账户 后，启用多重身份验证（MFA）保护您的管理用户。有关说明，请参阅 IAM 用户指南中的为 为 IAM 用户（控制台）启用虚拟 MFA 设备。

要授予其他用户访问您的 Amazon Web Services 账户 资源的权限，请创建 IAM 用户。为了保护您的 IAM 用户，请启用 MFA 并仅向 IAM 用户授予执行任务所需的权限。

有关创建和保护 IAM 用户的更多信息，请参阅 IAM 用户指南中的以下主题：

- 在您的 Amazon Web Services 账户 中创建 IAM 用户
- 适用于 Amazon 资源的访问管理
- IAM 基于身份的策略示例

步骤 3：创建访问密钥 ID 和秘密访问密钥

要进行 CLI 访问，您需要访问密钥 ID 和秘密访问密钥。如果可能，请使用临时凭证代替长期访问密钥。临时凭证包括访问密钥 ID、秘密访问密钥，以及一个指示凭证何时到期的安全令牌。有关更多信息，请参阅《Amazon 一般参考》中的管理 Amazon 访问密钥的最佳实践。

如果用户需要在 Amazon Web Services Management Console 之外与 Amazon 交互，则需要编程式访问权限。Amazon API 和 Amazon Command Line Interface 需要访问密钥。可能的话，创建临时凭证，该凭证由一个访问密钥 ID、一个秘密访问密钥和一个指示凭证何时到期的安全令牌组成。

要向用户授予编程式访问权限，请选择以下选项之一。

哪个用户需要编程式访问权限？	目的	方式
IAM	使用短期凭证签署对 Amazon CLI 或 Amazon API 的编程式请求（直接或使用 Amazon SDK）。	按照《IAM 用户指南》中将临时凭证用于 Amazon 资源中的说明进行操作。
IAM	（不推荐使用） 使用长期凭证签署对 Amazon CLI 或 Amazon API 的编程式请求（直接或使用 Amazon SDK）。	按照《IAM 用户指南》中管理 IAM 用户的访问密钥中的说明进行操作。

步骤 4：安装 Amazon CLI

Amazon CLI是一种开源工具，让您能够在命令行 Shell 中Amazon Web Services使用命令与进行交互。CAmazon SAM LI 要求Amazon CLI进行配置凭证等活动。要了解有关的更多信息Amazon CLI，请参阅什么是Amazon Command Line Interface？ 在Amazon Command Line Interface用户指南中。

要安装Amazon CLI，请参阅《Amazon Command Line Interface用户指南》中的Amazon CLI安装或更新的最新版本。

步骤 5：Amazon CLI使用配置Amazon凭证

要使用配置证书，请使用Amazon CLI

1. 从`aws configure`命令行运行此命令。
2. 配置以下内容。选择每个链接以了解更多信息：
    a. 访问密钥 ID
    b. 私有访问密钥
    c. Amazon Web Services 区域
    d. 输出格式

以下示例显示了示例值。

```
$ 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
```

将此信息Amazon CLI存储在credentials和文件中命名的配置config文件（设置集合）default中。这些文件位于主目录下的.aws文件中。预设情况下，当您运行的 Amazon CLI 命令未明确指定要使用的配置文件时，将使用此配置文件中的信息。有关该credentials文件的更多信息，请参阅Amazon Command Line Interface用户指南中的配置和凭证文件设置。

有关配置凭据（例如使用现有配置和凭证文件）的更多信息，请参阅Amazon Command Line Interface用户指南中的快速设置。

后续步骤

现在，您可以安装 CAmazon SAM LI 并开始使用了Amazon SAM。要安装Amazon SAM CLI，请参阅安装 Amazon SAM CLI (p. 15)。

安装 Amazon SAM CLI

在受支持的操作系统上安装最新版本的Amazon Serverless Application Model命令行界面 (Amazon SAMCLI)。

要了解如何管理当前安装的Amazon SAM CLI 版本，包括如何升级、卸载或管理夜间版本，请参阅管理 Amazon SAM CLI 版本 (p. 443)。

这是你第一次安装Amazon SAM CLI 吗？

在继续操作之前，请先完成上一节中的所有先决条件 (p. 13)。摘要包括：

1. 注册 Amazon 账户。
2. 创建 IAM 管理用户。
3. 创建访问密钥 ID 和秘密访问密钥。
4. 正在安装Amazon CLI.
5. 配置Amazon凭证。

主题

安装 Amazon SAM CLI

要安装 CAmazon SAM LI，请按照操作系统的说明进行操作。

Linux

x86_64 - command line installer

1. 将 Amazon SAMCLI .zip 文件下载到您选择的目录中。
2. 通过以下命令生成哈希值，验证已下载的安装程序文件的完整性和真实性：

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

   输出应类似下面的输出，类似下面的示例：

   ```
   <64-character SHA256 hash value> aws-sam-cli-linux-x86_64.zip
   ```

   将 64 个字符的 SHA-256 哈希值与 CLI 发行说明中所需的Amazon SAM CLI 版本的Amazon SAM 哈希值进行比较 GitHub。
3. 将安装文件解压缩到`sam-installation/`子目录中。

   Note

   如果您的操作系统没有内置的 unzip 命令，请使用等效命令。

   ```
   $ unzip aws-sam-cli-linux-x86_64.zip -d sam-installation
   ```
4. 安装 Amazon SAM CLI。

   ```
   $ sudo ./sam-installation/install
   ```
5. 验证安装。

   ```
   $ sam --version
   ```

   成功安装后，您应该会看到类似下面的输出，类似下面的输出，类似下面的输出：

   ```
   SAM CLI, version 1.58.0
   ```

ARM - command line installer

Note

我们建议将Amazon SAM CLI 安装到虚拟环境中，以确保故障排除时有一个干净的启动环境和隔离的环境。有关更多信息，请参阅使用以下方法将Amazon SAM CLI 安装到虚拟环境中 pip (p. 447)：

1. `pip`用于安装Amazon SAM CLI。

   ```
   $ pip install aws-sam-cli
   ```
2. 验证安装。

   ```
   $ sam --version
   ```

   成功安装后，您应该会看到类似下面的输出，类似下面的输出，类似下面的输出：

```
SAM CLI, version 1.58.0
```

Homebrew

Important

你必须已经在你的 Linux 计算机上Homebrew安装了。有关安装instructions，请参阅安装Homebrew以便与Amazon SAM CLI 一起使用 (p. 457)。

要通过以下命令来安装Amazon SAM CLIHomebrew，请运行以下命令：

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

验证安装。

```
$ sam --version
```

成功安装Amazon SAM CLI 后，您应该会看到类似下面的输出，类似下面的输出，类似下面的输出：

```
SAM CLI, version 1.58.0
```

macOS

使用其软件包安装程序或通过安装Amazon SAM CLIHomebrew。我们建议使用软件包安装程序。

使用软件包安装程序

软件包安装程序有两种安装方法可供选择：

1. GUI
2. Command line

您可以为所有用户安装，也可以只为当前用户安装。要为所有用户安装，需要超级用户授权。

安装步骤

使用以下任一选项安装Amazon SAM CLI。

GUI - All users

下载软件包安装程序

Note

如果您之前通过Homebrew或安装了Amazon SAM CLIpip，则需要先将其卸载。有关说明，请参阅 卸载Amazon SAM CLI (p. 444)。

1. 要开始安装，请将 macOS 下载pkg到您选择的目录中：
   - x86_64 (Intel) — aws-sam-cli-macos-x86_64.pkg
   - arm64 (Apple) — aws-sam-cli-macos-arm64.pkg
2. 通过以下命令生成哈希值，验证已下载的安装程序的完整性和真实性：

```
$ shasum -a 256 path-to-pkg-installer/name-of-pkg-installer

# Examples
$ shasum -a 256 /Users/myUser/Downloads/aws-sam-cli-macos-arm64.pkg
$ shasum -a 256 /Users/myUser/Downloads/aws-sam-cli-macos-x86_64.pkg
```

将您的 64 个字符的 SHA-256 哈希值与 Amazon SAMCLI 发行说明GitHub存储库中的相应值进行比较。

安装Amazon SAM命令行界面

1. 运行下载的文件并按照屏幕上的说明继续完成简介、自述和许可证步骤。
2. 对于“目标选择”，选择“为这台计算机的所有用户安装”。
3. 对于“安装类型”，选择Amazon SAM CLI 的安装位置，然后按安装。推荐的默认位置是`/usr/local/aws-sam-cli`。

   Note

   要使用sam命令调用Amazon SAM CLI，安装程序会自动在`/usr/local/bin/sam`和`/usr/local/aws-sam-cli/sam`或您选择的安装文件夹之间创建符号链接。
4. CAmazon SAM LI 将安装并显示“安装成功”消息。按“关闭”。

验证安装

- 运行以下Amazon SAM命令来验证 CLI 是否已正确安装并且您的符号链接已配置：

```
$ which sam
/usr/local/bin/sam
$ sam --version
SAM CLI, version 1.66.0
```

GUI - Current user

下载软件包安装程序

Note

如果您之前通过Homebrew或安装了Amazon SAM CLIpip，则需要先将其卸载。有关说明，请参阅 卸载Amazon SAM CLI (p. 444)。

1. 要开始安装，请将 macOS 下载pkg到您选择的目录中：
   - x86_64 (Intel) — aws-sam-cli-macos-x86_64.pkg
   - arm64 (Apple) — aws-sam-cli-macos-arm64.pkg
2. 通过以下命令生成哈希值，验证已下载的安装程序的完整性和真实性：

```
$ shasum -a 256 path-to-pkg-installer/name-of-pkg-installer

# Examples
$ shasum -a 256 /Users/myUser/Downloads/aws-sam-cli-macos-arm64.pkg
$ shasum -a 256 /Users/myUser/Downloads/aws-sam-cli-macos-x86_64.pkg
```

将您的 64 个字符的 SHA-256 哈希值与 Amazon SAMCLI 发行说明GitHub存储库中的相应值进行比较。

安装Amazon SAM命令行界面

1. 运行下载的文件并按照屏幕上的说明继续完成简介、自述和许可证步骤。
2. 对于 “目标选择”，选择 “仅为我安装”。如果没有看到此选项，请转到下一步。
3. 对于安装类型，执行以下操作：

   1. 选择Amazon SAM CLI 的安装位置。默认位置是 /usr/local/aws-sam-cli。选择您拥有写入权限的位置。要更改安装位置，请选择本地并选择您的位置。完成后按继续。
   2. 如果您在上一步中没有选择 “仅为我安装” 的选项，请选择 “更改安装位置” > “仅为我安装”，然后按 “继续”。
   3. 按 “安装”。
4. CAmazon SAM LI 将安装并显示 “安装成功” 消息。按 “关闭”。

创建符号链接

- 要使用sam命令调用Amazon SAM CLI，必须在 CAmazon SAM LI 程序和您的之间手动创建符号链接$PATH。通过修改并运行以下命令来创建符号链接：

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

  - *sudo* — 如果您的用户具有写入权限$PATH，sudo则不需要。否则，sudo 是必需的。
  - 路@@ *#* — 安装Amazon SAM CLI 程序的路径。例如，/Users/myUser/Desktop。
  - *path-to-symlink-directory*— 您的$PATH环境变量。默认位置是 /usr/local/bin。

验证安装

- 运行以下Amazon SAM命令来验证 CLI 是否已正确安装并且您的符号链接已配置：

```
$ which sam
/usr/local/bin/sam
$ sam --version
SAM CLI, version 1.66.0
```

Command line - All users

下载软件包安装程序

Note

如果您之前通过Homebrew或安装了Amazon SAM CLIpip，则需要先将其卸载。有关说明，请参阅 卸载Amazon SAM CLI (p. 444)。

1. 要开始安装，请将 macOS 下载pkg到您选择的目录中：

   - x86_64 (Intel) — aws-sam-cli-macos-x86_64.pkg
   - arm64 (Apple) — aws-sam-cli-macos-arm64.pkg
2. 通过以下命令生成哈希值，验证已下载的安装程序的完整性和真实性：

```
$ shasum -a 256 path-to-pkg-installer/name-of-pkg-installer

# Examples
$ shasum -a 256 /Users/myUser/Downloads/aws-sam-cli-macos-arm64.pkg
$ shasum -a 256 /Users/myUser/Downloads/aws-sam-cli-macos-x86_64.pkg
```

将您的 64 个字符的 SHA-256 哈希值与 Amazon SAMCLI 发行说明GitHub存储库中的相应值进行比较。

安装Amazon SAM命令行界面

- 修改并运行安装脚本：

```
$ 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

要使用sam命令调用Amazon SAM CLI，安装程序会自动在/usr/local/bin/sam和之间创建符号链接/usr/local/aws-sam-cli/sam。

验证安装

- 运行以下Amazon SAM命令来验证 CLI 是否已正确安装并且您的符号链接已配置：

```
$ which sam
/usr/local/bin/sam
$ sam --version
SAM CLI, version 1.66.0
```

Command line - Current user

下载软件包安装程序

Note

如果您之前通过Homebrew或安装了Amazon SAM CLIpip，则需要先将其卸载。有关说明，请参阅 卸载Amazon SAM CLI (p. 444)。

1. 要开始安装，请将 macOS 下载pkg到您选择的目录中：
   - x86_64 — aws-sam-cli-macos-x86_64.pkg
   - arm64 — aws-sam-cli-macos-arm64.pkg
2. 通过以下命令生成哈希值，验证已下载的安装程序的完整性和真实性：

```
$ shasum -a 256 path-to-pkg-installer/name-of-pkg-installer

# Examples
$ shasum -a 256 /Users/myUser/Downloads/aws-sam-cli-macos-arm64.pkg
$ shasum -a 256 /Users/myUser/Downloads/aws-sam-cli-macos-x86_64.pkg
```

将您的 64 个字符的 SHA-256 哈希值与 Amazon SAMCLI 发行说明GitHub存储库中的相应值进行比较。

安装Amazon SAM命令行界面

1. 确定您具有写入权限的安装目录。然后，使用模板创建xml文件并对其进行修改以反映您的安装目录。该目录必须已经存在。

例如，如果*path-to-my-directory*替换为/Users/myUser/Desktop，则将在那里安装aws-sam-cli程序文件夹。

```
<?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. 保存xml文件并通过运行以下命令验证其有效：

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

输出应显示将应用于Amazon SAM CLI 程序的首选项。

3. 运行以下命令来安装Amazon 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.
```

创建符号链接

- 要使用sam命令调用Amazon SAM CLI，必须在 CAmazon SAM LI 程序和您的之间手动创建符号链接$PATH。通过修改并运行以下命令来创建符号链接：

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

  - *sudo* — 如果您的用户具有写入权限$PATH，sudo则不需要。否则，sudo 是必需的。
  - 路@@ *#* — 安装Amazon SAM CLI 程序的路径。例如，/Users/myUser/Desktop。
  - *path-to-symlink-directory*— 您的$PATH环境变量。默认位置是 /usr/local/bin。

验证安装

- 运行以下Amazon SAM命令来验证 CLI 是否已正确安装并且您的符号链接已配置：

```
$ which sam
/usr/local/bin/sam
```

```
$ sam --version
SAM CLI, version 1.66.0
```

Homebrew

Important

您必须已在计算机上Homebrew安装。有关安装说明，请参阅安装Homebrew以便与Amazon SAM CLI 一起使用 (p. 457)。

按照以下步骤使用以下方法安装Amazon SAM CLIHomebrew：

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

验证安装：

```
$ sam --version
```

成功安装Amazon SAM CLI 后，您应该会看到类似下面的输出，类似下面的输出，类似下面的输出：

```
SAM CLI, version 1.58.0
```

Windows

Windows 安装程序 (MSI) 文件是 Windows 操作系统的软件包安装程序文件。

按照以下步骤使用 MSAmazon SAM I 文件安装 CLI。

1. 安装 64 位Amazon SAM命令行界面。

   Note

   如果你使用的是 32 位版本的 Windows，请参阅在 32 位 Windows 上安装Amazon SAM CLI (p. 464)。
2. 验证安装。

   完成安装后，通过打开新的命令提示符或 PowerShell 提示符进行验证。您应该能够`sam`从命令行调用。

   ```
   sam --version
   ```

   成功安装Amazon SAM CLI 后，您应该会看到类似下面的输出，类似下面的输出，类似下面的输出：

   ```
   SAM CLI, version 1.58.0
   ```
3. 启用长路径（仅限 Windows 10 及更高版本）。

   Important

   Amazon SAMCLI 应用程序模板存储库包含一些长文件路径，由`sam init`于 Windows 10 的MAX_PATH限制，这些路径可能会在运行时导致错误。要解决此问题，必须配置新的长路径行为。

   要启用长路径，请参阅微软 Windows 应用程序开发文档中的 Windows 10、版本 1607 及更高版本中启用长路径。
4. 安装 Git。

要使用`sam init`命令下载示例应用程序，还必须安装 Git。有关说明，请参阅安装 Git。

问题排查

如果您在安装Amazon SAM CLI 时遇到问题，请参阅安装错误 (p. 449)。

后续步骤

要了解有关Amazon SAM CLI 的更多内容并开始构建自己的无服务器应用程序的更多信息，请参阅以下内容：

- 教程：部署 Hello World 应用程序 (p. 23)— 关于以下、构建和部署基本无服务器应用程序的tep-by-step指令。
- 完整Amazon SAM研讨会 — 该研讨会旨在向您介绍所Amazon SAM提供的许多主要功能。
- Amazon SAM示例应用程序和模式 — 来自社区作者的示例应用程序和模式，您可以进一步试验。

教程：部署 Hello World 应用程序

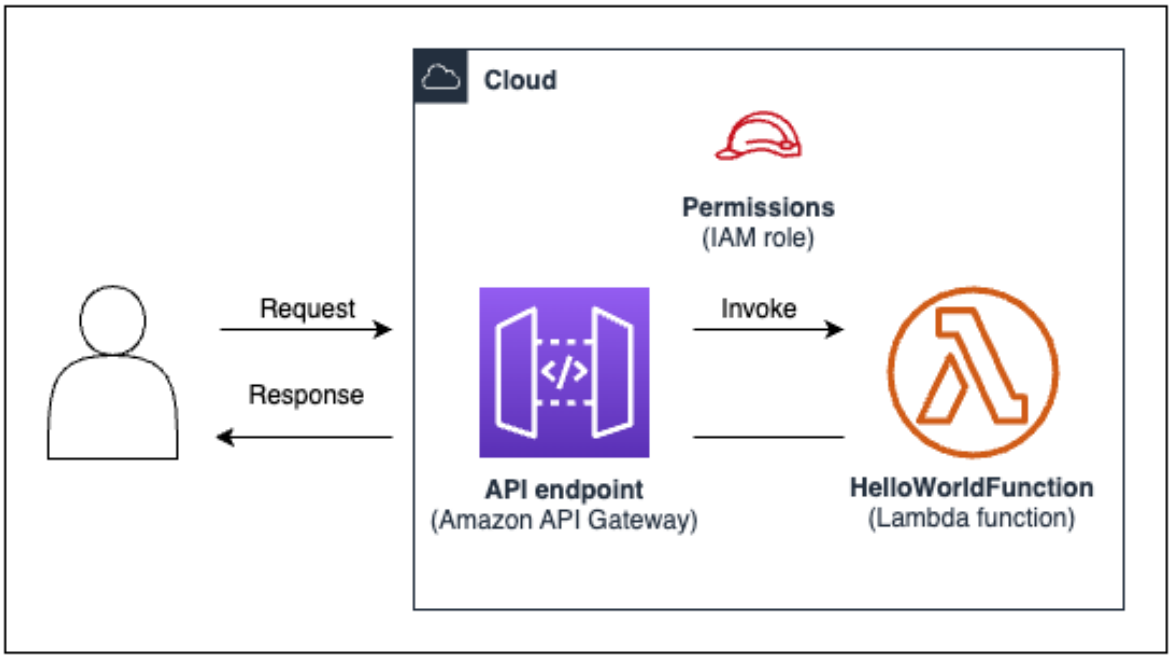
在本教程中，您将使用Amazon Serverless Application Model命令行界面 (Amazon SAMCLI) 完成以下操作：

- 初始化、构建和部署示例 Hello Worl d 应用程序。
- 进行本地更改并同步到Amazon CloudFormation。
- 在中测试您的应用程序Amazon Web Services 云。
- （可选）在您的开发主机上执行本地测试。
- 从中删除示例应用程序Amazon Web Services 云。

示例 Hello Worl d 应用程序实现了基本的 API 后端。它由以下资源组成：

- Amazon API Gateway — 您将用来调用函数的 API 终端节点。
- Amazon Lambda— 处理 HTTP API GET 请求并返回`hello world`消息的函数。
- Amazon Identity and Access Management(IAM) 角色 — 为服务提供安全交互的权限。

下图显示此应用程序的组件：



主题

先决条件

确认您是否已完成以下内容：

- Amazon SAM先决条件 (p. 13)
- 安装 Amazon SAM CLI (p. 15)

步骤 1：初始化示例 Hello World 应用程序

在此步骤中，您将使用Amazon SAM CLI 在本地计算机上创建示例 Hello Worl d 应用程序项目。

初始化示例 Hello World 应用程序

1. 在命令行中，从您选择的起始目录中，运行以下命令：

```
$ sam init
```

2. CAmazon SAM LI 将指导您初始化新应用程序。配置以下内容：

   1. 选择 “Amazon快速启动模板” 以选择起始模板。
   2. 选择 Hello World 示例模板并下载。
   3. 使用Python运行时和zip包类型。
   4. 在本教程中，选择不Amazon X-Ray跟踪。要了解更多信息，请参阅什么是Amazon X-Ray？ 在《Amazon X-Ray开发者指南》中。
   5. 在本教程中，选择不使用亚马逊 CloudWatch 应用程序洞察进行监控。要了解更多信息，请参阅《亚马逊 CloudWatch 用户指南》中的 “亚马逊 CloudWatch 应用程序见解”。
   6. 将您的应用程序命名为 sam-app。

   要使用Amazon SAM CLI 交互流程，请执行以下操作：

   - 方括号 ([ ]) 表示默认值。将答案留空以选择默认值。
   - 输入 “**y**是”，**n**输入 “否”。

   以下是sam init交互式流程示例：

```
$ 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]:

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]:

Project name [sam-app]:
```

3. CAmazon SAM LI 下载您的起始模板并在您的本地计算机上创建应用程序项目目录结构。以下是Amazon SAM CLI 输出的示例：

```
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. 在命令行中，移至新创建的sam-app目录。以下是Amazon SAM CLI 创建的示例：

```
$ 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
```

需要重点介绍的一些重要文件：

- hello_world/app.py— 包含您的 Lambda 函数代码。
- hello_world/requirements.txt— 包含您的 Lambda 函数需要的任何Python依赖关系。
- samconfig.toml— 应用程序的配置文件，用于存储Amazon SAM CLI 使用的默认参数。
- template.yaml— 包含您的应用程序基础架构代码的Amazon SAM模板。

现在，您的本地计算机上有一个完全编写的无服务器应用程序！

步骤 2：构建应用程序

在此步骤中，您将使用Amazon SAM CLI 来构建应用程序并为部署做准备。当你编译时，Amazon SAMCLI会创建一个.aws-sam目录并在那里组织你的函数依赖关系、项目代码和项目文件。

构建您的应用程序

- 在命令行中，从sam-app项目目录中，运行以下命令：

```
$ sam build
```

> **Note**
>
> 如果您的本地计算机Python上没有，请改用该sam build --use-container 命令。Amazon SAMCLI 将创建一个包含您的函数的运行时和依赖项的Docker容器。此命令需要在本地机器Docker上。要安装Docker，请参阅安装 Docker (p. 455)。

以下是Amazon SAM CLI 输出的示例：

```
$ 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
```

以下是Amazon SAM CLI 创建的.aws-sam目录的简短示例：

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

需要重点介绍的一些重要文件：

- build/HelloWorldFunction— 包含您的 Lambda 函数代码和依赖项。CAmazon SAM LI 为应用程序中的每个函数创建一个目录。
- build/template.yaml— 包含部署Amazon CloudFormation时引用的Amazon SAM模板副本。
- build.toml— 配置文件，用于存储Amazon SAM CLI 在构建和部署应用程序时引用的默认参数值。

现在，您可以向Amazon Web Services 云。

步骤 3：将应用程序部署到Amazon Web Services 云

Note

此步骤需要配置Amazon证书。有关更多信息，请参阅 Amazon SAM先决条件 (p. 13) 中的 步骤 5：Amazon CLI使用配置Amazon凭证 (p. 14)。

在此步骤中，您可以使用Amazon SAM CLI 将应用程序部署到Amazon Web Services 云。CAmazon SAM LI 将执行以下操作：

- 指导您配置应用程序设置以进行部署。
- 将应用程序文件上传到Simple Storage Service (Amazon S3)。
- 将您的Amazon SAM模板转换为Amazon CloudFormation模板。然后，它将您的模板上传到Amazon CloudFormation服务以配置您的Amazon资源。

部署 应用程序

1. 在命令行中，从sam-app项目目录中，运行以下命令：

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

2. 按照Amazon SAM CLI 交互流程配置您的应用程序设置。配置以下内容：

   1. Amazon CloudFormation堆栈名称-堆栈是可作为单个单元管理的一系列Amazon资源。要了解更多信息，请参阅Amazon CloudFormation用户指南中的使用堆栈。
   2. Amazon Web Services 区域用于将您的Amazon CloudFormation堆栈部署到。有关更多信息，请参阅 Amazon CloudFormation 用户指南中的 Amazon CloudFormation 端点。
   3. 在本教程中，选择不在部署前确认更改。
   4. 允许创建 IAM 角色 — 这允许Amazon SAM创建您的 API Gateway 资源和 Lambda 函数资源进行交互所必需的 IAM 角色。
   5. 在本教程中，选择不禁用回滚。
   6. 在未定义授权HelloWorldFunction 的情况下允许-显示此消息是因为您的 API Gateway 终端节点已配置为无需授权即可公开访问。由于这是您的 Hello World 应用程序的预期配置，请允许Amazon SAM CLI 继续。有关配置授权的更多信息，请参阅控制对 API Gateway API 的访问 (p. 330)。
   7. 将@@ 参数保存到配置文件中-这将使用您的部署首选项更新应用程序的samconfig.toml文件。
   8. 选择默认配置文件名。
   9. 选择默认配置环境。

   以下是sam deploy --guided交互流的示例：

```
$ 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]:
    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]: 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]:
```

3. CAmazon SAM LI 通过执行以下操作来部署您的应用程序：

   - CAmazon SAM LI 创建 Amazon S3 存储桶并上传您的.aws-sam目录。
   - CAmazon SAM LI 将您的Amazon SAM模板转换为Amazon CloudFormation并将其上传到Amazon CloudFormation服务。
   - Amazon CloudFormation配置您的资源。

   在部署期间，Amazon SAMCLI 会显示您的进度。下面是一个示例：

```
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-2:513423067560: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::513423067560: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-2:513423067560:function:sam-app-
HelloWorldFunction-yQDNe17r9maD
-----------------------------------------------------------------------------------------------------


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

您的应用程序现已部署并在Amazon Web Services 云! 中运行

步骤 4：运行应用程序

在此步骤中，您将向您的 API 终端节点发送 GET 请求并查看您的 Lambda 函数输出。

获取您的 API 终端节点值

1. 从Amazon SAM CLI 在上一步中显示的信息中，找到该`Outputs`部分。在本节中，找到您的`HelloWorldApi`资源以查找您的 HTTP 终端节点值。下面是一个示例：

```
---------------------------------------------------------------------------------------------------------
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. 或者，您可以使用sam list endpoints --output json命令来获取此信息。下面是一个示例：

```
$ 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']"
    ]
  }
]
```

调用函数

- 使用浏览器或命令行，向您的 API 终端节点发送 GET 请求。以下是使用 curl 命令的示例：

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

步骤 5：修改应用程序并将其同步到Amazon Web Services 云

在此步骤中，您可以使用Amazon SAM CLIsam sync --watch 命令将本地更改同步到Amazon Web Services 云。

使用 sam 同步

1. 在命令行中，从sam-app项目目录中，运行以下命令：

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

2. CAmazon SAM LI 会提示您确认您正在同步开发堆栈。由于该sam sync --watch命令会自动将本地更改实时部署到Amazon Web Services 云中，因此我们建议仅将其用于开发环境。

   Amazon SAMCLI 在开始监控本地更改之前执行初始部署。下面是一个示例：

```
$ 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::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.
```

接下来，您将修改您的 Lambda 函数代码。CAmazon SAM LI 将自动检测到此更改并将您的应用程序同步到 Amazon Web Services 云。

修改和同步您的应用程序

1. 在您选择的 IDE 中，打开该`sam-app/hello_world/app.py`文件。
2. 更改`message`并保存您的文件。以下是示例：

```
import json
...
def lambda_handler(event, context):
    ...
    return {
        "statusCode": 200,
        "body": json.dumps({
            "message": "hello everyone!",
            ...
        }),
    }
```

3. CAmazon SAM LI 会检测您的更改并将您的应用程序同步到Amazon Web Services 云。下面是一个示例：

```
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. 要验证您的更改，请再次向您的 API 终端节点发送 GET 请求。

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

步骤 6：（可选）在本地测试应用程序

Note

此步骤是可选的，因为它需要在本地计算机Docker上执行。

Important

要使用 CAmazon SAM LI 进行本地测试，必须已Docker安装和配置。有关更多信息，请参阅安装 Docker (p. 455)：

在此步骤中，您使用Amazon SAM CLIsam local 命令在本地测试您的应用程序。为实现此目的，Amazon SAMCLI 使用创建本地环境Docker。此本地环境模拟您的 Lambda 函数基于云的执行环境。

您将执行以下操作：

1. 为您的 Lambda 函数创建本地环境并调用它。
2. 在本地托管您的 HTTP API 端点托管，并使用它调用 Lambda 函数。

在本地调用您的 Lambda 函数

1. 在命令行中，从sam-app项目目录中，运行以下命令：

```
$ sam local invoke
```

2. CAmazon SAM LI 创建本地Docker容器并调用您的函数。下面是一个示例：

```
$ 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\"}"}
```

在本地托管您的 API

1. 在命令行中，从sam-app项目目录中，运行以下命令：

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

2. Amazon SAMCLI 为您的 Lambda 函数创建本地Docker容器，并创建本地 HTTP 服务器来模拟您的 API 终端节点。下面是一个示例：

```
$ 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. 使用浏览器或命令行，向本地 API 终端节点发送 GET 请求。以下是使用 curl 命令的示例：

```
$ curl http://127.0.0.1:3000/hello
{"message": "hello world"}
```

步骤 7：从Amazon Web Services 云

在此步骤中，您可以使用Amazon SAM CLIsam delete 命令从中删除您的应用程序Amazon Web Services 云。

要从... 中删除您的应用程序Amazon Web Services 云

1. 在命令行中，从sam-app项目目录中，运行以下命令：

```
$ sam delete
```

2. CAmazon SAM LI 将要求您确认。然后，它将删除您的应用程序的 Amazon S3 存储桶和Amazon CloudFormation堆栈。下面是一个示例：

```
$ 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
```

问题排查

要对Amazon SAM CLI 进行故障排除，请参阅Amazon SAMCLI 纠正 (p. 449)。

了解更多信息

要继续了解Amazon SAM，请参阅以下资源：

- 完整Amazon SAM研讨会 — 该研讨会旨在向您介绍所Amazon SAM提供的许多主要功能。
- 与 SAM 的会话 — 我们的Amazon无服务器开发者倡导者团队制作的关于使用的视频系列Amazon SAM。
- Serverless Land — 汇集了Amazon无服务器的最新信息、博客、视频、代码和学习资源的站点。

使用 Amazon Serverless Application Model（Amazon SAM）

使用Amazon Serverless Application Model (Amazon SAM) 构建和运行您的无服务器应用程序。

有关简介Amazon SAM，请参见什么是 Amazon SAM？ (p. 1)

有关使用Amazon SAM模板的信息，请参阅Amazon Serverless Application Model(Amazon SAM) 规格 (p. 77)。

主题

使用 Amazon SAM CLI

使用带有Amazon SAM模板和支持的第三方集成的Amazon Serverless Application Model命令行界面 (Amazon SAMCLI) 来构建和运行您的无服务器应用程序。

有关简介Amazon SAM，请参见什么是 Amazon SAM？ (p. 1)

主题

如何记录Amazon SAM CLI 命令

Amazon SAMCLI 命令使用以下格式记录：

- 提示-默认情况下，Linux提示会记录在案，并显示为 ($)。对于Windows特定的命令，用> 作提示。请勿在键入命令时包含提示符。
- 目录 – 当必须从特定目录执行命令时，目录名称将显示在提示符符号之前。
- 用户输入 – 您在命令行处输入的命令文本采用 **`user input`** 格式。
- 可替换文本-可变文本，例如文件名和参数，格式化为*`#####`*。在多行命令中或需要特定键盘输入的命令中，键盘输入也可显示为可替换文本。例如，*`##`*。
- 输出-作为命令响应返回的输出格式为`computer output`。

以下`sam deploy`命令和输出是一个示例：

```
$ 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`是您在命令行处输入的命令。
2. **`sam deploy --guided --template`**应按原样提供。
3. *`template.yaml`* 可以替换为你的特定文件名。
4. 输出开始于`Configuring SAM deploy`。
5. 在输出中，*`Enter`* 和 *`y`* 表示您提供的可替换值。

配置 Amazon SAM CLI

为Amazon Serverless Application Model命令行界面 (Amazon SAMCLI) 配置凭证、基本设置和项目设置。

主题

配置凭证和基本设置

使用Amazon Command Line Interface (Amazon CLI) 配置基本设置，例如Amazon凭据、默认区域名称和默认输出格式。配置完成后，您可以在Amazon SAM CLI 中使用这些设置。要了解有关更多信息，请参阅Amazon Command Line Interface用户指南：

- 配置基础知识
- 配置和凭证文件设置
- 的命名配置文件Amazon CLI
- 使用启用了 Itity Centity Centity

有关快速设置说明，请参阅步骤 5：Amazon CLI使用配置Amazon凭证 (p. 14)。

配置项目设置

您可以在配置文件中指定特定于项目的设置，例如参数值，以与Amazon SAM CLI 一起使用。

- 此配置文件采用TOML文件格式。要了解更多信息，请参阅TOML文档TOML中的。
- 默认配置文件名是samconfig.toml。
- 默认配置文件位置是项目的根目录。这与您的Amazon SAM模板文件位于同一位置。

以下是 samconfig.toml 文件的示例：

```
...
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
```

配置和使用不同的环境

环境是一种命名标识符，可用于指定项目设置的集合。

- 默认环境是[default]。
- 要创建新环境，请指定新的表头。例如，[prod]。
- 环境名称应始终是表头中的第一个条目。
- 在samconfig.toml文件中创建环境后，您可以使用--config-env选项进行指定。以下是示例：

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

配置和使用参数值

您可以为每个Amazon SAM CLI 命令指定参数值。

- 表头中的第二个条目标识了Amazon SAM CLI 命令。例如，为sam build命令[default.build]指定默认环境值。
- 当Amazon SAM CLI 命令包含子命令（例如）时sam local，使用_（下划线）。例如，[*environment*.local_invoke]。
- 当Amazon SAM CLI 命令或子命令包含-（连字符）时，将其替换为_（下划线）。例如，sam local start-api 改为 [*environment*.local_start_api]。

- 要为Amazon SAM CLI 指定参数值，请将参数条目添加到 TOML 表标题中。例如，[*environment*.*command*.parameters]。
- 要为所有命令指定参数值global，请使用代替命令。例如，[*environment*.global.parameters]。

指定参数值时：

- 每个配置条目都是一个TOML键值对。例如，在watch = false配置条目中，watch是密钥和false值。
- 配置密钥是Amazon SAM CLI 命令的长格式选项名称。
  - 有关命令和选项的列表，请参阅Amazon SAMCLI 命令参考 (p. 408)。
  - 例如，sam sync --template-file project.yaml可以按如下方式指定：

```
[default.sync.parameters]
template_file = "project.yaml"
```

- 配置值可以采用以下形式：
  - 布尔值可以是true或false。例如，confirm_changeset = true。
  - 对于采用字符串值形式的单个参数的选项，请使用" "（引号）。例如，region = "us-west-2"。
  - 对于采用参数值列表的选项，使用" "（引号）并使用 （空格）分隔每个值。例如，capabilities = "CAPABILITY_IAM CAPABILITY_NAMED_IAM"。
  - 当值包含键值对列表时，这些键值对以空格分隔，并且每对的值由编码" "（引号）包围。例如，tags = "project=\"my-application\" stage=\"production\""。
  - 对于可以多次指定的参数，该值是一个参数数组。例如，image_repositories = ["my-function-1=image-repo-1", "my-function-2=image-repo-2"]。

使用配置的值时，优先顺序如下：

- 您在命令行提供的参数值优先于配置文件和模板文件Parameters部分中的相应值。
- 如果在命令行中使用该--parameter-overrides选项，或者在配置文件中使用parameter_overrides密钥，则其值优先于模板文件Parameters部分中的值。
- 在您的配置文件中，为特定命令表提供的条目优先于[*environment*.global]部分中的条目。以下是 samconfig.toml 文件示例：

```
[default.global.parameters]
stack_name = "common-stack"

[default.deploy.parameters]
stack_name = "my-app-stack"
```

该sam deploy命令将使用堆栈名称my-app-stack。任何其他命令都将使用堆栈名称common-stack。

将Amazon SAM CLI 与配置文件一起使用

某些命令（例如sam deploy --guided）提供交互式流程，您可以使用该流程来设置配置值。以下是sam deploy --guided交互流tity Clity Centity Centy

```
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
```

在交互流程中，任何配置文件中已经存在的值将显示在[ ]（方括号）中。对于您在交互流程中提供的新值，Amazon SAMCLI 会将这些值写入您的配置文件。

在此示例中，Amazon SAMCLI 将以下内容写入您的配置文件：

```
[default.deploy]
[default.deploy.parameters]
capabilities = "CAPABILITY_IAM"
confirm_changeset = true
resolve_s3 = true
s3_bucket = "aws-sam-cli-managed-default-samclisourcebucket-1a4x26zabcde"
s3_prefix = "sam-app"
region = "us-west-2"
```

将值写入配置文件时，Amazon SAMCLI 按如下方式处理全局值：

- 如果输入的参数值存在于全局命令表中，并且特定的命令表（例如[*environment*.deploy.parameters]）Amazon SAM不将该值写入特定的命令表。
- 如果输入的参数值同时存在于全局命令表和特定命令表中，则从特定命令表中Amazon SAM删除键值对。在这里，Amazon SAM假设你想用全局命令值覆盖你的特定命令值。

使用 sam build

使用Amazon Serverless Application Model命令行接口 (Amazon SAMCLI)`sam build` 命令为无服务器应用程序做好准备，以执行开发工作流程中的后续步骤，例如本地测试或部署到Amazon Web Services 云。此命令创建一个`.aws-sam`目录，该目录以和`sam deploy`要求的格式和位置来构建`sam local`您的应用程序。

- 有关Amazon SAM CLI 的简介，请参阅什么是Amazon SAM CLI？ (p. 6)。
- 有关`sam build`命令选项的列表，请参见山姆·布莱德 (p. 408)。
- 有关在典型开发工作流程`sam build`中使用的示例，请参阅步骤 2：构建应用程序 (p. 26)。

> **Note**
>
> 使用`sam build`要求您从开发计算机上无服务器应用程序的基本组件开始。这包括Amazon SAM模板、Amazon Lambda函数代码以及任何特定语言的文件和依赖关系。要了解更多信息，请参阅 使用 sam init (p. 63)。

主题

使用 sam build 构建应用程序

在使用之前`sam build`，请考虑并配置以下内容：

1. Lambda 函数和层 — 该`sam build`命令可以构建 Lambda 函数和层。要了解有关 Lambda 层的更多信息，请参阅建筑层 (p. 354)。
2. Lambda 运行时提供在调用函数时，运行时会在执行环境中运行函数时。您可以配置原生和自定义运行时。
   a. 原生运行时 — 在支持的 Lambda 运行时中编写您的 Lambda 函数，并在中构建您的函数以使用原生 Lambda 运行时Amazon Web Services 云。
   b. 自定义运行时 — 使用任何编程语言创建 Lambda 函数，并使用在第三方构建makefile器中定义的自定义流程构建运行时，例如esbuild。要了解更多信息，请参阅 构建自定义运行时 (p. 355)。
3. Lambda 包类型 — Lambda 函数可以打包成以下 Lambda 部署包类型：
   a. .zip 文件存档-包含您的应用程序代码及其依赖项。
   b. 容器映像 — 包含基本操作系统、运行时、Lambda 扩展、应用程序代码及其依赖项。

可以在使用初始化应用程序时配置这些应用程序设置`sam init`。

- 要了解有关使用的更多信息`sam init`，请参阅使用 sam init (p. 63)。
- 要了解有关在应用程序中配置这些设置的更多信息，请参阅构建应用程序 (p. 342)。

生成应用程序

1. `cd`到你的项目的根源。这与您的Amazon SAM模板位于同一位置。

```
$ cd sam-app
```

2. 运行以下命令：

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

> **Note**
>
> 一个常用的选项是`--use-container`。要了解更多信息，请参阅 在提供的容器内构建 Lambda 函数 (p. 44)。

以下是Amazon SAM CLI 输出的示例：

```
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: python3.9 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
```

3. CAmazon SAM LI 创建.aws-sam构建目录。以下是示例：

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

根据您的应用程序的配置方式，Amazon SAMCLI 会执行以下操作：

1. 在.aws-sam/build目录中下载、安装和整理依赖关系。
2. 准备好您的 Lambda 代码。这可能包括编译您的代码、创建可执行的二进制文件和构建容器镜像。
3. 将构建构件复制到该.aws-sam目录中。格式将根据您的应用程序包类型而有所不同。
   a. 对于.zip 包类型，尚未对构件进行压缩，因此可用于本地测试。使用时，Amazon SAMCLI 会压缩您的应用程序sam deploy。
   b. 对于容器映像包类型，在本地创建容器映像并在.aws-sam/build.toml文件中引用。
4. 将Amazon SAM模板复制到.aws-sam目录中，并在必要时使用新的文件路径对其进行修改。

以下是构成.aws-sam目录中编译工件的主要组件：

- 构建目录 — 包含您的 Lambda 函数和相互独立结构的层。这会为.aws-sam/build目录中的每个函数或层生成唯一的结构。
- Amazon SAM模板-根据生成过程中的更改使用更新的值进行了修改。
- build.toml 文件 — 包含Amazon SAM CLI 使用的编译设置的配置文件。

本地测试和部署

使用sam local或使用执行本地测试时sam deploy，Amazon SAMCLI 会执行以下操作：

1. 它首先检查.aws-sam目录是否存在以及Amazon SAM模板是否位于该目录中。如果满足这些条件，Amazon SAMCLI 会将其视为您的应用程序的根目录。
2. 如果不满足这些条件，Amazon SAMCLI 会将Amazon SAM模板的原始位置视为应用程序的根目录。

开发时，如果对原始应用程序文件进行了更改，请先运行sam build以更新.aws-sam目录，然后再进行本地测试。

最佳实践

- 不要编辑该.aws-sam/build目录下的任何代码。相反，请更新项目文件夹中的原始源代码，然后运行sam build以更新.aws-sam/build目录。
- 修改原始文件时，运行sam build以更新.aws-sam/build目录。
- 您可能希望Amazon SAM CLI 引用项目的原始根目录而不是.aws-sam目录，例如在使用开发和测试时sam local。删除.aws-sam目录或.aws-sam目录中的Amazon SAM模板，让Amazon SAM CLI 将您的原始项目目录识别为根项目目录。准备就绪后，sam build再次运行以创建.aws-sam目录。
- 当你运行时sam build，.aws-sam/build目录每次都会被覆盖。该.aws-sam目录不是。如果要存储诸如日志之类的文件，请将其存储在中.aws-sam以防止它们被覆盖。

sam build 的选项

构建单个资源

提供资源的逻辑 ID 以仅构建该资源。以下是示例：

```
$ sam build HelloWorldFunction
```

要构建嵌套应用程序或堆栈的资源，请使用*<stack-logical-id>*/*<resource-logical-id>*以下格式提供应用程序或堆栈逻辑 ID 以及资源逻辑 ID：

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

在提供的容器内构建 Lambda 函数

该--use-container选项下载容器镜像并使用它来构建您的 Lambda 函数。然后在您的.aws-sam/build.toml文件中引用本地容器。

需要安装Docker此选项。有关说明，请参阅 安装 Docker (p. 455)。

以下是此命令的示例：

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

您可以通过--build-image选项指定要使用的容器映像。以下是示例：

```
$ sam build --use-container --build-image amazon/aws-sam-cli-build-image-nodejs12.x
```

要指定用于单个函数的容器镜像，请提供函数逻辑 ID。以下是示例：

```
$ sam build --use-container --build-image Function1=amazon/aws-sam-cli-build-image-
python3.8
```

将环境变量传递到构建容器

使用--container-env-var将环境变量传递到构建容器。以下是示例：

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

要传递文件中的环境变量，请使用--container-env-var-file选项。以下是示例：

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

env.json文件示例：

```
{
  "MyFunction1": {
    "GITHUB_TOKEN": "TOKEN1"
  },
  "MyFunction2": {
    "GITHUB_TOKEN": "TOKEN2"
  }
}
```

加快包含多个功能的应用程序的构建

当您在具有多个功能的应用程序sam build上运行时，Amazon SAMCLI 会逐个构建每个函数。要加快构建过程，请使用--parallel选项。这会同时构建所有的函数和层。

以下是此命令的示例：

```
$ sam build —-parallel
```

问题排查

要对Amazon SAM CLI 进行故障排除，请参阅Amazon SAMCLI 纠正 (p. 449)。

示例

构建使用本机运行时和.zip 包类型的应用程序

对于此示例，请参见教程：部署 Hello World 应用程序 (p. 23)。

构建使用原生运行时和图像包类型的应用程序

首先，我们运行sam init初始化一个新的应用程序。在交互流程中，我们选择Image包裹类型。以下是示例：

```
$ 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?
        ...
        11 - java8
        12 - nodejs18.x
        13 - nodejs16.x
        14 - nodejs14.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

...
```

CAmazon SAM LI 初始化应用程序并创建以下项目目录：

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

接下来，我们sam build开始构建我们的应用程序：

```
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
```

构建包含已编译编程语言的应用程序

在此示例中，我们使用Go运行时构建了一个包含 Lambda 函数的应用程序。

首先，我们使用以下方法初始化一个新应用程序`sam init`，并将我们的应用程序配置为使用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

...
```

CAmazon SAM LI 初始化应用程序。以下是应用程序目录结构的示例：

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

我们参考该README.md文件以了解该应用程序的要求。

```
...
## 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)
...
```

接下来，我们运行sam local invoke来测试我们的函数。此命令出错Go，因为未安装在我们的本地计算机上：

```
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"}%
```

接下来，我们sam build开始构建我们的应用程序。由于未安装在本地计算机上Go，因此我们遇到了错误：

```
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
```

虽然我们可以配置本地计算机来正确构建我们的函数，但我们改用了带的--use-container选项sam build。Amazon SAMCLI 下载容器镜像，使用原生版本构建我们的函数 GoModulesBuilder，并将生成的二进制文件复制到我们的.aws-sam/build/HelloWorldFunction目录中。

```
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
```

下面是.aws-sam目录的示例：

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

接下来，我们运行sam local invoke。我们的函数成功调用：

```
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"}%
```

了解更多信息

要了解有关使用sam build命令的更多信息，请参阅以下内容：

- 学习Amazon SAM：sam build — Serverless Land “学习Amazon SAM” 系列已上线YouTube。
- 学习Amazon SAM | sam build | E3 — Serverless Land “学习Amazon SAM” 系列已上线YouTube。
- Amazon SAM构建：它如何为部署提供构件（Sessions With SAM S2E8）—Amazon SAM 系列开启的会话YouTube。
- Amazon SAM自定义版本：如何使用 Makefile 在 SAM (S2E9) 中自定义内部版本 — 开启Amazon SAM系列的会话YouTube。

使用 sam deploy

使用Amazon Serverless Application Model命令行接口 (Amazon SAMCLI)sam deploy 命令将您的无服务器应用程序部署到Amazon Web Services 云。

- 有关Amazon SAM CLI 的简介，请参阅什么是Amazon SAM CLI？ (p. 6)。
- 有关sam deploy命令选项的列表，请参见sam deploy (p. 412)。
- 有关在典型开发工作流程sam deploy中使用的示例，请参阅步骤 3：将应用程序部署到Amazon Web Services 云 (p. 28)。

主题

先决条件

要使用`sam deploy`，请完成以下操作安装Amazon SAM CLI：

- Amazon SAM先决条件 (p. 13).
- 安装 Amazon SAM CLI (p. 15).

在使用之前`sam deploy`，我们建议您对以下内容有一个基本的了解：

- 配置 Amazon SAM CLI (p. 38).
- 使用 sam init (p. 63).
- 使用 sam build (p. 41).

使用 sam 部署应用程序

在您第一次部署无服务器应用程序时，请使用`--guided`选项。CAmazon SAM LI 将引导您通过交互式流程配置应用程序的部署设置。

使用交互式流程部署应用程序

1. 转到项目的根目录。这与您的Amazon SAM模板位于同一位置。

```
$ cd sam-app
```

2. 运行以下命令：

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

3. 在交互流程中，Amazon SAMCLI 会提示您提供配置应用程序部署设置的选项。

   方括号 (`[ ]`) 表示默认值。将答案留空以选择默认值。默认值是从以下配置文件中获取的：

   - `~/.aws/config`— 您的常规Amazon账户设置。
   - `~/.aws/credentials`— 您的Amazon账户凭证。
   - *<project>*`/samconfig.toml`— 您的项目的配置文件。

   通过回答Amazon SAM CLI 提示提供值。例如，您可以输入**y** yes、**n** n o 或字符串值。

   CAmazon SAM LI 将您的响应写入到您的项目`samconfig.toml`文件中。对于后续部署，您可以使用这些配置值`sam deploy`进行部署。要重新配置这些值，请`sam deploy --guided`再次使用或直接修改您的配置文件。

   下面是一个示例输出：

```
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. 接下来，Amazon SAMCLI 将您的应用程序部署到Amazon Web Services 云。部署期间，命令提示符中显示进度。以下是部署的主要阶段：

   - 对于将 .zip 文件存档打包到 Amazon Simple Storage Service (Amazon S3) 存储桶。Amazon LambdaAmazon SAM如有必要，Amazon SAMCLI 将创建一个新的存储桶。
   - 对于将 Lambda 函数程序包上载，Amazon SAMCLI 将映像上传到 Amazon Elastic Container Registry (Amazon ECR)。如有必要，Amazon SAMCLI 将创建一个新的存储库。
   - CAmazon SAM LI 创建Amazon CloudFormation更改集并将您的应用程序Amazon CloudFormation作为堆栈部署到。
   - CAmazon SAM LI 使用您的 Lambda 函数的新`CodeUri`值修改您部署的Amazon SAM模板。

   以下是Amazon SAM CLI 部署输出的一个示例中：

```
        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-2:513423067560: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       AWS::ApiGateway::RestA   ServerlessRestApi        Resource
 creation
                         pi                                                Initiated
CREATE_COMPLETE          AWS::ApiGateway::RestA   ServerlessRestApi        -
                         pi
CREATE_IN_PROGRESS       AWS::Lambda::Permissio   HelloWorldFunctionHell   -
                         n                        oWorldPermissionProd
CREATE_IN_PROGRESS       AWS::ApiGateway::Deplo   ServerlessRestApiDeplo   -
                         yment                    yment47fc2d5f9d
CREATE_IN_PROGRESS       AWS::Lambda::Permissio   HelloWorldFunctionHell   Resource
 creation
                         n                        oWorldPermissionProd     Initiated
CREATE_IN_PROGRESS       AWS::ApiGateway::Deplo   ServerlessRestApiDeplo   Resource
 creation
                         yment                    yment47fc2d5f9d          Initiated
CREATE_COMPLETE          AWS::ApiGateway::Deplo   ServerlessRestApiDeplo   -
                         yment                    yment47fc2d5f9d
CREATE_IN_PROGRESS       AWS::ApiGateway::Stage   ServerlessRestApiProdS   -
                                                  tage
CREATE_IN_PROGRESS       AWS::ApiGateway::Stage   ServerlessRestApiProdS   Resource
 creation
                                                  tage                     Initiated
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::513423067560: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-2:513423067560:function:sam-app-

HelloWorldFunction-XPqNX4TBu7qn

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


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

5. 要查看部署的应用程序，请执行以下操作：

   1. 使用 URL https://console.aws.amazon.com/cloudformation 直接打开Amazon CloudFormation控制台。
   2. 选择堆栈。
   3. 按应用程序名称识别您的堆栈并将其选中。

部署前验证更改

您可以将Amazon SAM CLI 配置为显示您的Amazon CloudFormation更改集并在部署之前要求确认。

在部署之前确认更改

1. 在部署期间`sam deploy --guided`，在部署之前输入**Y**以确认更改。

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

   或者，您可以使用以下内容修改您的`samconfig.toml`文件：

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

2. 在部署期间，Amazon SAMCLI 将要求您在部署之前确认更改。以下是示例：

```
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-2:513423067560:changeSet/samcli-deploy1680559234/d9f58a77-98bc-41cd-
b9f4-433a5a450d7a
Previewing CloudFormation changeset before deployment
=====================================================
Deploy this changeset? [y/N]: y
```

在部署期间指定其他参数

您可以在部署时指定要配置的其他参数值。您可以通过修改Amazon SAM模板和在部署期间配置参数值来实现此目的。

指定其他参数

1. 修改Amazon SAM模板的`Parameters`部分。以下是示例：

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

2. 运行 `sam deploy --guided`。下面是一个示例输出：

```
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
```

为您的 Lambda 函数配置代码签名

您可以在部署时为 Lambda 函数配置代码签名。您可以通过修改Amazon SAM模板和在部署期间配置代码签名来实现此目的。

配置代码签名

1. `CodeSigningConfigArn`在您的Amazon SAM模板中指定。以下是示例：

```
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. 运行 `sam deploy --guided`。CAmazon SAM LI 将提示您配置代码签名。下面是一个示例输出：

```
#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):
```

最佳实践

- 使用时`sam deploy`，Amazon SAMCLI 会部署位于`.aws-sam`目录中的应用程序构建工件。当您对应用程序的原始文件进行更改时，请在部署之前运行`sam build`以更新`.aws-sam`目录。
- 首次部署应用程序时，`sam deploy --guided`用于配置部署设置。对于后续部署，您可以使用`sam deploy`配置的设置进行部署。

sam 部署选项

以下是的常用选项`sam deploy`。有关所有选项的列表，请参阅sam deploy (p. 412)。

使用引导式交互流程部署您的应用程序

使用该--guided选项通过交互式流程配置应用程序的部署设置。以下是示例：

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

您的应用程序的部署设置保存在您的项目samconfig.toml文件中。有关 CAmazon SAM LI 设置的值的描述，请参阅将Amazon SAM CLI 与配置文件一起使用 (p. 40)。

问题排查

要对Amazon SAM CLI 进行故障排除，请参阅Amazon SAMCLI 纠正 (p. 449)。

示例

部署包含打包为.zip 文件存档的 Lambda 函数的 Hello World 应用程序

有关示例，请参见步骤 3：将应用程序部署到Amazon Web Services 云 (p. 28) Hello World 应用程序教程中的。

部署包含打包为容器映像的 Lambda 函数的 Hello World 应用程序

首先，我们sam init用来创建我们的 Hello World 应用程序。在交互流程中，我们选择Python3.9运行时和Image包类型。

```
$ 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
...
```

接下来cd，我们进入项目的根目录并运行sam build。Amazon SAMCLI 使用在本地构建我们的 Lambda 函数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
```

接下来，我们运行`sam deploy --guided`部署我们的应用程序。CAmazon SAM LI 指导我们配置部署设置。然后，Amazon SAMCLI 将我们的应用程序部署到Amazon Web Services 云。

```
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": "513423067560.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-2:513423067560: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

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       AWS::ApiGateway::RestA   ServerlessRestApi        Resource
 creation
                         pi                                                Initiated

CREATE_COMPLETE          AWS::ApiGateway::RestA   ServerlessRestApi        -
                         pi

CREATE_IN_PROGRESS       AWS::Lambda::Permissio   HelloWorldFunctionHell   -
                         n                        oWorldPermissionProd

CREATE_IN_PROGRESS       AWS::ApiGateway::Deplo   ServerlessRestApiDeplo   -
                         yment                    yment47fc2d5f9d

CREATE_IN_PROGRESS       AWS::Lambda::Permissio   HelloWorldFunctionHell   Resource
 creation
                         n                        oWorldPermissionProd     Initiated

CREATE_IN_PROGRESS       AWS::ApiGateway::Deplo   ServerlessRestApiDeplo   Resource
 creation
                         yment                    yment47fc2d5f9d          Initiated

CREATE_COMPLETE          AWS::ApiGateway::Deplo   ServerlessRestApiDeplo   -
                         yment                    yment47fc2d5f9d

CREATE_IN_PROGRESS       AWS::ApiGateway::Stage   ServerlessRestApiProdS   -
                                                  tage

CREATE_IN_PROGRESS       AWS::ApiGateway::Stage   ServerlessRestApiProdS   Resource
 creation
```

```
                                            tage                        Initiated
CREATE_COMPLETE          AWS::ApiGateway::Stage   ServerlessRestApiProdS   -
                                                  tage
CREATE_COMPLETE          AWS::Lambda::Permissio   HelloWorldFunctionHell   -
                         n                        oWorldPermissionProd
CREATE_COMPLETE          AWS::CloudFormation::S   sam-app                  -
                         tack
-------------------------------------------------------------------------------------------------

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

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

Description         Implicit IAM Role created for Hello World function

Value               arn:aws:iam::513423067560: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-2:513423067560:function:sam-app-
HelloWorldFunction-
kyg6Y2iNRUPg

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


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

了解更多信息

要了解有关使用Amazon SAM CLI`sam deploy` 命令的更多信息，请参阅以下内容：

- 完整Amazon SAM研讨会：模块 3-手动部署 — 学习如何使用Amazon SAM CLI 构建、打包和部署无服务器应用程序。

使用 sam init

Amazon Serverless Application Model命令行接口 (Amazon SAMCLI)`sam init` 命令提供初始化新的无服务器应用程序的选项，该应用程序包括：

- 用于定义基础设施代码的Amazon SAM模板。

- 用于组织应用程序的文件夹结构。
- 为您的Amazon Lambda功能进行配置。

有关Amazon SAM CLI 的简介，请参阅什么是Amazon SAM CLI？ (p. 6)。

有关Amazon SAM CLI`sam init` 命令选项的列表，请参阅sam init (p. 414)。

主题

初始化一个新的无服务器应用程序

使用Amazon SAM CLI 初始化新的无服务器应用程序

1. `cd`到起始目录。
2. 在命令行运行以下命令：

```
$ sam init
```

3. CAmazon SAM LI 将引导您完成交互式流程，创建新的无服务器应用程序。

选择一个起始模板

模板由以下内容组成：

1. 基础设施代码的Amazon SAM模板。
2. 用于组织项目文件的起始项目目录。例如，这可能包括：
   a. 您的 Lambda 函数代码及其依赖项的结构。
   b. 包含用于本地测试的测试事件`events`的文件夹。
   c. 支持单元测试`tests`的文件夹。
   d. 用于配置项目设置的`samconfig.toml`文件。
   e. `ReadMe`文件和其他基本的起始项目文件。

   以下是起始项目目录的示例：

```
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
```

您可以从可用的 “Amazon快速入门模板” 列表中进行选择，也可以提供自己的自定义模板位置。

选择Amazon快速启动模板

1. 出现提示时，选择 “Amazon快速启动模板”。
2. 首先选择一个Amazon快速入门模板。以下是示例：

```
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
```

选择自己的自定义模板位置

1. 出现提示后，选择自定义模板位置。

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

2. CAmazon SAM LI 将提示您提供模板位置。

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

   为您的模板.zip 文件存档提供以下任意位置：

   - GitHub存储库-存储库中.zip 文件的路径。GitHub该文件必须位于存储库的根目录中。
   - Mercurial存储库-存储库中.zip 文件的路径。Mercurial该文件必须位于存储库的根目录中。
   - .zip 路径 — 您的.zip 文件的 HTTPS 或本地路径。
3. CAmazon SAM LI 将使用您的自定义模板初始化您的无服务器应用程序。

选择运行时

当您选择Amazon快速启动模板时，Amazon SAMCLI 会提示您为 Lambda 函数选择运行时。Amazon SAMCLI 显示的选项列表是 Lambda 原生支持的运行时。

- 运行时提供在执行环境中运行的语言特定环境。
- 部署到时Amazon Web Services 云，Lambda 服务会在执行环境中调用您的函数。

您可以在自定义运行时中使用任何其他编程语言。为此，您需要手动创建起始应用程序结构。然后，您可以使用配置自定义模板位置`sam init`来快速初始化应用程序。

根据您的选择，Amazon SAMCLI 为您的 Lambda 函数代码和依赖项创建起始目录。

如果 Lambda 在您的运行时支持多个依赖项管理器，则系统将提示您选择首选的依赖项管理器。

选择包裹类型

当您选择Amazon快速启动模板和运行时，Amazon SAMCLI 会提示您选择包类型。包类型决定如何部署您的 Lambda 函数以与 Lambda 服务一起使用。支持的两种包类型是：

1. 容器映像 — 包含基本操作系统、运行时、Lambda 扩展、应用程序代码及其依赖项。
2. .zip 文件存档-包含您的应用程序代码及其依赖项。

要了解有关部署包类型的更多信息，请参阅Amazon Lambda开发人员指南中的 Lambda 部署包。

以下是将 Lambda 函数打包为容器镜像的应用程序的示例目录结构。CAmazon SAM LI 下载图像并在函数的目录`Dockerfile`中创建以指定图像。

```
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
```

以下是应用程序的示例目录结构，其函数打包为.zip 文件存档。

```
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
```

配置Amazon X-Ray跟踪

您可以选择激活跟Amazon X-Ray踪。要了解更多信息，请参阅什么是Amazon X-Ray？ 在《Amazon X-Ray开发者指南》中。

如果您激活，Amazon SAMCLI 将配置您的Amazon SAM模板。以下是示例：

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

使用 “亚马逊 CloudWatch 应用程序洞察” 配置监控

您可以选择使用亚马逊 CloudWatch 应用程序洞察激活监控。要了解更多信息，请参阅《亚马逊 CloudWatch 用户指南》中的 “亚马逊 CloudWatch 应用程序见解”。

如果您激活，Amazon SAMCLI 将配置您的Amazon SAM模板。以下是示例：

```
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
```

命名您的应用程序

为您的应用程序提供名称。CAmazon SAM LI 使用此名称为您的应用程序创建顶级文件夹。

sam init 的选项

以下是您可以在该`sam init`命令中使用的一些主要选项。有关所有选项的列表，请参阅sam init (p. 414)。

使用自定义模板位置初始化应用程序

使用`--location`选项并提供支持的自定义模板位置。以下是示例：

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

在没有交互流程的情况下初始化应用程序

使用该`--no-interactive`选项并在命令行中提供您的配置选项，以跳过交互流程。以下是示例：

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

问题排查

要对Amazon SAM CLI 进行故障排除，请参阅Amazon SAMCLI 纠正 (p. 449)。

示例

使用 Hello WorldAmazon Starter 模板初始化一个新的无服务器应用程序

有关此示例，请参阅 步骤 1：初始化示例 Hello World 应用程序 (p. 24)教程：部署 Hello World 应用程序。

使用自定义模板位置初始化新的无服务器应用程序

以下是为您的自定义模板提供GitHub位置的示例：

```
$ 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
```

以下是本地文件路径的示例：

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

以下是 HTTPS 可访问的路径示例：

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

了解更多信息

要了解有关使用`sam init`命令的更多信息，请参阅以下内容：

- 学习Amazon SAM：sam init —Serverless Land “学习Amazon SAM” 系列开始YouTube了。
- 构建与Amazon SAM CLI 配合使用的无服务器应用程序（使用 SAM S2E7 的会话）—Amazon SAM 系列开启的会话YouTube。

使用 sam sync

Amazon Serverless Application Model命令行界面 (Amazon SAMCLI)`sam sync` 命令提供了将本地应用程序更改快速同步到的选项Amazon Web Services 云。`sam sync`在开发应用程序时使用：

1. 自动检测本地更改并将其同步到Amazon Web Services 云。
2. 自定义将哪些本地更改同步到Amazon Web Services 云。
3. 在云端准备应用程序以进行测试和验证。

使用sam sync，您可以创建快速开发工作流程，缩短将本地更改同步到云端进行测试和验证所需的时间。

> Note
>
> 建议在开发环境中使用该sam sync命令。对于生产环境，我们建议使用sam deploy或配置持续集成和交付 (CI/CD) 管道。要了解更多信息，请参阅 部署无服务器应用程序 (p. 366)。

该sam sync命令是其中的一部分Amazon SAMAccelerate。 Amazon SAMAccelerate提供了一些工具，您可以使用这些工具来加快在中开发和测试无服务器应用程序的体验Amazon Web Services 云。

主题

自动检测本地更改并同步到Amazon Web Services 云

sam sync使用--watch选项运行，开始将应用程序同步到Amazon Web Services 云。这将可执行以下操作：

1. 生成应用程序-此过程与使用sam build命令类似。
2. 部署您的应用程序-Amazon SAM CLIAmazon CloudFormation 使用您的默认设置将您的应用程序部署到该位置。使用以下默认值：
   a. Amazon在您的.aws用户文件夹中找到凭据和常规配置设置。
   b. 在应用程序samconfig.toml文件中找到的应用程序部署设置。

   如果找不到默认值，Amazon SAMCLI 将通知您并退出同步过程。
3. 注意本地更改 —Amazon SAM CLI 保持运行并监视应用程序的本地更改。这就是该--watch选项提供的。

   默认情况下，此选项可能处于启用状态。有关默认值，请参阅您的应用程序的samconfig.toml文件。以下是 文件示例：

```
...
[default.sync]
[default.sync.parameters]
watch = true
...
```

4. 将@@ 本地更改同步到Amazon Web Services 云 — 当您进行本地更改时，Amazon SAMCLI 会检测这些更改并Amazon Web Services 云通过可用的最快方法将其同步到。根据更改的类型，可能会发生以下情况：
   a. 如果您更新的资源支持Amazon服务 API，Amazon SAMCLI 将使用它来部署您的更改。这样可以快速同步以更新您的资源Amazon Web Services 云。

b. 如果您更新的资源不支持Amazon服务 API，Amazon SAMCLI 将执行Amazon CloudFormation部署。这将更新您的整个应用程序Amazon Web Services 云。虽然没有那么快，但它确实使您不必手动启动部署。

由于该sam sync命令会自动更新您的应用程序Amazon Web Services 云，因此建议仅用于开发环境。运行时sam sync，系统将要求您确认：

```
**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
```

自定义将哪些本地更改同步到Amazon Web Services 云

提供选项以自定义将哪些本地更改同步到Amazon Web Services 云。这可以加快在云中查看本地更改以进行测试和验证所需的时间。

例如，提供仅同步代码更改的--code选项，例如Amazon Lambda函数代码。在开发过程中，如果您专门关注 Lambda 代码，这将使您的更改快速传输到云中进行测试和验证。以下是示例：

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

要仅同步特定 Lambda 函数或层的代码更改，请使用--resource-id选项。以下是示例：

```
$ sam sync --code --resource-id HelloWorldFunction --resource-id HelloWorldLayer
```

在云端准备应用程序以进行测试和验证

该sam sync命令会自动在中找到更新应用程序的最快方法Amazon Web Services 云。这可以加快您的开发和云测试工作流程。通过利用Amazon服务 API，您可以快速开发、同步和测试支持的资源。有关动手示例，请参阅 “完整Amazon SAM研讨会” 中的模块 6-Amazon SAM 加速。

sam 同步命令的选项

以下是一些可用于修改sam sync命令的主要选项。有关所有选项的列表，请参阅山姆同步 (p. 436)。

执行一次性Amazon CloudFormation部署

使用该--no-watch选项关闭自动同步。以下是示例：

```
$ sam sync --no-watch
```

CAmazon SAM LI 将执行一次性Amazon CloudFormation部署。此命令将sam build和sam deploy命令执行的操作组合在一起。

跳过初始Amazon CloudFormation部署

您可以自定义每次运行时sam sync是否需要Amazon CloudFormation部署。

- --no-skip-deploy-sync规定每次运行时sam sync都需要Amazon CloudFormation部署。这样可以确保您的本地基础架构与之同步Amazon CloudFormation，防止偏移。使用此选项确实会增加开发和测试工作流程的时间。

- 提供`--skip-deploy-sync`可选Amazon CloudFormation部署。Amazon SAMCLI 会将您的本地Amazon SAM模板与您部署的Amazon CloudFormation模板进行比较，如果未检测到更改，则会跳过初始Amazon CloudFormation部署。跳过Amazon CloudFormation部署可以节省将本地更改同步到的时间Amazon Web Services 云。

  如果未检测到任何更改，Amazon SAMCLI 仍将在以下情况下执行Amazon CloudFormation部署：

  - 如果自上次Amazon CloudFormation部署以来已经 7 天或更长时间。
  - 如果检测到大量 Lambda 函数代码更改，则Amazon CloudFormation部署成为更新应用程序的最快方法。

以下是示例：

```
$ sam sync --skip-deploy-sync
```

同步嵌套堆栈中的资源

同步嵌套堆栈中的资源

1. 使用提供根堆栈`--stack-name`。
2. 使用以下格式识别嵌套堆栈中的资源：*`nestedStackId/resourceId`*。
3. 使用提供嵌套堆栈中的资源`--resource-id`。

   以下是示例：

   ```
   $ sam sync --code --stack-name sam-app --resource-id myNestedStack/HelloWorldFunction
   ```

有关创建嵌套应用程序的更多信息，请参阅使用嵌套应用 (p. 328)。

指定要更新的特定Amazon CloudFormation堆栈

要指定要更新的特定Amazon CloudFormation堆栈，请提供`--stack-name`选项。以下是示例：

```
$ sam sync --stack-name dev-sam-app
```

问题排查

要对Amazon SAM CLI 进行故障排除，请参阅Amazon SAMCLI 纠正 (p. 449)。

示例

使用 sam 同步更新 Hello World 应用程序

在此示例中，我们首先初始化示例 Hello World 应用程序。要了解有关此应用程序的更多信息，请参阅教程：部署 Hello World 应用程序 (p. 23)。

运行`sam sync`开始生成和部署过程。

```
$ 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::513423067560: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-2:513423067560: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.
```

部署完成后，我们会修改`HelloWorldFunction`代码。CAmazon SAM LI 检测到此更改并将我们的应用程序同步到Amazon Web Services 云。由于Amazon Lambda支持Amazon服务 API，因此会执行快速同步。

```
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.
```

接下来，我们在应用程序的Amazon SAM模板中修改我们的 API 端点。我们改/hello为/helloworld。

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  HelloWorldFunction:
    ...
    Properties:
      ...
      Events:
        HelloWorld:
          Type: Api
          Properties:
            Path: /helloworld
            Method: get
```

由于Amazon API Gateway 资源不支持Amazon服务 API，因此Amazon SAM CLI 会自动执行Amazon CloudFormation部署。下面是一个示例输出：

```
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::513423067560: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-2:513423067560:function:sam-app-
HelloWorldFunction-2PlN6TPTQoco
-----------------------------------------------------------------------------------------------------------


Stack update succeeded. Sync infra completed.

Infra sync completed.
```

了解更多信息

有关所有`sam sync`选项的描述，请参见山姆同步 (p. 436)。

Amazon Serverless Application Model(Amazon SAM) 规格

您可以使用该Amazon SAM规范来定义您的无服务器应用程序。本部分提供可在Amazon SAM模板中使用的模板部分、资源类型、资源属性、数据类型、资源属性、内部函数和 API Gateway 扩展的Amazon SAM详细信息。

Amazon SAM模板是Amazon CloudFormation模板的扩展，其中包含一些额外的组件，使它们更易于使用。有关Amazon CloudFormation模板的完整参考，请参阅《Amazon CloudFormation用户指南》中的 “Amazon CloudFormation模板参考”。

主题

Amazon SAM模板剖析

网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的Amazon SAM模板文件严格遵循Amazon CloudFormation模板文件，中对此进行了介绍模板剖析中的Amazon CloudFormation用户指南. 两者之间的主要区别Amazon SAM和模板文件Amazon CloudFormation模板文件如下：

- 转换声明。该声明`Transform: AWS::Serverless-2016-10-31`是必需的Amazon SAM模板文件。此声明标识Amazon CloudFormation模板文件作为Amazon SAM模板文件。有关转换的更多信息，请参阅转换中的Amazon CloudFormation用户指南.
- “全局变量” 部分。这些区域有：`Globals`部分对是唯一的Amazon SAM. 它定义了所有无服务器函数和 API 通用的属性。所有`AWS::Serverless::Function`、`AWS::Serverless::Api`, 和`AWS::Serverless::SimpleTable`资源继承在`Globals`部分。有关本节的更多信息，请参阅Amazon SAM模板的全局变量部分 (p. 79).
- 资源部分。InAmazon SAM模板化`Resources`部分可以包含Amazon CloudFormation资源和Amazon SAM资源的费用。有关 的更多信息Amazon CloudFormation资源，请参阅Amazon资源和属性类型参考中的Amazon CloudFormation用户指南. 有关 Amazon SAM 资源的更多信息，请参阅 Amazon SAM资源和属性参考 (p. 82)。
- 参数部分。中声明的对象`Parameters`部分导致`sam deploy --guided`命令向用户显示其他提示。有关已声明对象和相应提示的示例，请参阅sam deploy (p. 412)中的Amazon SAMCLI 命令参考。

的所有其他部分Amazon SAM模板文件对应于Amazon CloudFormation同名的模板文件部分。

YAML

以下示例显示 YAML 格式的模板片段。

```
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
```

模板部分

Amazon SAM模板可以包含几个主要部分。只有`Transform`和`Resources`部分是必填的。

您可以按任意顺序包含模板部分。但是，在您构建模板时，使用以下列表中显示的逻辑顺序可能会很有用。这是因为一个部分中的值可能是指前一节中的值。

转换（必填）

适用于Amazon SAM模板，您必须将此部分的值包含为`AWS::Serverless-2016-10-31`.

其他转换是可选的。有关转换的更多信息，请参阅转换中的Amazon CloudFormation用户指南.

全局变量（可选）(p. 79)

所有无服务器函数、API 和简单表都通用的属性。所有`AWS::Serverless::Function`、`AWS::Serverless::Api`,和`AWS::Serverless::SimpleTable`资源继承在`Globals`部分。

本节对是唯一的Amazon SAM. 中没有相应的部分Amazon CloudFormation模板。

Description (可选)

一个描述模板的文本字符串。

本节直接对应于`Description`部分Amazon CloudFormation模板。

元数据（可选）

提供有关模板的其他信息的对象。

本节直接对应于`Metadata`部分Amazon CloudFormation模板。

Parameters（可选）

要在运行时 (创建或更新堆栈时) 传递到模板的值。您可引用模板的 `Resources` 和 `Outputs` 部分中的参数。

使用`--parameter-overrides`的参数`sam deploy`命令和配置文件中的条目-优先于Amazon SAM模板文件。有关的更多信息`sam deploy`命令，请参阅sam deploy (p. 412)中的Amazon SAMCLI 命令参考。有关配置文件的更多信息，请参阅Amazon SAM CLI 配置文件 (p. 441)。

Mappings（可选）

可用来指定条件参数值的密钥和关键值的映射，与查找表类似。您可以使用Fn::FindInMap中的内部函数Resources和Outputs部分。

本节直接对应于Mappings部分Amazon CloudFormation模板。

条件（可选）

用于控制是否创建某些资源或者是否在堆栈创建或更新过程中为某些资源属性分配值的条件。例如，您可以根据堆栈是用于生产环境还是用于测试环境来按照条件创建资源。

本节直接对应于Conditions部分Amazon CloudFormation模板。

Resources（必需）

堆栈资源及其属性，如 Amazon Elastic Compute Cloud (Amazon EC2) 实例或 Amazon Simple Storage Service (Amazon S3) 存储桶。您可引用模板的 Resources 和 Outputs 部分中的资源。

此部分类似于Resources部分Amazon CloudFormation模板。InAmazon SAM模板，此部分可以包含Amazon SAM除此之外的资源Amazon CloudFormation资源的费用。

Outputs（可选）

每当您查看堆栈的属性时返回的值。例如，您可以声明 S3 存储桶名称的输出，然后调用aws cloudformation describe-stacks Amazon Command Line Interface(Amazon CLI) 命令来查看该名称。

本节直接对应于Outputs部分Amazon CloudFormation模板。

后续步骤

下载和部署包含Amazon SAM模板文件，请参阅Amazon SAM 入门 (p. 13)然后按照中的说明操作教程：部署 Hello World 应用程序 (p. 23).

Amazon SAM模板的全局变量部分

有时，您在Amazon SAM模板中声明的资源具有通用配置。例如，您的应用程序可能包含多个具有相同Runtime、MemoryVPCConfigEnvironment、和Cors配置AWS::Serverless::Function的资源。与其在每个资源中重复这些信息，不如在本Globals节中声明一次，然后让您的资源继承它们。

该Globals部分
由AWS::Serverless::Function、AWS::Serverless::ApiAWS::Serverless::HttpApi、和AWS::Serverless::SimpleTable资源支持。

示例：

```
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
```

在此示例中，HelloWorldFunction和ThumbnailFunction使用 “nodejs12.x” 表示Runtime，“180” 秒，使用 “index.handler” 表示Handler。Timeout HelloWorldFunction除了继承的 TABLE_NAME 之外，还添加了 MESSAGE 环境变量。 ThumbnailFunction继承所有Globals属性并添加 API 事件源。

支持的资源和属性

Amazon SAM支持以下资源和属性。

```
Globals:
  Function:
    Handler:
    Runtime:
    CodeUri:
    DeadLetterQueue:
    Description:
    MemorySize:
    Timeout:
    VpcConfig:
    Environment:
    Tags:
    Tracing:
    KmsKeyArn:
    Layers:
    AutoPublishAlias:
    DeploymentPreference:
    PermissionsBoundary:
    ReservedConcurrentExecutions:
    ProvisionedConcurrencyConfig:
    AssumeRolePolicyDocument:
    EventInvokeConfig:
    Architectures:
    EphemeralStorage:

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

```
    Domain:

  HttpApi:
    Auth:
    AccessLogSettings:
    StageVariables:
    Tags:

  SimpleTable:
    SSESpecification:
```

Note

不支持任何未包含在上式列表中的资源和属性。不支持它们的一些原因包括：1) 它们会带来潜在的安全问题，或者 2) 它们使模板难以理解。

隐式展示

Amazon SAM当您在Events部分中声明 API 时，会创建隐式 API。你可以用Globals来覆盖隐式 API 的所有属性。

可覆盖

资源可以覆盖您在该Globals部分中声明的属性。例如，您可以向环境变量映射添加新变量，也可以重写全局声明的变量。但是资源无法删除该Globals部分中指定的属性。

更笼统地说，该Globals部分声明了您的所有资源共享的属性。有些资源可以为全局声明的属性提供新值，但它们无法将其删除。如果某些资源使用某个属性而其他资源没有，则您不得Globals在本节中声明它们。

下面几节介绍覆盖（overriding）如何处理不同的数据类型。

原始数据类型被替换

原始数据类型包括字符串、数字、布尔式等。

该Resources部分中指定的值将替换该Globals部分中的值。

示例：

```
Globals:
  Function:
    Runtime: nodejs12.x

Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      Runtime: python3.9
```

fRuntime or 设置MyFunction为python3.9。

地图已合并

地图也称为字典或键值对的集合。

该Resources部分中的地图条目与全局地图条目合并。如果存在重复项，则小Resource节条目将覆盖该分Globals区条目。

示例：

```
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
```

的环境变量设置为MyFunction以下内容：

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

列表是累加性的

列表也称为数组。

该Globals部分中的列表条目位于该Resources部分的列表之前。

示例：

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

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

fSecurityGroupIdsMyFunction or 设置为VpcConfig以下内容：

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

Amazon SAM资源和属性参考

本节包含Amazon SAM资源和属性类型的参考信息。

主题

- AWS::Serverless::Application (p. 116)
- AWS::Serverless::Connector (p. 120)
- AWS::Serverless::Function (p. 129)
- AWS::Serverless::HttpApi (p. 208)
- AWS::Serverless::LayerVersion (p. 228)
- AWS::Serverless::SimpleTable (p. 232)
- AWS::Serverless::StateMachine (p. 235)

AWS::Serverless::Api

创建可通过 HTTPS 终端节点调用的Amazon API Gateway 资源和方法的集合。

无需将AWS::Serverless::Api (p. 83)资源明确添加到Amazon无服务器应用程序定义模板中。这种类型的资源是通过在模板中定义的、不引用资源的AWS::Serverless::Function (p. 129)资源上定义的 Api 事件的并AWS::Serverless::Api (p. 83)集隐式创建的。

应使用AWS::Serverless::Api (p. 83) OpenApi资源来定义和记录 API，这样可以提供更多配置底层 Amazon API Gateway 资源的能力。

我们建议您使用Amazon CloudFormation挂钩或 IAM 策略来验证 API Gateway 资源是否附加了授权者以控制对它们的访问权限。

有关使用Amazon CloudFormation挂钩的更多信息，请参阅 Amazon CloudFormationCLI 用户指南和apigw-enforce-authorizer GitHub 存储库中的注册挂钩。

有关使用 IAM 策略的更多信息，请参阅 API Gateway 开发人员指南中的要求 API 路由获得授权。

> Note
>
> 当您部署到时Amazon CloudFormation，Amazon SAM会将您的Amazon SAM资源转换为Amazon CloudFormation资源。有关更多信息，请参阅Amazon CloudFormation (p. 261)：

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

```
Type: AWS::Serverless::Api
Properties:
  AccessLogSetting: AccessLogSetting
  AlwaysDeploy (p. 84): Boolean
  ApiKeySourceType: String
  Auth: ApiAuth (p. 92)
  BinaryMediaTypes: List
  CacheClusterEnabled: Boolean
  CacheClusterSize: String
  CanarySetting: CanarySetting
  Cors: String | CorsConfiguration (p. 109)
  DefinitionBody: JSON
  DefinitionUri: String | ApiDefinition (p. 108)
  Description: String
  DisableExecuteApiEndpoint: Boolean
  Domain: DomainConfiguration (p. 111)
  EndpointConfiguration: EndpointConfiguration (p. 115)
  FailOnWarnings: Boolean
  GatewayResponses: Map
  MergeDefinitions (p. 87): Boolean
```

```
  MethodSettings: MethodSettings (p. 87)
  MinimumCompressionSize: Integer
  Mode: String
  Models: Map
  Name: String
  OpenApiVersion: String
  StageName: String
  Tags: Map
  TracingEnabled: Boolean
  Variables: Map
```

属性

`AccessLogSetting`

为阶段配置访问日志设置。

类型：AccessLogSetting

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::ApiGateway::Stage`资源的`AccessLogSetting`属性。

`AlwaysDeploy`

即使未检测到 API 的更改，也要始终部署 API。

类型：布尔值

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`ApiKeySourceType`

用于根据使用计划对请求进行计量的 API 键的源。有效值为 `HEADER` 和 `AUTHORIZER`。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::ApiGateway::RestApi`资源的`ApiKeySourceType`属性。

`Auth`

配置授权以控制对 API Gateway API 的访问。

有关使用配置访问权限的更多信息，Amazon SAM请参阅控制对 API Gateway API 的访问 (p. 330)。

类型：ApiAuth (p. 92)

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`BinaryMediaTypes`

您的 API 可能返回的 MIME 类型列表。使用它来启用对 API 的二进制支持。在 mime 类型中使用 ~1 代替/。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性类似于AWS::ApiGateway::RestApi资源的BinaryMediaTypes属性。的列表 BinaryMediaTypes 已添加到Amazon CloudFormation资源和 OpenAPI 文档中。

CacheClusterEnabled

指示是否为阶段启用缓存。要缓存响应，还必须将设置CachingEnabled为 untrue derMethodSettings。

类型：布尔值

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::ApiGateway::Stage资源的CacheClusterEnabled属性。

CacheClusterSize

阶段的缓存群集大小。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::ApiGateway::Stage资源的CacheClusterSize属性。

CanarySetting

将金丝雀设置配置为常规部署阶段。

类型：CanarySetting

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::ApiGateway::Stage资源的CanarySetting属性。

Cors

管理所有 API Gateway API 的跨域资源共享 (CORS)。将允许的域指定为字符串或使用其他 Cors 配置指定字典。注意：CORSAmazon SAM 需要修改你的 OpenAPI 定义。因此，只有使用定义行内时 OpenApi ，它才有效 DefinitionBody。

有关 CORS 的更多信息，请参阅 API Gateway 开发者指南中的 API Gateway REST API 资源启用 CORS。

类型：字符串 | CorsConfiguration (p. 109)

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

DefinitionBody

描述你的 API 的 OpenAPI 规范。如果DefinitionUri既未指定DefinitionBody也未指定，SAM 将根据您的模板配置DefinitionBody为您生成。

类型：JSON

必需：否

Amazon CloudFormation兼容性：此属性类似于AWS::ApiGateway::RestApi资源的Body属性。如果提供了某些属性，则可以在传递到 DefinitionBody 之前将内容插入或修改到中 CloudFormation。相应的属性包括AuthBinaryMediaTypesCors、GatewayResponses、Models、、和 ApiEventSource 类型的属性AWS::Serverless::Function。

`DefinitionUri`

定义 API 的 OpenAPI 文档的 Amazon S3 URI、本地文件路径或位置对象。此属性引用的Amazon S3 对象必须是有效的 OpenAPI 文件。如果DefinitionUri既未指定DefinitionBody也未指定，SAM 将根据您的模板配置DefinitionBody为您生成。

如果提供了本地文件路径，则模板必须经过包含sam deploy或sam package命令的工作流程，才能正确转换定义。

引用的外部 OpenApi 文件不支持内部函数DefinitionUri。改为使用带有 Include Transfor m 的DefinitionBody属性将 OpenApi 定义导入到模板中。

类型：字符串 | ApiDefinition (p. 108)

必需：否

Amazon CloudFormation兼容性：此属性类似于AWS::ApiGateway::RestApi资源的BodyS3Location属性。嵌套的 Amazon S3 属性的命名方式不同。

`Description`

对 Api 资源的描述。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::ApiGateway::RestApi资源的Description属性。

`DisableExecuteApiEndpoint`

指定客户端是否可以使用默认 execute-api 端点调用您的 API。默认情况下，客户端可以使用默认设置调用您的 APIhttps://{api_id}.execute-api.{region}.amazonaws.com。如果要求客户端使用自定义域名来调用 API，请指定 True。

类型：布尔值

必需：否

Amazon CloudFormation兼容性：此属性类似于AWS::ApiGateway::RestApi资源的DisableExecuteApiEndpoint属性。它直接传递给x-amazon-apigateway-endpoint-configuration扩展的disableExecuteApiEndpoint属性，扩展名被添加到AWS::ApiGateway::RestApi资源的Body属性中。

`Domain`

为此 API Gateway API 配置自定义域。

类型：DomainConfiguration (p. 111)

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`EndpointConfiguration`

REST API 的终端节点类型。

类型：EndpointConfiguration (p. 115)

必需：否

Amazon CloudFormation兼容性：此属性类似于AWS::ApiGateway::RestApi资源的EndpointConfiguration属性。嵌套配置属性的命名不同。

FailOnWarnings

指定在遇到警告时回滚 API 创建 (true) 或不回滚 API 创建 (false)。默认值为 false。

类型：布尔值

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::ApiGateway::RestApi资源的FailOnWarnings属性。

GatewayResponses

为 API 配置网关响应。网关响应是 API Gateway 直接返回或通过 Lambda 授权者返回的响应。有关更多信息，请参阅网关响应的 Api Gateway OpenApi 扩展文档。

类型：地图

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

MergeDefinitions

Amazon SAM从您的 API 事件源生成OpenAPI规范。指定将其Amazon SAM合并true到AWS::Serverless::Api资源中定义的行内OpenAPI规范中。指定false不合并。

MergeDefinitions需要定义AWS::Serverless::Api的DefinitionBody属性。MergeDefinitions与的DefinitionUri属性不兼容AWS::Serverless::Api。

默认值：false

类型：布尔值

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

MethodSettings

配置 API 阶段的所有设置，包括日志、指标、cachetTL、限制。

类型：清单 MethodSetting

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::ApiGateway::Stage资源的MethodSettings属性。

MinimumCompressionSize

允许根据客户端的 Accept-Encoding 标头压缩响应正文。当响应体大小大于或等于您配置的阈值时，会触发压缩。最大正文大小阈值为 10 MB（10,485,760 字节）。-支持以下压缩类型：gzip、deflate 和身份。

类型：整数

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::ApiGateway::RestApi资源的MinimumCompressionSize属性。

Mode

此属性仅在您使用 OpenAPI 定义 REST API 时才适用。Mode 确定 API Gateway 如何处理资源更新。有关更多信息，请参阅AWS::ApiGateway::RestApi资源类型的模式属性。

有效值：overwrite 或 merge

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::ApiGateway::RestApi资源的Mode属性。

Models

您的 API 方法要使用的架构。这些架构可以使用 JSON 或 YAML 进行描述。有关示例模型，请参阅本页底部的示例部分。

类型：地图

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

Name

API Gateway RestApi 资源的名称

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::ApiGateway::RestApi资源的Name属性。

OpenApiVersion

OpenApi 要使用的版本。这可以2.0用于 Swagger 规范，也可以是 OpenApi 3.0 版本中的一个，比如3.0.1。有关 OpenAPI 的更多信息，请参阅 OpenAPI 规范。

> **Note**
>
> Amazon SAM创建一个Stage默认情况下名为的阶段。将此属性设置为任何有效值都将阻止舞台的创建Stage。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

StageName

阶段的名称，API Gateway 将它用作调用的统一资源标识符 (URI) 中的第一个路径分段。

要引用舞台资源，请使用*<api-logical-id>*.Stage。有关引用指定资源时生成的AWS::Serverless::Api (p. 83)资源的更多信息，请参阅Amazon CloudFormation指定 Amazon# Serverless# Api 时生成的资源 (p. 263)。有关生成的Amazon CloudFormation资源的一般信息，请参阅Amazon CloudFormation (p. 261)。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性类似于AWS::ApiGateway::Stage资源的StageName属性。它在 SAM 中是必需的，但在 API Gateway 中不是必需的

附加说明：隐式 API 的阶段名为 “Prod”。

Tags

一个映射 (字符串到字符串映射)，指定要添加到此 API Gateway 阶段的标签。有关标签的有效密钥和值的详细信息，请参阅Amazon CloudFormation用户指南中的资源标签。

类型：地图

必需：否

Amazon CloudFormation兼容性：此属性类似于AWS::ApiGateway::Stage资源的Tags属性。SAM 中的 Tags 属性由 Key: Value 对组成； CloudFormation 它由标签对象列表组成。

TracingEnabled

指示是否为阶段启用通过 X-Ray 进行的主动跟踪。有关 X-Ray 的更多信息，请参阅 API Gateway 开发人员指南中的使用 X-Ray 跟踪用户对 REST API 的请求。

类型：布尔值

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::ApiGateway::Stage资源的TracingEnabled属性。

Variables

一个定义阶段变量的映射 (字符串到字符串映射)，其中变量名作为键，变量值作为值。变量名称只能包含字母数字字符。值必须匹配以下正则表达式：[A-Za-z0-9._~:/?#&=,-]+。

类型：地图

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::ApiGateway::Stage资源的Variables属性。

返回值

Ref

当向Ref内部函数提供此资源的逻辑 ID 时，它将返回底层 API Gateway API 的 ID。

有关使用该Ref函数的更多信息，请参阅Ref《Amazon CloudFormation用户指南》。

Fn::GetAtt

Fn::GetAtt 返回一个此类型指定属性的值。以下为可用属性和示例返回值。

有关使用的更多信息Fn::GetAtt，请参阅Fn::GetAttAmazon CloudFormation用户指南中的。

RootResourceId

RestApi 资源的根资源 ID，例如 a0bc123d4e。

示例

SimpleApiExample

一个 Hello WorldAmazon SAM 模板文件，其中包含带有 API 终端节点的 Lambda 函数。这是可运行的无服务器应用程序的完整Amazon SAM模板文件。

YAML

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

ApiCorsExample

一个Amazon SAM模板片段，其中包含在外部 Swagger 文件中定义的 API 以及 Lambda 集成和 CORS 配置。这只是显示AWS::Serverless::Api (p. 83)定义的Amazon SAM模板文件的一部分。

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

带有使用 Amazon Cognito 授权针对该 API 的请求的 API 的Amazon SAM模板片段。这只是显示AWS::Serverless::Api (p. 83)定义的Amazon SAM模板文件的一部分。

YAML

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

ApiModelsExample

包含包含模型架构的 API 的Amazon SAM模板片段。这只是Amazon SAM模板文件的一部分，显示了包含两个模型架构的AWS::Serverless::Api (p. 83)定义。

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
```

缓存示例

一个 Hello WorldAmazon SAM 模板文件，其中包含带有 API 终端节点的 Lambda 函数。该 API 为一种资源和方法启用了缓存。有关缓存的更多信息，请参阅 API Gateway 开发人员指南中的启用 API 缓存以增强响应能力。

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
      CacheClusterEnabled: true
      CacheClusterSize: '0.5'
      MethodSettings:
        - ResourcePath: /
          HttpMethod: GET
          CachingEnabled: true
          CacheTtlInSeconds: 300

  ApiFunction: # Adds a GET api endpoint at "/" to the ApiGatewayApi via an Api event
    Type: AWS::Serverless::Function
    Properties:
      Events:
        ApiEvent:
          Type: Api
          Properties:
            Path: /
            Method: get
            RestApiId:
              Ref: ApiGatewayApi
      Runtime: python3.7
      Handler: index.handler
      InlineCode: |
        def handler(event, context):
            return {'body': 'Hello World!', 'statusCode': 200}
```

ApiAuth

配置授权以控制对 API Gateway API 的访问。

有关使用配置访问权限的更多信息和示例，Amazon SAM请参阅控制对 API Gateway API 的访问 (p. 330)。

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

```
  AddDefaultAuthorizerToCorsPreflight: Boolean
  ApiKeyRequired: Boolean
  Authorizers: CognitoAuthorizer (p. 97) | LambdaTokenAuthorizer (p. 102)
| LambdaRequestAuthorizer (p. 99)
  DefaultAuthorizer: String
  InvokeRole: String
  ResourcePolicy: ResourcePolicyStatement (p. 106)
  UsagePlan: ApiUsagePlan (p. 95)
```

属性

`AddDefaultAuthorizerToCorsPreflight`

如果设置了`DefaultAuthorizer`和`Cors`属性，则设置`AddDefaultAuthorizerToCorsPreflight`将导致将默认授权方添加到 OpenAPI 部分中的`Options`属性中。

类型：布尔值

必填项：否

默认值：True

Amazon CloudFormation兼容性：此属性是独一无二的Amazon SAM，没有Amazon CloudFormation等效属性。

`ApiKeyRequired`

如果设置为 true，则所有 API 事件都需要 API 键。有关 API 密钥的更多信息，请参阅 API Gate way 开发者指南中的使用 API 密钥创建和使用使用计划。

类型：布尔值

必填项：否

Amazon CloudFormation兼容性：此属性是独一无二的Amazon SAM，没有Amazon CloudFormation等效属性。

`Authorizers`

授权方用于控制对 API Gateway API 的访问。

有关更多信息，请参阅 控制对 API Gateway API 的访问 (p. 330)。

类型：CognitoAuthorizer (p. 97)| LambdaTokenAuthorizer (p. 102)| LambdaRequestAuthorizer (p. 99)

必填项：否

默认值：None（无）

Amazon CloudFormation兼容性：此属性是独一无二的Amazon SAM，没有Amazon CloudFormation等效属性。

其他注意事项：SAM 将授权方添加到 Api 的 OpenApi 定义中。

`DefaultAuthorizer`

为 API Gateway API 指定默认授权方，默认情况下，该授权方将用于授权 API 调用。

> **Note**
>
> 如果将与此 API EventSource 关联的函数的 Api 配置为使用 IAM 权限，则必须将此属性设置为`AWS_IAM`，否则会出现错误。

类型：字符串

必填项：否

默认值：None（无）

Amazon CloudFormation兼容性：此属性是独一无二的Amazon SAM，没有Amazon CloudFormation等效属性。

`InvokeRole`

将所有资源和方法的集成凭证设置为此值。

`CALLER_CREDENTIALS`映射到`arn:aws:iam::*:user/*`，它使用调用者凭证调用端点。

有效值：`CALLER_CREDENTIALS`、`NONE`、`IAMRoleArn`

类型：字符串

必填项：否

默认值：CALLER_CREDENTIALS

Amazon CloudFormation兼容性：此属性是独一无二的Amazon SAM，没有Amazon CloudFormation等效属性。

ResourcePolicy

为 API 上的所有方法和路径配置资源策略。

类型：ResourcePolicyStatement (p. 106)

必填项：否

Amazon CloudFormation兼容性：此属性是独一无二的Amazon SAM，没有Amazon CloudFormation等效属性。

其他注意事项：也可以AWS::Serverless::Function使用在个人身上定义此设置ApiFunctionAuth (p. 154)。对于使用的 API 来说，这是必需的EndpointConfiguration: PRIVATE。

UsagePlan

配置与此 API 关联的使用计划。有关使用计划的更多信息，请参阅 API Gate way 开发者指南中的使用 API 密钥创建和使用使用计划。

设置此Amazon SAM属性后，此属性会生成三个额外Amazon CloudFormation资源：an AWS::ApiGateway::UsagePlanAWS::ApiGateway::UsagePlanKey、an 和AWS::ApiGateway::ApiKey。有关此方案的信息，请参阅指定了 UsagePlan 属性 (p. 263)。有关生成的Amazon CloudFormation资源的一般信息，请参阅Amazon CloudFormation (p. 261)。

类型：ApiUsagePlan (p. 95)

必填项：否

Amazon CloudFormation兼容性：此属性是独一无二的Amazon SAM，没有Amazon CloudFormation等效属性。

示例

CognitoAuth

Cognito 身份验证示例

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

为 API Gateway API 配置使用计划。有关使用计划的更多信息，请参阅创建和使用带 API 密钥的使用计划中的API Gateway 开发人员指南.

语法

要在您的Amazon Serverless Application Model(Amazon SAM) 模板，请使用以下语法。

YAML

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

属性

`CreateUsagePlan`

确定如何配置此使用计划。有效值包括 PER_API、SHARED 和 NONE。

PER_API创建AWS::ApiGateway::UsagePlan、AWS::ApiGateway::ApiKey, 和AWS::ApiGateway::UsagePlanKey特定于此 API 的资源。这些资源的逻辑 ID 为*<api-logical-id>*UsagePlan、*<api-logical-id>*ApiKey, 和*<api-logical-id>*UsagePlanKey，。

SHARED创建AWS::ApiGateway::UsagePlan、AWS::ApiGateway::ApiKey, 和AWS::ApiGateway::UsagePlanKey在任何同时具有的 API 之间共享的资源CreateUsagePlan: SHARED在相同的Amazon SAM模板。这些资源的逻辑 ID 为ServerlessUsagePlan、ServerlessApiKey, 和ServerlessUsagePlanKey，。如果您使用此选项，我们建议您仅在一个 API 资源上为此使用计划添加其他配置，以避免定义冲突和状态不确定。

NONE禁用使用计划与此 API 的创建或关联。只有在这种情况下才需要SHARED要么PER_API已在Amazon SAM模板的全局变量部分 (p. 79).

有效值：PER_API、SHARED 和 NONE

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM而且没有Amazon CloudFormation等效项。

`Description`

使用计划的描述。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`Description`一个的财产`AWS::ApiGateway::UsagePlan`资源。

`Quota`

配置用户可在指定时间间隔内发出的请求的数量。

类型：QuotaSettings

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`Quota`一个的财产`AWS::ApiGateway::UsagePlan`资源。

`Tags`

与使用计划关联的任意标签（键值对）的数组。

此属性使用CloudFormation 标签类型.

类型：List

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`Tags`一个的财产`AWS::ApiGateway::UsagePlan`资源。

`Throttle`

配置整体请求速率（每秒平均请求数）和突发容量。

类型：ThrottleSettings

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`Throttle`一个的财产`AWS::ApiGateway::UsagePlan`资源。

`UsagePlanName`

使用计划的名称。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`UsagePlanName`一个的财产`AWS::ApiGateway::UsagePlan`资源。

示例

UsagePlan

以下是使用计划示例。

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

定义 Amazon Cognito 用户池授权方。

有关更多信息以及示例，请参阅 控制对 API Gateway API 的访问 (p. 330)。

语法

要在您的中声明此实体Amazon Serverless Application Model(Amazon SAM) 模板，请使用以下语法。

YAML

```
AuthorizationScopes: List
Identity: CognitoAuthorizationIdentity (p. 98)
UserPoolArn: String
```

属性

`AuthorizationScopes`

此授权者的授权范围列表。

类型：List

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`Identity`

此属性可用于指定`IdentitySource`在收到的授权人请求中。

类型：Cognito 授权身份 (p. 98)

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`UserPoolArn`

可以参用用户池/指定要添加本认知授权器的userpool arn

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

示例

CognitoAuth

Cognito 身份验证示例

YAML

```
Auth:
  Authorizers:
    MyCognitoAuth:
      AuthorizationScopes:
        - scope1
        - scope2
      UserPoolArn:
        Fn::GetAtt:
          - MyCognitoUserPool
          - Arn
      Identity:
        Header: MyAuthorizationHeader
        ValidationExpression: myauthvalidationexpression
```

CognitoAuthorizationIdentity

此属性可用于在授权者的传入请求中指定 IdentitySource。有关的更多信息 IdentitySource 参阅ApiGateway 授权者 OpenAPI 扩展.

语法

要在您的中声明此实体Amazon Serverless Application Model(Amazon SAM) 模板，请使用以下语法。

YAML

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

属性

`Header`

在 OpenAPI 定义中指定授权的标头名称。

类型：字符串

必需：否

默认值：授权

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`ReauthorizeEvery`

生存时间 (TTL) 期间（以秒为单位），用于指定 API Gateway 缓存授权方结果的时长。如果您指定一个大于 0 的值，API Gateway 将缓存授权方响应。默认情况下，API Gateway 将此属性设为 300。最大值为 3600 秒（1 小时）。

类型：整数

必需：否

默认值：300

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`ValidationExpression`

指定验证表达式以验证传入身份

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

示例

CogntoAuth 身份

YAML

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

LambdaRequestAuthorizer

配置 Lambda 授权方以通过 Lambda 函数控制对 API 的访问。

有关更多信息以及示例，请参阅 控制对 API Gateway API 的访问 (p. 330)。

语法

要在模板中声明此实体，请使用以下语法。Amazon Serverless Application ModelAmazon SAM

YAML

```
AuthorizationScopes: List
DisableFunctionDefaultPermissions (p. 100): Boolean
FunctionArn: String
FunctionInvokeRole: String
FunctionPayloadType: String
Identity: LambdaRequestAuthorizationIdentity (p. 101)
```

属性

`AuthorizationScopes`

此授权者的授权范围列表。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`DisableFunctionDefaultPermissions`

指定`true`Amazon SAM防止自动创建`AWS::Lambda::Permissions`资源以在您的`AWS::Serverless::Api`资源和授权方 Lambda 函数之间预置权限。

默认值：`false`

类型：布尔值

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`FunctionArn`

指定 Lambda 函数的函数 ARN，该函数为 API 提供授权。

> **Note**
>
> Amazon SAM为指定时`FunctionArn`将自动创建`AWS::Lambda::Permissions`资源`AWS::Serverless::Api`。`AWS::Lambda::Permissions`资源在您的 API 和授权方 Lambda 函数之间配置权限。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`FunctionInvokeRole`

将授权者凭证添加到 Lambda 授权者的 OpenApi 定义中。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`FunctionPayloadType`

此属性可用于定义 API 的 Lambda 授权器的类型。

有效值：`TOKEN` 或 `REQUEST`

类型：字符串

必需：否

默认值：`TOKEN`

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`Identity`

此属性可用于`IdentitySource`在向授权者发送的传入请求中指定。只有当该属性设置为时，才需要此`FunctionPayloadType`属性`REQUEST`。

类型：LambdaRequestAuthorizationIdentity (p. 101)

必需：条件

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

示例

LambdaRequestAuth

YAML

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

LambdaRequestAuthorizationIdentity

此属性可用于在授权者的传入请求中指定 IdentitySource。有关标识的更多信息，请参阅ApiGateway 授权者 OpenAPI 扩展.

语法

要在您的中声明此实体Amazon Serverless Application Model(Amazon SAM) 模板，请使用以下语法。

YAML

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

属性

`Context`

将给定的上下文字符串转换为格式的映射表达式`context.contextString`.

类型：List

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`Headers`

将标题转换为格式映射表达式的逗号分隔字符串`method.request.header.name`.

类型：List

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`QueryStrings`

将给定的查询字符串转换为格式映射表达式的逗号分隔字符串`method.request.querystring.queryString`.

类型：List

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`ReauthorizeEvery`

生存时间 (TTL) 期间（以秒为单位），用于指定 API Gateway 缓存授权方结果的时长。如果您指定一个大于 0 的值，API Gateway 将缓存授权方响应。默认情况下，API Gateway 将此属性设为 300。最大值为 3600 秒（1 小时）。

类型：整数

必需：否

默认值：300

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`StageVariables`

将给定阶段变量转换为格式映射表达式的逗号分隔字符串`stageVariables.stageVariable`.

类型：List

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

示例

lambda 请求身份

YAML

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

LambdaTokenAuthorizer

配置 Lambda 授权方以通过 Lambda 函数控制对 API 的访问。

有关更多信息以及示例，请参阅 控制对 API Gateway API 的访问 (p. 330)。

语法

要在模板中声明此实体，请使用以下语法。Amazon Serverless Application ModelAmazon SAM

YAML

```
AuthorizationScopes: List
DisableFunctionDefaultPermissions (p. 103): Boolean
FunctionArn: String
FunctionInvokeRole: String
FunctionPayloadType: String
Identity: LambdaTokenAuthorizationIdentity (p. 104)
```

属性

`AuthorizationScopes`

此授权者的授权范围列表。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`DisableFunctionDefaultPermissions`

指定`true`Amazon SAM防止自动创建`AWS::Lambda::Permissions`资源以在您的`AWS::Serverless::Api`资源和授权方 Lambda 函数之间预置权限。

默认值：`false`

类型：布尔值

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`FunctionArn`

指定 Lambda 函数的函数 ARN，该函数为 API 提供授权。

> **Note**
>
> Amazon SAM为指定时`FunctionArn`将自动创建`AWS::Lambda::Permissions`资源`AWS::Serverless::Api`。`AWS::Lambda::Permissions`资源在您的 API 和授权方 Lambda 函数之间配置权限。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`FunctionInvokeRole`

将授权者凭证添加到 Lambda 授权者的 OpenApi 定义中。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`FunctionPayloadType`

此属性可用于定义 Api 的 Lambda 授权器的类型。

有效值：`TOKEN` 或 `REQUEST`

类型：字符串

必需：否

默认值：`TOKEN`

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`Identity`

此属性可用于`IdentitySource`在向授权者发送的传入请求中指定。只有当该属性设置为时，才需要此`FunctionPayloadType`属性`REQUEST`。

类型：LambdaTokenAuthorizationIdentity (p. 104)

必需：条件

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

示例

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

此属性可用于为授权者 IdentitySource 在传入的请求中指定。有关更多信息， IdentitySource 请参阅 A ApiGateway uthorizer OpenApi 扩展。

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

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

属性

`Header`

在 OpenApi 定义中为授权指定标题名称。

类型：字符串

必需：否

默认：授权

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

`ReauthorizeEvery`

time-to-live (TTL) 期间（以秒为单位），用于指定 API Gateway 缓存授权者结果的时长。如果您指定一个大于 0 的值，API Gateway 将缓存授权方响应。默认情况下，API Gateway 将此属性设为 300。最大值为 3600 秒（1 小时）。

类型：整数

必需：否

默认值：300

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

`ValidationExpression`

指定用于验证传入身份的验证表达式。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

示例

LambdaTokenIdentity

YAML

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

ResourcePolicyStatement

为 API 的所有方法和路径配置资源策略。有关资源策略的更多信息，请参阅《API Gateway 开发者指南》中的 “使用 API Gateway 资源策略控制 API 的访问”。

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

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

属性

`AwsAccountBlacklist`

要屏蔽的Amazon账户。

类型：字符串列表

必需：否

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

`AwsAccountWhitelist`

允许的Amazon账户。有关此属性的示例用法，请参阅本页底部的示例部分。

类型：字符串列表

必需：否

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

`CustomStatements`

适用于此 API 的自定义资源策略声明列表。有关此属性的示例用法，请参阅本页底部的示例部分。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

`IntrinsicVpcBlacklist`

要屏蔽的虚拟私有云 (VPC) 列表，其中每个 VPC 都指定为引用，例如动态引用或`Ref`内部函数。有关此属性的示例用法，请参阅本页底部的示例部分。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

`IntrinsicVpcWhitelist`

允许的 VPC 列表，其中每个 VPC 都指定为引用，例如动态引用或`Ref`内部函数。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

`IntrinsicVpceBlacklist`

要屏蔽的 VPC 终端节点列表，其中每个 VPC 终端节点都指定为引用，例如动态引用或`Ref`内部函数。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

`IntrinsicVpceWhitelist`

允许的 VPC 终端节点列表，其中每个 VPC 终端节点都指定为引用，例如动态引用或`Ref`内部函数。有关此属性的示例用法，请参阅本页底部的示例部分。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

`IpRangeBlacklist`

要屏蔽的 IP 地址或地址范围。有关此属性的示例用法，请参阅本页底部的示例部分。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

`IpRangeWhitelist`

允许的 IP 地址或地址范围。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

`SourceVpcBlacklist`

要屏蔽的源 VPC 或 VPC 终端节点。源 VPC 名称必须以开头`"vpc-"`，源 VPC 终端节点名称必须以开头`"vpce-"`。有关此属性的示例用法，请参阅本页底部的示例部分。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

`SourceVpcWhitelist`

允许的源 VPC 或 VPC 终端节点。源 VPC 名称必须以开头`"vpc-"`，源 VPC 终端节点名称必须以开头`"vpce-"`。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

示例

资源策略示例

以下示例屏蔽了两个 IP 地址和一个源 VPC，并允许一个Amazon账户。

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

定义 API 的 OpenAPI 文档。

语法

要在您的Amazon Serverless Application Model(Amazon SAM) 模板，请使用以下语法。

YAML

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

属性

`Bucket`

存储 OpenAPI 文件的 Amazon S3 存储桶的名称。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给`Bucket`的财产`AWS::ApiGateway::RestApi S3Location`数据类型。

`Key`

OpenAPI 文件的 Amazon S3 密钥。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给`Key`的财产`AWS::ApiGateway::RestApi S3Location`数据类型。

`Version`

对于版本控制的对象，则为 OpenAPI 文件的版本。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`Version`的财产`AWS::ApiGateway::RestApi S3Location`数据类型。

示例

示例定义

API 定义示例

YAML

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

CorsConfiguration

为 API Gateway API 管理跨域资源共享 (CORS)。将允许的域指定为字符串，或者指定带有其他 Cors 配置的字典。注意：Cors 要求 SAM 修改你的 OpenAPI 定义，因此它只适用于`DefinitionBody`财产。

有关 CORS 的更多信息，请参阅为 API Gateway 启用 CORS REST API 资源中的API Gateway 开发人员指南.

注意：如果在 OpenAPI 和属性级别都设置了 CorsConfiguration，Amazon SAM将它们合并，优先使用属性。

语法

要在您的中声明此实体Amazon Serverless Application Model(Amazon SAM) 模板，请使用以下语法。

YAML

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

属性

`AllowCredentials`

指示是否允许请求包含凭据的布尔值。

类型：Boolean

必需：否

Amazon CloudFormation兼容性：该属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`AllowHeaders`

要允许的标题字符串。

类型：字符串

必需：否

Amazon CloudFormation兼容性：该属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`AllowMethods`

包含要允许的 HTTP 方法的字符串。

类型：字符串

必需：否

Amazon CloudFormation兼容性：该属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`AllowOrigin`

要允许的原始字符串。

类型：字符串

必需：是

Amazon CloudFormation兼容性：该属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`MaxAge`

包含缓存 CORS 印前检查请求的秒数的字符串。

类型：字符串

必需：否

Amazon CloudFormation兼容性：该属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

示例

CorsConfiguration

Core 配置示例。这只是一个Amazon SAM显示模板文件AWS::Serverless::Api (p. 83)配置了 Cora 的定义。

YAML

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

DomainConfiguration

为 API 配置自定义域。

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

```
  BasePath: List
  NormalizeBasePath: Boolean
  CertificateArn: String
  DomainName: String
  EndpointConfiguration: String
  MutualTlsAuthentication: MutualTlsAuthentication
  OwnershipVerificationCertificateArn: String
  Route53: Route53Configuration (p. 114)
  SecurityPolicy: String
```

属性

`BasePath`

要使用Amazon API Gateway 域名配置的基本路径列表。

类型：清单

必需：否

默认：/

Amazon CloudFormation兼容性：此属性类似于AWS::ApiGateway::BasePathMapping资源的BasePath属性。 Amazon SAM创建多个AWS::ApiGateway::BasePathMapping资源，每个资源在此属性中BasePath指定一个。

NormalizeBasePath

指示是否允许在BasePath属性定义的基本路径中使用非字母数字字符。设置为时True，将从基本路径中删除非字母数字字符。

NormalizeBasePath与该BasePath属性一起使用。

类型：布尔值

必需：否

默认：真

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

CertificateArn

此域名的终端节点的 Amazon 资源名称 (ARN)。Amazon Amazon Certificate Manager是唯一受支持的源。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性类似于AWS::ApiGateway::DomainName资源的CertificateArn属性。如果设置EndpointConfiguration为REGIONAL（默认值），则CertificateArn映射到 i RegionalCertificateArnnAWS::ApiGateway::DomainName。如果设置EndpointConfiguration为，则CertificateArn映射到 i CertificateArnnAWS::ApiGateway::DomainName。EDGE

其他说明：对于终EDGE端节点，您必须在us-east-1Amazon区域中创建证书。

DomainName

API Gateway API 的自定义域名。不支持大写字母。

Amazon SAM设置此属性时生成AWS::ApiGateway::DomainName资源。有关此方案的信息，请参阅指定了 DomainName 属性 (p. 263)。有关生成的Amazon CloudFormation资源的信息，请参阅Amazon CloudFormation (p. 261)。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性直接传递给AWS::ApiGateway::DomainName资源的DomainName属性。

EndpointConfiguration

定义要映射到自定义域的 API Gateway 端点的类型。此属性的值决定了该CertificateArn属性的映射方式Amazon CloudFormation。

有效值：REGIONAL 或 EDGE

类型：字符串

必需：否

默认值：REGIONAL

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`MutualTlsAuthentication`

自定义域名的相互 TLS 安全性 (TLS) 身份验证配置。

类型：MutualTlsAuthentication

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::ApiGateway::DomainName`资源的`MutualTlsAuthentication`属性。

`OwnershipVerificationCertificateArn`

ACM 颁发的用于验证自定义域所有权的公有证书的 ARN。仅在配置双向 TLS 并指定导入或私有 CA 证书 ARN 作为 Arn 时才需要`CertificateArn`。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::ApiGateway::DomainName`资源的`OwnershipVerificationCertificateArn`属性。

`Route53`

定义亚马逊 Route 53 配置。

类型：Route53 配置 (p. 114)

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`SecurityPolicy`

TLS 版本加上此域名的密码套件。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::ApiGateway::DomainName`资源的`SecurityPolicy`属性。

示例

DomainName

DomainName 示例

YAML

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

```
  EndpointConfiguration: EDGE
  Route53:
    HostedZoneId: Z1PA6795UKMFR9
  BasePath:
    - foo
    - bar
```

Route53Configuration

为 API 配置 Route53 记录集。

语法

要在您的Amazon Serverless Application Model(Amazon SAM) 模板，请使用以下语法。

YAML

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

属性

`DistributionDomainName`

配置 API 自定义域名的自定义分配。

类型：字符串

必需：否

默认值：使用 API Gateway 分发。

Amazon CloudFormation兼容性：此属性将直接传递给`DNSName`一个的财产`AWS::Route53::RecordSetGroup AliasTarget`资源。

附加说明：域名CloudFront 分配.

`EvaluateTargetHealth`

如果为 true，则别名记录将继承引用的运行状况Amazon资源，例如 Elastic Load Balancing 负载均衡器或托管区域中的其他记录。

类型：Boolean

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`EvaluateTargetHealth`一个的财产`AWS::Route53::RecordSetGroup AliasTarget`资源。

附加说明：如果别名目标为 CloudFront，则无法将 valuateTargeThealth 设置为 true。

`HostedZoneId`

要在其中创建记录的托管区域的 ID。

指定 `HostedZoneName` 或 `HostedZoneId`，但不能同时指定两者。如果您拥有多个使用相同域名的托管区域，则必须使用 `HostedZoneId` 指定托管区域。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给HostedZoneId一个的财产AWS::Route53::RecordSetGroup RecordSet资源。

`HostedZoneName`

要在其中创建记录的托管区域的名称。

指定 HostedZoneName 或 HostedZoneId，但不能同时指定两者。如果您拥有多个使用相同域名的托管区域，则必须使用 HostedZoneId 指定托管区域。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给HostedZoneName一个的财产AWS::Route53::RecordSetGroup RecordSet资源。

`IpV6`

设置此属性时，Amazon SAM创建AWS::Route53::RecordSet资源和集类型到AAAA对于提供的HostedZone。

类型：Boolean

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

示例

Route 53 配置示例

此示例演示如何配置 Route 53。

YAML

```
Domain:
  DomainName: www.example.com
  CertificateArn: arn-example
  EndpointConfiguration: EDGE
  Route53:
    HostedZoneId: Z1PA6795UKMFR9
    EvaluateTargetHealth: true
    DistributionDomainName: xyz
```

EndpointConfiguration

REST API 的终端节点类型。

语法

要在您的Amazon Serverless Application Model(Amazon SAM) 模板，请使用以下语法。

YAML

```
  Type: String
```

```
  VPCEndpointIds: List
```

属性

Type

REST API 的终端节点类型。

有效值：EDGE要么REGIONAL要么PRIVATE

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给Types的财产AWS::ApiGateway::RestApi EndpointConfiguration数据类型。

VPCEndpointIds

要创建 Route53 别名的 REST API 的 VPC 终端节点 ID 列表。

类型：List

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给VpcEndpointIds的财产AWS::ApiGateway::RestApi EndpointConfiguration数据类型。

示例

EndpointConfiguration

端点配置示例

YAML

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

AWS::Serverless::Application

将来自Amazon Serverless Application Repository或来自 Amazon S3 存储桶的无服务器应用程序嵌入为嵌套应用程序。嵌套应用程序作为嵌套AWS::CloudFormation::Stack资源部署，嵌套资源可以包含多个其他资源，包括其他AWS::Serverless::Application (p. 116)资源。

Note

部署到时Amazon CloudFormation，Amazon SAM将您的Amazon SAM资源转换为Amazon CloudFormation资源。有关更多信息，请参阅Amazon CloudFormation (p. 261)：

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

```
Type: AWS::Serverless::Application
Properties:
  Location: String | ApplicationLocationObject (p. 119)
  NotificationARNs: List
  Parameters: Map
  Tags: Map
  TimeoutInMinutes: Integer
```

属性

`Location`

嵌套应用程序的模板 URL、文件路径或位置对象。

如果提供了模板 URL，则它必须遵循CloudFormation TemplateUrl 文档中指定的格式并包含有效的 CloudFormation 或 SAM 模板。ApplicationLocationObject (p. 119)可用于指定已发布到的应用程序Amazon Serverless Application Repository。

如果提供了本地文件路径，则模板必须经过包含`sam deploy`或`sam package`命令的工作流程，才能正确转换应用程序。

类型：字符串 | ApplicationLocationObject (p. 119)

必需：是

Amazon CloudFormation兼容性：此属性类似于`AWS::CloudFormation::Stack`资源的`TemplateURL`属性。该 CloudFormation 版本不需要从ApplicationLocationObject (p. 119)中检索应用程序Amazon Serverless Application Repository。

`NotificationARNs`

将发送有关堆栈事件的通知的通知的通知的通知的通知的通知的通知的通知的通知的现有 Amaion 主题。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::CloudFormation::Stack`资源的`NotificationARNs`属性。

`Parameters`

应用程序参数值。

类型：地图

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::CloudFormation::Stack`资源的`Parameters`属性。

`Tags`

一个映射（字符串到字符串），指定要添加到此应用程序的标签的标签。密钥和值只能包含字母数字字符。密钥的长度可以在 1 到 127 个 Unicode 字符之间，并且不能带有前缀 aws:。值的长度可以在 1 到 255 个 Unicode 字符之间。

类型：地图

必需：否

Amazon CloudFormation兼容性：此属性类似于AWS::CloudFormation::Stack资源的Tags属性。SAM 中的 Tags 属性由 Key: Value 对组成； CloudFormation 它由标签对象列表组成。创建堆栈后，SAM 将自动向该应用程序添加lambda:createdBy:SAM标签。此外，如果此应用程序来自于Amazon Serverless Application Repository，则 SAM 还将自动添加两个附加标签serverlessrepo:applicationId:*ApplicationId*和serverlessrepo:semanticVersion:*SemanticVe*

`TimeoutInMinutes`

Amazon CloudFormation 等待嵌套堆栈达到 CREATE_COMPLETE 状态的时间长度（以分钟为单位）。默认值为无超时。在 Amazon CloudFormation 检测到嵌套堆栈已达到 CREATE_COMPLETE 状态时，它在父堆栈中将嵌套堆栈资源标记为 CREATE_COMPLETE，然后继续创建父堆栈。如果在嵌套堆栈达到 CREATE_COMPLETE 之前超时期限过期，则 Amazon CloudFormation 将嵌套堆栈标记为失败并回滚嵌套堆栈和父堆栈。

类型：整数

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::CloudFormation::Stack资源的TimeoutInMinutes属性。

返回值

Ref

当该资源的逻辑 ID 提供给Ref内部函数时，它将返回基础资源的AWS::CloudFormation::Stack资源名称。

有关使用该Ref函数的更多信息，请参阅Ref《Amazon CloudFormation用户指南》。

Fn::GetAtt

Fn::GetAtt 返回一个此类型指定属性的值。以下为可用属性和示例返回值。

有关使用的更多信息Fn::GetAtt，请参阅Fn::GetAttAmazon CloudFormation用户指南中的。

`Outputs.ApplicationOutputName`

带有名称的堆栈输出的值*ApplicationOutputName*。

示例

特区应用程序

使用Serverless Application Repository 中的模板的应用程序

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
```

速正常关应用程序

来自 S3 网址的应用程序

YAML

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

ApplicationLocationObject

已发布到Amazon Serverless Application Repository.

语法

要在您的中声明此实体Amazon Serverless Application Model(Amazon SAM) 模板，请使用以下语法。

YAML

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

属性

`ApplicationId`

应用程序的 Amazon 资源名称 (ARN)。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`SemanticVersion`

应用程序的语义版本。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

示例

我的申请

示例应用程序位置物

YAML

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

```
  SemanticVersion: 1.0.0
```

AWS::Serverless::Connector

其中配置其中包含其中包含其中定义的资源。有关连接器的简介，请参见使用Amazon SAM连接器管理资源权限 (p. 273)。

有关生成的Amazon CloudFormation资源的更多信息，请参阅Amazon CloudFormation指定时生成的资源AWS::Serverless::Connector (p. 264)。

要提供有关连接器的反馈，请在serverless-application-model Amazon GitHub 存储库中提交新问题。

> Note
>
> 当您部署到时Amazon CloudFormation，Amazon SAM会将您的Amazon SAM资源转换为Amazon CloudFormation资源。有关更多信息，请参阅Amazon CloudFormation (p. 261)：

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下其中其中一种语法。

> Note
>
> 我们建议在大多数用例中使用嵌入式连接器语法。嵌入到源资源中可以使其随着时间的推移更易于阅读和维护。当您需要引用不在同一Amazon SAM模板中的源资源（例如嵌套堆栈中的资源或共享资源）时，请使用AWS::Serverless::Connector语法。

嵌入式连接器

```
<source-resource-logical-id>:
  Connectors:
    <connector-logical-id>:
      Properties:
        Destination: ResourceReference (p. 125) | List of ResourceReference (p. 125)
        Permissions: List
        SourceReference: SourceReference (p. 128)
```

AWS::Serverless::Connector

```
Type: AWS::Serverless::Connector
Properties:
  Destination: ResourceReference (p. 125) | List of ResourceReference (p. 125)
  Permissions: List
  Source: ResourceReference (p. 125)
```

属性

`Destination`

目标资源。

类型:ResourceReference (p. 125)| 清单 ResourceReference (p. 125)

必需：是

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`Permissions`

允许源资源对目标资源执行的权限类型。

`Read`包括允许从资源读取数据的Amazon Identity and Access Management (IAM) 操作。

`Write`包括允许启动数据并将数据写入资源的 IAM 操作。

有效值：`Read` 或 `Write`

类型：清单

必需：是

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`Source`

源资源。使用`AWS::Serverless::Connector`语法时为必填项。

类型：ResourceReference (p. 125)

必需：条件

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`SourceReference`

源资源。

> **Note**
>
> 在为源资源定义其他属性时，与嵌入式连接器语法一起使用。

类型：SourceReference (p. 128)

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

示例

嵌入式连接器

以下示例使用嵌入式连接器定义Amazon Lambda函数与 Amazon DynamoDB 表之间的`Write`数据连接：

```
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
```

```
    ...
```

以下示例使用嵌入式连接器来定义Read和Write权限：

```
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
    ...
```

以下示例使用嵌入式连接器定义具有以下属性的源资源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

以下示例使用AWS::Serverless::Connector (p. 120)资源来读取和写入 Amazon DynamoDB 表的Amazon Lambda函数：

```
MyConnector:
  Type: AWS::Serverless::Connector
  Properties:
    Source:
      Id: MyFunction
    Destination:
      Id: MyTable
    Permissions:
      - Read
      - Write
```

以下示例使用AWS::Serverless::Connector (p. 120)资源让 Lambda 函数写入 Amazon SNS 主题，这两个资源位于同一个模板中：

```
MyConnector:
  Type: AWS::Serverless::Connector
  Properties:
    Source:
      Id: MyLambda
    Destination:
      Id: MySNSTopic
    Permissions:
      - Write
```

以下示例使用AWS::Serverless::Connector (p. 120)资源让 Amazon SNS 主题写入 Lambda 函数，然后该函数写入 Amazon DynamoDB 表，所有资源都位于同一个模板中：

```
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
```

以下是上面示例中转换后的Amazon CloudFormation模板：

```
"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

对资源类型使用的AWS::Serverless::Connector (p. 120)资源的引用。

> Note
>
> 对于同一模板中的资源，请提供Id。对于不在同一个模板中的资源，请使用其他属性的组合。有关更多信息，请参阅 Amazon SAM连接器参考 (p. 451)。

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

```
Arn: String
Id: String
Name: String
Qualifier: String
QueueUrl: String
ResourceId: String
RoleName: String
Type: String
```

属性

`Arn`

资源的 ARN。

类型：字符串

必填项：条件

Amazon CloudFormation兼容性：此属性是独一无二的Amazon SAM，没有Amazon CloudFormation等效属性。

`Id`

同一模板中资源的逻辑 ID。

> Note
>
> 指定后Id，如果连接器生成Amazon Identity and Access Management (IAM) 策略，则将从资源中推断出与这些策略关联的 IAM 角色Id。如果Id未指定，则为连接器提供RoleName资源，以便将生成的 IAM 策略附加到 IAM 角色。

类型：字符串

必填项：条件

Amazon CloudFormation兼容性：此属性是独一无二的Amazon SAM，没有Amazon CloudFormation等效属性。

`Name`

资源的名称。

类型：字符串

必填项：条件

Amazon CloudFormation兼容性：此属性是独一无二的Amazon SAM，没有Amazon CloudFormation等效属性。

`Qualifier`

缩小其范围的资源的限定符。 `Qualifier`替换资源约束 ARN 末尾的`*`值。有关示例，请参阅 API Gateway 调用 Lambda 函数 (p. 127)。

> **Note**
>
> 限定词的定义因资源类型而异。有关支持的源资源和目标资源类型的列表，请参阅Amazon SAM连接器参考 (p. 451)。

类型：字符串

必填项：条件

Amazon CloudFormation兼容性：此属性是独一无二的Amazon SAM，没有Amazon CloudFormation等效属性。

`QueueUrl`

Amazon SQS 队列 URL。此属性仅适用于 Amazon SQS 资源。

类型：字符串

必填项：条件

Amazon CloudFormation兼容性：此属性是独一无二的Amazon SAM，没有Amazon CloudFormation等效属性。

`ResourceId`

资源的 ID。例如，API Gateway API ID。

类型：字符串

必填项：条件

Amazon CloudFormation兼容性：此属性是独一无二的Amazon SAM，没有Amazon CloudFormation等效属性。

`RoleName`

与资源关联的角色名称。

> **Note**
>
> 指定后`Id`，如果连接器生成 IAM 策略，则将从资源中推断出与这些策略关联的 IAM 角色`Id`。如果`Id`未指定，则为连接器提供`RoleName`资源，以便将生成的 IAM 策略附加到 IAM 角色。

类型：字符串

必填项：条件

Amazon CloudFormation兼容性：此属性是独一无二的Amazon SAM，没有Amazon CloudFormation等效属性。

`Type`

资源的Amazon CloudFormation类型。有关更多信息，请参阅Amazon资源和属性类型参考。

类型：字符串

必填项：条件

Amazon CloudFormation兼容性：此属性是独一无二的Amazon SAM，没有Amazon CloudFormation等效属性。

示例

API Gateway 调用 Lambda 函数

以下示例使用AWS::Serverless::Connector (p. 120)资源允许 Amazon API Gateway 调用Amazon Lambda函数。

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

对资源类型使用的源AWS::Serverless::Connector (p. 120)资源的引用。

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

```
Qualifier: String
```

属性

`Qualifier`

缩小其范围的资源的限定词。 `Qualifier`替换资源约束 ARN 末尾的`*`值。

> **Note**
>
> 限定符定义因资源类型而异。有关支持的源和目标资源类型的列表，请参阅Amazon SAM连接器参考 (p. 451)。

类型：字符串

必需：条件

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

示例

以下示例使用嵌入式连接器定义具有以下属性的源资源**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

创建触发该Amazon Lambda函数的函数、Amazon Identity and Access Management (IAM) 执行角色和事件源映射。

该AWS::Serverless::Function (p. 129)资源还支持Metadata资源属性，因此您可以指示Amazon SAM构建应用程序所需的自定义运行时。有关构建自定义运行时环境的更多信息，请参阅构建自定义运行时 (p. 355)。

> **Note**
>
> 当您部署到时Amazon CloudFormation，Amazon SAM会将您的Amazon SAM资源转换为Amazon CloudFormation资源。有关更多信息，请参阅Amazon CloudFormation (p. 261)：

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

```
Type: AWS::Serverless::Function
Properties:
  Architectures: List
  AssumeRolePolicyDocument: JSON
  AutoPublishAlias: String
  AutoPublishAliasAllProperties (p. 131): Boolean
  AutoPublishCodeSha256: String
  CodeSigningConfigArn: String
  CodeUri: String | FunctionCode (p. 206)
  DeadLetterQueue: Map | DeadLetterQueue (p. 141)
  DeploymentPreference: DeploymentPreference (p. 142)
  Description: String
  Environment: Environment
  EphemeralStorage: EphemeralStorage
  EventInvokeConfig: EventInvokeConfiguration (p. 145)
  Events: EventSource (p. 150)
  FileSystemConfigs: List
  FunctionName: String
```

```
  FunctionUrlConfig: FunctionUrlConfig (p. 207)
  Handler: String
  ImageConfig: ImageConfig
  ImageUri: String
  InlineCode: String
  KmsKeyArn: String
  Layers: List
  MemorySize: Integer
  PackageType: String
  PermissionsBoundary: String
  Policies: String | List | Map
  ProvisionedConcurrencyConfig: ProvisionedConcurrencyConfig
  ReservedConcurrentExecutions: Integer
  Role: String
  RolePath: String
  Runtime: String
  RuntimeManagementConfig (p. 137): RuntimeManagementConfig
  SnapStart (p. 138): SnapStart
  Tags: Map
  Timeout: Integer
  Tracing: String
  VersionDescription: String
  VpcConfig: VpcConfig
```

属性

`Architectures`

该函数的指令集架构。

有关此属性的更多信息，请参阅Amazon Lambda开发者指南中的 Lambda 指令集架构。

有效值：`x86_64`或中的一个`arm64`

类型：清单

必需：否

默认值：`x86_64`

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Lambda::Function`资源的`Architectures`属性。

`AssumeRolePolicyDocument`

为 AssumeRolePolicyDocument 为此函数创建`Role`的默认值添加一个。如果未指定此属性，则为该函数Amazon SAM添加默认的代入角色。

类型：JSON

必需：否

Amazon CloudFormation兼容性：此属性类似于`AWS::IAM::Role`资源的`AssumeRolePolicyDocument`属性。 Amazon SAM将此属性添加到为此函数生成的 IAM 角色中。如果为该函数提供了一个角色的 Amazon Resource Name (ARN)，该属性将不执行任何操作。

`AutoPublishAlias`

Lambda 的名称。有关 Lambda 别名的更多信息，请参阅Amazon Lambda开发人员指南中的 Lambda 函数别名。有关使用此属性的示例，请参见逐步部署无服务器应用程序 (p. 460)。

Amazon SAM设置此属性时生成AWS::Lambda::Version和AWS::Lambda::Alias资源。有关此方案的信息，请参阅指定了 AutoPublishAlias 属性 (p. 265)。有关生成的Amazon CloudFormation资源的一般信息，请参阅Amazon CloudFormation (p. 261)。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`AutoPublishAliasAllProperties`

指定何时创建新`AWS::Lambda::Version`的。当`true`修改 Lambda 函数中的任何属性时，就会创建一个新的 Lambda 版本。时`false`，只有在修改了以下任何属性时才会创建新的 Lambda 版本：

- `Environment`, `MemorySize`, 或者 `SnapStart`。
- 导致`Code`属性更新的任何更改，例如`CodeDictImageUri`、或`InlineCode`。

需要定义`AutoPublishAlias`此属性。

如果`AutoPublishSha256`还指定了，则其行为优先于`AutoPublishAliasAllProperties: true`。

类型：布尔值

必需：否

默认值：`false`

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`AutoPublishCodeSha256`

用于确定是否应发布新的 Lambda 版本的字符串值以及中的`CodeUri`值。只有在还定义了此属性时`AutoPublishAlias`才会使用。

此属性解决了Amazon SAM模板具有以下特征时出现的问题：`DeploymentPreference`对象配置为渐进部署（如中所述逐步部署无服务器应用程序 (p. 460)），该`AutoPublishAlias`属性已设置且不会在部署之间更改，该`CodeUri`属性已设置且在部署之间不发生变化。

当存储在 Amazon Simple Storage Service (Amazon S3) 位置的部署包被包含更新的 Lambda 函数代码的新部署包取代，但该`CodeUri`属性保持不变（而不是将新的部署包上传到新的 Amazon S3 位置并更改为新位置）时，就会出现这种情况。`CodeUri`

在这种情况下，要成功触发渐进部署，必须为提供唯一值`AutoPublishCodeSha256`。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`CodeSigningConfigArn`

`AWS::Lambda::CodeSigningConfig`资源的 ARN，用于为此函数启用代码签名。有关代码签名的更多信息，请参阅为Amazon SAM应用程序配置代码签名 (p. 339)。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Lambda::Function`资源的`CodeSigningConfigArn`属性。

`CodeUri`

函数代码的 Amazon S3 URI、本地文件夹或FunctionCode (p. 206)对象的路径。此属性仅在该`PackageType`属性设置为时适用`Zip`，否则会被忽略。

备注:

1. 如果该`PackageType`属性设置为`Zip`（默认），`InlineCode`则需要`CodeUri`或中的一个。

2. 如果提供了 Amazon S3 URI 或FunctionCode (p. 206)对象，则引用的Amazon S3 对象必须是有效的 Lambda 部署包。

3. 如果提供了本地文件夹的路径，则要正确转换代码，模板必须经过包括sam deploy (p. 412)或山姆·布莱德 (p. 408)在内的工作流程sam package (p. 431)。默认情况下，相对路径是相对于Amazon SAM模板的位置进行解析的。

类型：字符串 | FunctionCode (p. 206)

必需：条件

Amazon CloudFormation兼容性：此属性类似于`AWS::Lambda::Function`资源的`Code`属性。嵌套的 Amazon S3 属性的命名方式不同。

`DeadLetterQueue`

配置Amazon Simple Notification Service (Amazon SNS) 主题或 Amazon Simple Queue Service (Amazon SQS) 队列，Lambda 将 有关死信队列功能的更多信息，请参阅Amazon Lambda开发者指南中的Amazon Lambda函数死信队列。

> **Note**
>
> 如果您的 Lambda 函数的事件源是 Amazon SQS 队列，请为源队列而不是 Lambda 函数配置死信队列。您为该函数配置的死信队列用于该函数的异步调用队列，而不是用于事件源队列。

类型：地图 | DeadLetterQueue (p. 141)

必需：否

Amazon CloudFormation兼容性：此属性类似于`AWS::Lambda::Function`资源的`DeadLetterConfig`属性。Amazon CloudFormation在 type 中派生自`TargetArn`，而在中，Amazon SAM你必须将类型和一起传递`TargetArn`。

`DeploymentPreference`

用于实现渐进式 Lambda 部署的设置。

如果指定了`DeploymentPreference`对象，则Amazon SAM创建一个AWS::CodeDeploy::Application被调用对象`ServerlessDeploymentApplication`（每个堆栈一个）、一个AWS::CodeDeploy::DeploymentGroup被调用*`<function-logical-id>`*`DeploymentGroup`者和一个AWS::IAM::Role被调用对象`CodeDeployServiceRole`。

类型：DeploymentPreference (p. 142)

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

另请参阅：有关此属性的更多信息，请参阅逐步部署无服务器应用程序 (p. 460)。

`Description`

该函数的描述。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::Function资源的Description属性。

`Environment`

运行时环境的配置。

类型：环境

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::Function资源的Environment属性。

`EphemeralStorage`

一个对象，用于指定 Lambda 函数在中可用的磁盘空间（以 MB 为单位）/tmp。

有关此属性的更多信息，请参阅Amazon Lambda开发者指南中的 Lambda 执行环境。

类型：EphemeralStorage

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::Function资源的EphemeralStorage属性。

`EventInvokeConfig`

描述 Lambda 函数的事件调用配置的对象。

类型：EventInvokeConfiguration (p. 145)

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`Events`

指定触发此函数的事件。事件由一种类型和一组依赖于该类型的属性组成。

类型：EventSource (p. 150)

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`FileSystemConfigs`

指定Amazon EEEFS File System (Amazon EFS) 文件系统连接设置的FileSystemConfig对象列表。

如果模板包含AWS::EFS::MountTarget资源，您还必须指定DependsOn资源属性，以确保在函数之前创建或更新装载目标。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::Function资源的FileSystemConfigs属性。

FunctionName

函数的名称。如果没有指定名称，将为您生成一个唯一的名称。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::Function资源的FunctionName属性。

FunctionUrlConfig

描述函数 URL 的对象。函数 URL 是一个 HTTPS 端点，可用于调用函数。

有关更多信息，请参阅《Amazon Lambda开发者指南》中的函数 URL。

类型：FunctionUrlConfig (p. 207)

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

Handler

代码中为开始执行而调用的函数。只有当该属性设置为时，才需要此PackageType属性Zip。

类型：字符串

必需：条件

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::Function资源的Handler属性。

ImageConfig

用于配置 Lambda 容器映像设置的对象。有关更多信息，请参阅Amazon Lambda开发者指南中的 Lambda 中使用容器镜像。

类型：ImageConfig

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::Function资源的ImageConfig属性。

ImageUri

Lambda 函数的容器镜像的 Amazon Elastic Container Registry (Amazon ECR) 存储库的 URI。此属性仅在该PackageType属性设置为时适用Image，否则会被忽略。有关更多信息，请参阅Amazon Lambda开发者指南中的 Lambda 中使用容器镜像。

> Note
>
> 如果该PackageType属性设置为Image，则其中一个ImageUri为必填项，或者您必须在Amazon SAM模板文件中使用必要的Metadata条目来构建应用程序。有关更多信息，请参阅构建应用程序 (p. 342)：

使用必需Metadata条目构建应用程序的优先级高于ImageUri，因此，如果您同时指定这两个条目，ImageUri则会被忽略。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::FunctionCode数据类型的ImageUri属性。

InlineCode

直接在模板中编写的 Lambda 函数代码。此属性仅在该PackageType属性设置为时适用Zip，否则会被忽略。

Note

如果该PackageType属性设置为Zip（默认），InlineCode则需要CodeUri或中的一个。

类型：字符串

必需：条件

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::FunctionCode数据类型的ZipFile属性。

KmsKeyArn

Lambda 用于加密和解密函数的环境变量的Amazon Key Management Service (Amazon KMS) 密钥的 ARN。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::Function资源的KmsKeyArn属性。

Layers

此函数应使用的LayerVersion ARN 列表。此处指定的顺序是运行 Lambda 函数时导入它们的顺序。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::Function资源的Layers属性。

MemorySize

每次调用函数分配的内存大小（以 MB 为单位）。

类型：整数

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::Function资源的MemorySize属性。

PackageType

Lambda 函数的部署包类型。有关更多信息，请参阅Amazon Lambda开发人员指南中的 Lambda 部署包。

备注:

1. 如果此属性设置为Zip（默认），则InlineCode应用CodeUri或并ImageUri被忽略。

2. 如果将此属性设置为Image，则仅ImageUri适用，并且CodeUri和InlineCode都被忽略。CLAmazon SAM I 可以auto 创建存储函数容器映像所需的 Amazon ECR 存储库。有关更多信息，请参阅sam deploy (p. 412)：

有效值：Zip 或 Image

类型：字符串

必需：否

默认值：Zip

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::Function资源的PackageType属性。

PermissionsBoundary

用于该函数的执行角色的权限边界的 ARN。只有在为你生成角色时，此属性才有效。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::IAM::Role资源的PermissionsBoundary属性。

Policies

此功能需要的一项或多项策略。它们将附加到此函数的默认角色中。

此属性接受单个字符串或字符串列表，可以是Amazon托管Amazon SAM策略或策略模板的名称，也可以是以 YAML 格式化的内联 IAM 策略文档。

有关Amazon托管策略的更多信息，请参阅 IAM 用户指南中的Amazon托管策略。有关Amazon SAM策略模板的更多信息Amazon SAM策略模板 (p. 278)，请参阅Amazon Serverless Application Model开发者指南。有关内联策略的更多信息，请参阅 IAM 用户指南中的内联策略。

Note

如果设置该Role属性，将忽略该属性。

类型：字符串 | 列表 | 地图

必需：否

Amazon CloudFormation兼容性：此属性类似于AWS::IAM::Role资源的Policies属性。 Amazon SAM除了 JSONAmazon SAM 策略文档外，还支持Amazon托管策略名称和策略模板。 Amazon CloudFormation仅接受 JSON 策略文档。

ProvisionedConcurrencyConfig

为函数的别名预置的并发配置。

Note

ProvisionedConcurrencyConfig只有在设置了时AutoPublishAlias才能指定。否则将出错。

类型：ProvisionedConcurrencyConfig

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::Alias资源的ProvisionedConcurrencyConfig属性。

`ReservedConcurrentExecutions`

您要为该函数预留的最大数量。

有关此属性的更多信息，请参阅Amazon Lambda开发人员指南中的 Lambda 函数扩展。

类型：整数

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Lambda::Function`资源的`ReservedConcurrentExecutions`属性。

`Role`

用作该函数的执行角色的 IAM 角色的 ARN。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性类似于`AWS::Lambda::Function`资源的`Role`属性。这是必填项，Amazon CloudFormation但不是必填项Amazon SAM。如果未指定角色，则会为您创建一个逻辑 ID 为的角色`<function-logical-id>`Role。

`RolePath`

该函数的 IAM 执行角色的路径。

在为您生成角色时使用此属性。在使用`Role`属性指定角色时不要使用。

类型：字符串

必需：条件

Amazon CloudFormation兼容性：此属性直接传递给`AWS::IAM::Role`资源的`Path`属性。

`Runtime`

函数的运行时的标识符。只有当该属性设置为时，才需要此`PackageType`属性`Zip`。

> **Note**
>
> 如果您为此属性指定了`provided`标识符，则可以使用`Metadata`资源属性Amazon SAM来指示构建此函数所需的自定义运行时。有关构建自定义运行时环境的更多信息，请参阅构建自定义运行时 (p. 355)。

类型：字符串

必需：条件

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Lambda::Function`资源的`Runtime`属性。

`RuntimeManagementConfig`

为您的 Lambda 函数配置运行时管理选项，例如运行时环境更新、回滚行为和选择特定的运行时版本。要了解更多信息，请参阅Amazon Lambda开发者指南中的 Lambda 运行时更新。

> **Note**
>
> 如果已配置，`AutoPublishAlias`则`RuntimeManagementConfig`将`$LATEST`同时适用于新创建的函数版本。

类型：RuntimeManagementConfig

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::Function资源的RuntimeManagementConfig属性。

SnapStart

创建任何新 Lambda 函数版本的快照。快照是初始化函数的缓存状态，包括其所有依赖关系。该函数仅初始化一次，缓存状态将在future 的所有调用中重复使用，从而通过减少必须初始化函数的次数来提高应用程序性能。要了解更多信息，请参阅Amazon Lambda开发者指南 SnapStart中的 “使用 Lambda 提高启动性能”。

类型：SnapStart

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::Function资源的SnapStart属性。

Tags

一个映射（字符串到字符串），指定添加到该函数的标签。有关标签的有效密钥和值的详细信息，请参阅Amazon Lambda开发者指南中的标签密钥和值要求。

创建堆栈时，Amazon SAM会自动向此 Lambda 函数以及为此函数生成的默认角色添加lambda:createdBy:SAM标签。

类型：地图

必需：否

Amazon CloudFormation兼容性：此属性类似于AWS::Lambda::Function资源的Tags属性。中的Tags属性由键值对Amazon SAM组成（而Amazon CloudFormation此属性由Tag对象列表组成）。此外，还Amazon SAM会自动向此 Lambda 函数以及为此函数生成的默认角色添加lambda:createdBy:SAM标签。

Timeout

该函数在停止之前可以运行的最长时间（秒）。

类型：整数

必需：否

原定设置值：3

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::Function资源的Timeout属性。

Tracing

指定函数的 X-Ray 跟踪模式的字符串。有关 X-Ray ay Amazon Lambda的更多信息，请参阅Amazon Lambda开发人员指南Amazon X-Ray中的使用。

有效值：Active 或 PassThrough

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性类似于AWS::Lambda::Function资源的TracingConfig属性。如果该Tracing属性设置为Active且未指定该Role属性，则Amazon SAM将

该arn:aws:iam::aws:policy/AWSXrayWriteOnlyAccess策略添加到它为您创建的 Lambda 执行角色中。

`VersionDescription`

指定在新 Lambda 版本资源上添加的Description字段。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::Version资源的Description属性。

`VpcConfig`

使该函数能够访问虚拟Private Cloud (VPC) 中的私有资源的配置。

类型：VpcConfig

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::Function资源的VpcConfig属性。

返回值

Ref

当该资源的逻辑 ID 提供给Ref内部函数时，它将返回底层 Lambda 函数的资源名称。

有关使用该Ref函数的更多信息，请参阅Ref《Amazon CloudFormation用户指南》。

Fn::GetAtt

Fn::GetAtt 返回一个此类型指定属性的值。以下为可用属性和示例返回值。

有关使用的更多信息Fn::GetAtt，请参阅Fn::GetAtt《Amazon CloudFormation用户指南》。

`Arn`

底层 Lambda 函数的 ARN。

示例

简单的函数

以下是 Amazon S3 存储桶中包类型Zip（默认）和函数代码的AWS::Serverless::Function (p. 129)资源的基本示例。

YAML

```
Type: AWS::Serverless::Function
Properties:
  Handler: index.handler
  Runtime: python3.9
  CodeUri: s3://bucket-name/key-name
```

函数属性示例

以下是使用InlineCode、、、、Layers和Api事件源的包类型Zip（默认）的示例。AWS::Serverless::Function (p. 129)TracingPoliciesAmazon EFS

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示例

以下是包类型的 Lambda 函数的示例Image。ImageConfig

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 例子

配置为根据当前行为更新其运行时环境的 Lambda 函数：

```
TestFunction
  Type: AWS::Serverless::Function
  Properties:
    ...
    Runtime: python3.9
    RuntimeManagementConfig:
      UpdateRuntimeOn: Auto
```

配置为在函数更新时更新其运行时环境的 Lambda 函数：

```
TestFunction
  Type: AWS::Serverless::Function
  Properties:
    ...
    Runtime: python3.9
    RuntimeManagementConfig:
      UpdateRuntimeOn: FunctionUpdate
```

配置为手动更新其运行时环境的 Lambda 函数：

```
TestFunction
  Type: AWS::Serverless::Function
  Properties:
    ...
    Runtime: python3.9
    RuntimeManagementConfig:
      RuntimeVersionArn: arn:aws:lambda:us-
east-1::runtime:4c459dd0104ee29ec65dcad056c0b3ddbe20d6db76b265ade7eda9a066859b1e
      UpdateRuntimeOn: Manual
```

SnapStart 例子

为future 版本 SnapStart 开启的 Lambda 函数示例：

```
TestFunc
  Type: AWS::Serverless::Function
  Properties:
    ...
    SnapStart:
      ApplyOn: PublishedVersions
```

DeadLetterQueue

指定 SQS 队列或 SNS 主题Amazon Lambda(Lambda) 在无法处理事件时将其发送到的时候。有关死信队列功能的更多信息，请参阅Amazon Lambda函数死信队列.

SAM 将自动向您的 Lambda 函数执行角色添加适当的权限，以授予 Lambda 服务对资源的访问权限。SQS 队列和 SNS: SendMessage 将被添加到针对 SNS 主题的访问权限。

语法

要在您的中声明此实体Amazon Serverless Application Model(Amazon SAM) 模板，请使用以下语法。

YAML

```
  TargetArn: String
  Type: String
```

属性

`TargetArn`

Amazon SQS 队列或 Amazon SNS 主题的 Amazon Resource Name (ARN)。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给`TargetArn`的财产`AWS::Lambda::Function DeadLetterConfig`数据类型。

`Type`

死信队列的类型。

有效值：`SNS`、`SQS`

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

示例

DeadLetter 队列

SNS 主题的死信队列示例。

YAML

```
DeadLetterQueue:
  Type: SNS
  TargetArn: arn:aws:sns:us-east-2:123456789012:my-topic
```

DeploymentPreference

指定配置以启用渐进 Lambda 部署。有关配置渐进 Lambda 部署的更多信息，请参阅逐步部署无服务器应用程序 (p. 460)。

> **Note**
>
> 你必须在你的`AutoPublishAlias`中指定一个AWS::Serverless::Function (p. 129)才能使用`DeploymentPreference`对象，否则会导致错误。

语法

要在Amazon Serverless Application Model ((Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

```
  Alarms: List
  Enabled: Boolean
  Hooks: Hooks (p. 145)
  PassthroughCondition: Boolean
  Role: String
```

```
  TriggerConfigurations: List
  Type: String
```

属性

`Alarms`

您希望由部署引发的任何错误触发的 CloudWatch 警报列表。

此属性接受`Fn::If`内部函数。有关使用的示例模板，请参阅本主题底部的示例部分`Fn::If`。

类型：List

必填项：否

Amazon CloudFormation兼容性：此属性是独一无二的Amazon SAM，没有Amazon CloudFormation等效属性。

`Enabled`

是否启用此部署首选项。

类型：布尔值

必填项：否

默认值：True

Amazon CloudFormation兼容性：此属性是独一无二的Amazon SAM，没有Amazon CloudFormation等效属性。

`Hooks`

验证在流量转移之前和之后运行的 Lambda 函数。

类型：钩子 (p. 145)

必填项：否

Amazon CloudFormation兼容性：此属性是独一无二的Amazon SAM，没有Amazon CloudFormation等效属性。

`PassthroughCondition`

如果为 True，并且启用了此部署首选项，则函数的条件将传递给生成 CodeDeploy 的资源。通常，应将其设置为 True。否则，即使函数的条件解析为 False，也会创建 CodeDeploy 资源。

类型：布尔值

必填项：否

Amazon CloudFormation兼容性：此属性是独一无二的Amazon SAM，没有Amazon CloudFormation等效属性。

`Role`

用于流量转移的 CodeDeploy IAM 角色 ARN。如果提供，则不会创建 IAM 角色。

类型：字符串

必填项：否

Amazon CloudFormation兼容性：此属性是独一无二的Amazon SAM，没有Amazon CloudFormation等效属性。

`TriggerConfigurations`

要与部署组关联的触发器配置列表。用于通知 SNS 主题生命周期事件。

类型：List

必填项：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::CodeDeploy::DeploymentGroup`资源的`TriggerConfigurations`属性。

`Type`

目前有两种类别的部署类型：Linear 和 Canary 有关可用部署类型的更多信息，请参阅逐步部署无服务器应用程序 (p. 460)。

类型：字符串

必填项：是

Amazon CloudFormation兼容性：此属性是独一无二的Amazon SAM，没有Amazon CloudFormation等效属性。

示例

DeploymentPreference 带有交通前后挂钩。

包含流量前后挂钩的部署首选项示例。

YAML

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

DeploymentPreference 使用 Fn

`Fn::If`用于配置警报的部署首选项示例。在本示例中，如果`MyCondition`是，`Alarm1`则进行配置`trueAlarm2`，如果`MyCondition`是，`Alarm5`则进行配置`false`。

YAML

```
DeploymentPreference:
  Enabled: true
  Type: Canary10Percent10Minutes
  Alarms:
    Fn::If:
      - MyCondition
      - - Alarm1
      - - Alarm2
        - Alarm5
```

Hooks

验证在流量转移之前和之后运行的 Lambda 函数。

> **Note**
>
> 此属性中引用的 Lambda 函数配置结果AWS::Lambda::Alias资源的CodeDeployLambdaAliasUpdate对象。有关更多信息，请参阅《Amazon CloudFormation用户指南》中的CodeDeployLambdaAliasUpdate 政策。

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

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

属性

`PostTraffic`

流量转移后运行的 Lambda 函数。

类型：字符串

必填项：否

Amazon CloudFormation兼容性：此属性是独一无二的Amazon SAM，没有Amazon CloudFormation等效属性。

`PreTraffic`

在流量转移之前运行的 Lambda 函数。

类型：字符串

必填项：否

Amazon CloudFormation兼容性：此属性是独一无二的Amazon SAM，没有Amazon CloudFormation等效属性。

示例

挂钩

示例钩子函数

YAML

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

EventInvokeConfiguration

配置选项异步的Lambda 别名或版本调用。

语法

要在你的Amazon Serverless Application Model(Amazon SAM) 模板，请使用以下语法。

YAML

```
  DestinationConfig: EventInvokeDestinationConfiguration (p. 147)
  MaximumEventAgeInSeconds: Integer
  MaximumRetryAttempts: Integer
```

属性

`DestinationConfig`

一个配置对象，用于在 Lambda 处理事件后指定事件目标。

类型：事件调用目标配置 (p. 147)

必需：否

Amazon CloudFormation兼容性：此属性类似于`DestinationConfig`的财产`AWS::Lambda::EventInvokeConfig`资源。SAM 需要一个额外的参数 “类型”，该参数在 CloudFormation 中不存在。

`MaximumEventAgeInSeconds`

Lambda 发送到函数以进行处理的请求的最长期限。

类型：整数

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`MaximumEventAgeInSeconds`的财产`AWS::Lambda::EventInvokeConfig`资源。

`MaximumRetryAttempts`

在函数返回错误之前重试的最大次数。

类型：整数

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`MaximumRetryAttempts`的财产`AWS::Lambda::EventInvokeConfig`资源。

示例

MaximumEventAgeInSeconds

示例：MaximumEventAgeInSeconds

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

一个配置对象，用于在 Lambda 处理事件后指定事件目标。

语法

要在您的Amazon Serverless Application Model(Amazon SAM）模板，请使用以下语法。

YAML

```
  OnFailure: OnFailure (p. 148)
  OnSuccess: OnSuccess (p. 149)
```

属性

`OnFailure`

处理失败的事件的目标。

类型：OnFailure (p. 148)

必需：否

Amazon CloudFormation兼容性：此属性类似于`OnFailure`的财产`AWS::Lambda::EventInvokeConfig`资源。需要`Type`，另外一个仅限 SAME 的酒店。

`OnSuccess`

已成功处理的事件的目标。

类型：OnSuccess (p. 149)

必需：否

Amazon CloudFormation兼容性：此属性类似于`OnSuccess`的财产`AWS::Lambda::EventInvokeConfig`资源。需要`Type`，另外一个仅限 SAME 的酒店。

示例

OnSuccess

OnSuccess 示例

YAML

```
EventInvokeConfig:
  DestinationConfig:
    OnSuccess:
      Type: SQS
      Destination: arn:aws:sqs:us-west-2:012345678901:my-queue
    OnFailure:
      Type: Lambda
      Destination: !GetAtt DestinationLambda.Arn
```

OnFailure

处理失败的事件的目标。

语法

要在您的中声明此实体Amazon Serverless Application Model(Amazon SAM) 模板，请使用以下语法。

YAML

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

属性

`Destination`

目标资源的 Amazon Resource Name (ARN)。

类型：字符串

必需：条件

Amazon CloudFormation兼容性：此属性类似于`OnFailure`的财产`AWS::Lambda::EventInvokeConfig`资源。SAM 将向与此函数关联的自动生成的 IAM 角色添加任何必要的权限，以访问此属性中引用的资源。

附加说明：如果类型是 lambdA/EventBridge，则需要目的地。

`Type`

目标中引用的资源的类型。支持的类型包括`SQS`、`SNS`、`Lambda`, 和`EventBridge`.

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

附加说明：如果类型是 SQS/SNS 且`Destination`属性留空，然后 SAM 自动生成 SQS/SNS 资源。要引用资源，请使用*<function-logical-id>*`.DestinationQueue`对于 SQS 或*<function-logical-id>*`.DestinationTopic`对于 SNS。如果类型是 lambdA/EventBridge，`Destination`是必需的。

示例

具有 SQS 和 Lambda 目标的 EventInvoke 配置示例

在此示例中，没有为 SQS onSuccess 配置指定目标，因此 SAM 隐式创建了 SQS 队列并添加任何必要的权限。另外，在此示例中，在 onFailure 配置中指定了模板文件中声明的 Lambda 资源的目标，因此 SAM 会向此 Lambda 函数添加必要的权限以调用目标 Lambda 函数。

YAML

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

```
      Type: Lambda
      Destination: !GetAtt DestinationLambda.Arn  # Arn of a Lambda function declared in
 the template file.
```

有 SNS 目标的 EventInvoke 配置示例

在此示例中，为 onSuccess 配置的模板文件中声明的 SNS 主题提供了 “目标”。

YAML

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

OnSuccess

已成功处理的事件的目标。

语法

要在您的中声明此实体Amazon Serverless Application Model(Amazon SAM) 模板，请使用以下语法。

YAML

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

属性

`Destination`

目标资源的 Amazon Resource Name (ARN)。

类型：字符串

必需：条件

Amazon CloudFormation兼容性：此属性类似于`OnSuccess`的财产`AWS::Lambda::EventInvokeConfig`资源。SAM 将向与此函数关联的自动生成的 IAM 角色添加任何必要的权限，以访问此属性中引用的资源。

附加说明：如果类型是 lambdA/EventBridge，则需要目的地。

`Type`

目标中引用的资源的类型。支持的类型包括`SQS`、`SNS`、`Lambda`, 和`EventBridge`.

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM而且没有Amazon CloudFormation等效函数。

附加说明：如果类型是 SQS/SNS 且`Destination`属性留空，然后 SAM 自动生成 SQS/SNS 资源。要引用资源，请使用*<function-logical-id>*`.DestinationQueue`对于 SQS 或*<function-logical-id>*`.DestinationTopic`对于 SNS。如果类型是 lambdA/EventBridge，`Destination`是必需的。

示例

具有 SQS 和 Lambda 目标的 EventInvoke 配置示例

在此示例中，没有为 SQS onSuccess 配置指定目标，因此 SAM 隐式创建了 SQS 队列并添加任何必要的权限。另外，在此示例中，在 onFailure 配置中指定了模板文件中声明的 Lambda 资源的目标，因此 SAM 会向此 Lambda 函数添加必要的权限以调用目标 Lambda 函数。

YAML

```
EventInvokeConfig:
  DestinationConfig:
    OnSuccess:
      Type: SQS
    OnFailure:
      Type: Lambda
      Destination: !GetAtt DestinationLambda.Arn  # Arn of a Lambda function declared in
 the template file.
```

有 SNS 目标的 EventInvoke 配置示例

在此示例中，为 onSuccess 配置的模板文件中声明的 SNS 主题提供了 “目标”。

YAML

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

EventSource

描述触发函数的事件来源的对象。每个事件都由一种类型和一组依赖于该类型的属性组成。有关每个事件源的事件源的源的事件源的源的源的事件源的源的配置信息，请参阅适用于每个事件源的事件源的源的事件源的源的源的事件源的源的事件源的源的事件源的源的事件源的

语法

要在模板中声明此实体，请使用以下语法。Amazon Serverless Application ModelAmazon SAM

YAML

```
  Properties: AlexaSkill (p. 151) | Api (p. 152) | CloudWatchEvent (p. 160)
| CloudWatchLogs (p. 162) | Cognito (p. 163) | DocumentDB (p. 164) | DynamoDB (p. 167)
| EventBridgeRule (p. 171) | HttpApi (p. 176) | IoTRule (p. 180) | Kinesis (p. 181)
| MQ (p. 184) | MSK (p. 187) | S3 (p. 190) | Schedule (p. 191) | ScheduleV2 (p. 195)
| SelfManagedKafka (p. 198) | SNS (p. 201) | SQS (p. 204)
  Type: String
```

属性

`Properties`

描述此事件映射属性的对象。属性集必须符合定义的类型。

类型：AlexaSkill (p. 151)| Ap i (p. 152) | CloudWatchEventCloudWatchLogs (p. 162) (p. 160)| Cog (p. 163)nito | Doc (p. 164) umentDB (p. 167) | EventBridgeRule (p. 171)|

HttpApi (p. 176)| iotRule (p. 180) | Kinesis (p. 181) | MQ (p. 184) | MSK (p. 187) | S3 | S (p. 190)cheduleV2 (p. 195) | SelfManagedKafka (p. 198)| SNS (p. 201) | SQS (p. 204) (p. 191)

必需：是

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`Type`

事件类型。

有效
值：`AlexaSkillApiCloudWatchEvent`、`CloudWatchLogsCognito`、`DocumentDB`、`DynamoDBEventBridge`

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

示例

API 活动

使用 API 事件的示例

YAML

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

AlexaSkill

描述`AlexaSkill`事件源类型。

语法

要在您的中声明此实体Amazon Serverless Application Model(Amazon SAM) 模板，请使用以下语法。

YAML

```
  SkillId: String
```

属性

`SkillId`

Alexa 技能的 Alexa 技能 ID。有关 Skill ID 的更多信息，请参阅配置 Lambda 函数的触发器在 Alexa 技能工具包文档中。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

示例

AlexasKillTrigger

Alexa 技能事件示例

YAML

```
AlexaSkillEvent:
  Type: AlexaSkill
```

Api

描述Api事件源类型。如果AWS::Serverless::Api (p. 83)资源已定义，路径和方法值必须与 API 的 OpenAPI 定义中的操作对应。

如果没有AWS::Serverless::Api (p. 83)被定义，函数输入和输出是 HTTP 请求和 HTTP 响应的表示。

例如，使用 JavaScript API，可以通过返回带有 StatusCode 和 body 键的对象来控制响应的状态代码和正文。

语法

要在Amazon Serverless Application Model(Amazon SAM) 模板，请使用以下语法。

YAML

```
  Auth: ApiFunctionAuth (p. 154)
  Method: String
  Path: String
  RequestModel: RequestModel (p. 158)
  RequestParameters: String | RequestParameter (p. 159)
  RestApiId: String
```

属性

`Auth`

此特定 Api + 路径 + 方法的身份验证配置。

对于覆盖 API 很有用`DefaultAuthorizer`否则在单个路径上设置 auth 配置`DefaultAuthorizer`已指定或覆盖默认值`ApiKeyRequired`设置。

类型：APIFFunctionAuth (p. 154)

必需：否

Amazon CloudFormation兼容性：该属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`Method`

调用此函数的 HTTP 方法。

类型：字符串

必需：是

Amazon CloudFormation兼容性：该属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`Path`

调用此函数的 Uri 路径。必须从开始/.

类型：字符串

必需：是

Amazon CloudFormation兼容性：该属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`RequestModel`

请求模型用于此特定的 Api + 路径 + 方法。这应该引用中指定的模型的名称。`Models`的部分AWS::Serverless::Api (p. 83)资源。

类型：RequestModel (p. 158)

必需：否

Amazon CloudFormation兼容性：该属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`RequestParameters`

请求此特定的 Api + 路径 + 方法的参数配置。所有参数名称必须以开头`method.request`并且必须限于`method.request.header`、`method.request.querystring`，或者`method.request.path`.

如果参数是字符串而不是函数请求参数对象，那么`Required`和`Caching`将默认为 false。

类型：String |RestParameters (p. 159)

必需：否

Amazon CloudFormation兼容性：该属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`RestApiId`

RestApi 资源的标识符，该资源必须包含具有给定路径和方法的操作。通常，将其设置为引用AWS::Serverless::Api (p. 83)在此模板中定义的资源。

如果不定义此属性，Amazon SAM创建默认值AWS::Serverless::Api (p. 83)使用生成的资源`OpenApi`文档. 该资源包含所有路径和方法的联合`Api`同一模板中未指定`RestApiId`.

这不能引用AWS::Serverless::Api (p. 83)在另一个模板中定义的资源。

类型：字符串

必需：否

Amazon CloudFormation兼容性：该属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

示例

apeVent

Api 事件的例子

YAML

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

ApiFunctionAuth

在事件级别为特定 API、路径和方法配置授权。

语法

要在您的中声明此实体Amazon Serverless Application Model(Amazon SAM）模板，请使用以下语法。

YAML

```
  ApiKeyRequired: Boolean
  AuthorizationScopes: List
  Authorizer: String
  InvokeRole: String
  ResourcePolicy: ResourcePolicyStatement (p. 155)
```

属性

`ApiKeyRequired`

需要此 API、路径和方法的 API 密钥。

类型：Boolean

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`AuthorizationScopes`

要应用于此 API、路径和方法的授权范围。

您指定的作用域将覆盖由`DefaultAuthorizer`属性（如果你已经指定了它）。

类型：List

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`Authorizer`

这些区域有：`Authorizer`对于特定函数

如果您已在 API 上指定了全局授权者并希望公开特定的函数，请通过设置`Authorizer`到`NONE`.

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`InvokeRole`

指定InvokeRole用于AWS_IAM授权。

类型：字符串

必需：否

默认值：CALLER_CREDENTIALS

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

附加说明：CALLER_CREDENTIALS映射到arn:aws:iam::*:user/*，它使用调用者凭据来调用终端节点。

`ResourcePolicy`

在 API 上为此路径配置资源策略。

类型：资源策略声明 (p. 155)

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

示例

函数身份验证

下面的示例指定了函数级别的授权。

YAML

```
Auth:
  ApiKeyRequired: true
  Authorizer: NONE
```

ResourcePolicyStatement

为 API 的所有方法和路径配置资源策略。有关资源策略的更多信息，请参阅《API Gateway 开发者指南》中的 “使用 API Gateway 资源策略控制 API 的访问”。

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

```
  AwsAccountBlacklist: List
  AwsAccountWhitelist: List
  CustomStatements: List
  IntrinsicVpcBlacklist: List
  IntrinsicVpcWhitelist: List
  IntrinsicVpceBlacklist: List
  IntrinsicVpceWhitelist: List
  IpRangeBlacklist: List
```

```
  IpRangeWhitelist: List
  SourceVpcBlacklist: List
  SourceVpcWhitelist: List
```

属性

`AwsAccountBlacklist`

要屏蔽的Amazon账户。

类型：字符串列表

必需：否

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

`AwsAccountWhitelist`

允许的Amazon账户。有关此属性的示例用法，请参阅本页底部的示例。

类型：字符串列表

必需：否

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

`CustomStatements`

适用于此 API 的自定义资源策略声明列表。有关此属性的示例用法，请参阅本页底部的示例。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

`IntrinsicVpcBlacklist`

要屏蔽的虚拟私有云 (VPC) 列表，其中每个 VPC 都指定为引用，例如动态引用或`Ref`内部函数。有关此属性的示例用法，请参阅本页底部的示例。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

`IntrinsicVpcWhitelist`

允许的 VPC 列表，其中每个 VPC 都指定为引用，例如动态引用或`Ref`内部函数。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

`IntrinsicVpceBlacklist`

要屏蔽的 VPC 终端节点列表，其中每个 VPC 终端节点都指定为引用，例如动态引用或`Ref`内部函数。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

`IntrinsicVpceWhitelist`

允许的 VPC 终端节点列表，其中每个 VPC 终端节点都指定为引用，例如动态引用或`Ref`内部函数。有关此属性的示例用法，请参阅本页底部的示例。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

`IpRangeBlacklist`

要屏蔽的 IP 地址或地址范围。有关此属性的示例用法，请参阅本页底部的示例。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

`IpRangeWhitelist`

允许的 IP 地址或地址范围。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

`SourceVpcBlacklist`

要屏蔽的源 VPC 或 VPC 终端节点。源 VPC 名称必须以开头`"vpc-"`，源 VPC 终端节点名称必须以开头`"vpce-"`。有关此属性的示例用法，请参阅本页底部的示例。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

`SourceVpcWhitelist`

允许的源 VPC 或 VPC 终端节点。源 VPC 名称必须以开头`"vpc-"`，源 VPC 终端节点名称必须以开头`"vpce-"`。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

示例

资源策略示例

以下示例屏蔽了两个 IP 地址和一个源 VPC，并允许一个Amazon账户。

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

为特定的 Api + 路径 + 方法配置请求模型。

语法

要在您的中声明此实体Amazon Serverless Application Model(Amazon SAM) 模板，请使用以下语法。

YAML

```
  Model: String
  Required: Boolean
  ValidateBody: Boolean
  ValidateParameters: Boolean
```

属性

`Model`

在 “模型” 属性中定义的模型的名称AWS::Serverless::Api (p. 83).

类型：字符串

必需：是

Amazon CloudFormation兼容性：该属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`Required`

添加`required`属性位于给定 API 终端节点的 OpenAPI 定义的参数部分。

类型：Boolean

必需：否

Amazon CloudFormation兼容性：该属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`ValidateBody`

指定 API Gateway 是否使用`Model`以验证请求正文。有关更多信息，请参阅 。在 API Gateway 中启用请求验证中的API Gateway 开发人员指南.

类型：Boolean

必需：否

Amazon CloudFormation兼容性：该属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`ValidateParameters`

指定 API Gateway 是否使用`Model`以验证请求路径参数、查询字符串和标头。有关更多信息，请参阅 。在 API Gateway 中启用请求验证中的API Gateway 开发人员指南.

类型：Boolean

必需：否

Amazon CloudFormation兼容性：该属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

示例

请求模型

请求模型示例

YAML

```
RequestModel:
  Model: User
  Required: true
  ValidateBody: true
  ValidateParameters: true
```

RequestParameter

为特定的 Api + 路径 + 方法配置请求参数。

或者`Required`要么`Caching`需要为请求参数指定属性

语法

要在您的中声明此实体Amazon Serverless Application Model(Amazon SAM) 模板，请使用以下语法。

YAML

```
  Caching: Boolean
```

```
  Required: Boolean
```

属性

`Caching`

添加`cacheKeyParameters`一节到 API Gateway OpenAPI 定义

类型：Boolean

必需：条件

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`Required`

此字段指定是否需要参数

类型：Boolean

必需：条件

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

示例

请求参数

设置请求参数的示例

YAML

```
RequestParameters:
  - method.request.header.Authorization:
      Required: true
      Caching: true
```

CloudWatchEvent

描述`CloudWatchEvent`事件源类型的对象。

Amazon Serverless Application Model(Amazon SAM) 在设置此事件类型时生成AWS::Events::Rule资源。

重要说明：EventBridgeRule (p. 171)是首选使用的事件源类型，而不是`CloudWatchEvent`。`EventBridgeRule`并`CloudWatchEvent`使用相同的底层服务、API 和Amazon CloudFormation资源。但是，只Amazon SAM会添加对新功能的支持`EventBridgeRule`。

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

```
  Enabled: Boolean
  EventBusName: String
  Input: String
  InputPath: String
```

```
  Pattern: EventPattern
  State: String
```

属性

`Enabled`

指示是否启用规则。

要禁用规则，请将此属性设置为`false`。

> **Note**
>
> 指定`Enabled`或`State`属性，但不能同时指定两者。

类型：布尔值

必填项：否

Amazon CloudFormation兼容性：此属性类似于`AWS::Events::Rule`资源的`State`属性。如果将此属性设置为 thtrue enAmazon SAM pass`ENABLED`，否则通过`DISABLED`。

`EventBusName`

要与该规则关联的事件总线。如果省略此属性，则Amazon SAM使用默认事件总线。

类型：字符串

必填项：否

默认值：默认事件总线

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Events::Rule`资源的`EventBusName`属性。

`Input`

传递到目标的有效 JSON 文本。如果使用此属性，则不会将事件文本本身的任何内容传递到目标。

类型：字符串

必填项：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Events::Rule Target`资源的`Input`属性。

`InputPath`

当您不希望将整个匹配的事件传递到目标时，使用该`InputPath`属性描述要传递事件的哪个部分。

类型：字符串

必填项：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Events::Rule Target`资源的`InputPath`属性。

`Pattern`

描述哪些事件路由到指定目标。有关更多信息，请参阅 Amazon EventBridge 用户指南 EventBridge中的事件和事件模式。

类型：EventPattern

必填项：是

Amazon CloudFormation兼容性：此属性直接传递给AWS::Events::Rule资源的EventPattern属性。

State

规则的状态。

可接受的值：DISABLED | ENABLED

Note

指定Enabled或State属性，但不能同时指定两者。

类型：字符串

必填项：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Events::Rule资源的State属性。

示例

CloudWatchEvent

以下是CloudWatchEvent事件源类型的示例。

YAML

```
CWEvent:
  Type: CloudWatchEvent
  Properties:
    Enabled: false
    Input: '{"Key": "Value"}'
    Pattern:
      detail:
        state:
          - running
```

CloudWatchLogs

描述CloudWatchLogs事件源类型。

此事件会生成AWS::Logs::SubscriptionFilter资源，并指定订阅筛选器并将其与指定日志组关联。

语法

要在您的Amazon Serverless Application Model(Amazon SAM) 模板，请使用以下语法。

YAML

```
  FilterPattern: String
  LogGroupName: String
```

属性

FilterPattern

限制提交到目标内容的筛选表达式Amazon资源。有关筛选器模式语法的更多信息，请参阅筛选器和模式语法。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给FilterPattern一个的财产AWS::Logs::SubscriptionFilter资源。

`LogGroupName`

要与订阅筛选器关联的日志组。将筛选上传到此日志组的所有日志事件并提交到指定Amazon资源（如果过滤器模式与日志事件匹配）。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给LogGroupName一个的财产AWS::Logs::SubscriptionFilter资源。

示例

订阅筛选器

Cloudwatch 订阅筛选器示例

YAML

```
CWLog:
  Type: CloudWatchLogs
  Properties:
    LogGroupName:
      Ref: CloudWatchLambdaLogsGroup
    FilterPattern: My pattern
```

Cognito

描述Cognito事件源类型。

语法

要在您的中声明此实体Amazon Serverless Application Model(Amazon SAM) 模板，请使用以下语法。

YAML

```
  Trigger: List
  UserPool: String
```

属性

`Trigger`

新用户池的 Lambda 触发器配置信息。

类型：List

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给LambdaConfig的财产AWS::Cognito::UserPool资源。

`UserPool`

对同一模板中定义的 UserPool 的引用

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

示例

Cognito 事件

Cognito 事件示例

YAML

```
CognitoUserPoolPreSignup:
  Type: Cognito
  Properties:
    UserPool:
      Ref: MyCognitoUserPool
    Trigger: PreSignUp
```

DocumentDB

描述DocumentDB事件源类型的对象。有关更多信息，请参阅《Amazon Lambda开发者指南》中的“Amazon Lambda与 Amazon DocumentDB 一起使用”。

语法

要在Amazon SAM模板中声明此实体，请使用以下语法：

YAML

```
BatchSize (p. 164): Integer
Cluster (p. 165): String
CollectionName (p. 165): String
DatabaseName (p. 165): String
Enabled (p. 165): Boolean
FilterCriteria (p. 165): FilterCriteria
FullDocument (p. 165): String
MaximumBatchingWindowInSeconds (p. 166): Integer
SecretsManagerKmsKeyId (p. 166): String
SourceAccessConfigurations (p. 166): List
StartingPosition (p. 166): String
StartingPositionTimestamp (p. 167): Double
```

属性

`BatchSize`

要在单个批次中检索的最大项目数。

类型：整数

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的BatchSize属性。

`Cluster`

Amazon DocumentDB 集群的 Amazon 资源名称（ARN）。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的EventSourceArn属性。

`CollectionName`

要在数据库中使用的集合的名称。如果您未指定集合，Lambda 会使用所有集合。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMappingDocumentDBEventSourceConfig数据类型的CollectionName属性。

`DatabaseName`

要在 Amazon DocumentDB 集群中使用的数据库的名称。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMappingDocumentDBEventSourceConfig数据类型的DatabaseName属性。

`Enabled`

如果为true，则事件源映射处于活动状态。要暂停轮询和调用，请设置为false。

类型：布尔值

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的Enabled属性。

`FilterCriteria`

定义用于确定 Lambda 是否应处理事件的条件的对象。有关更多信息，请参阅Amazon Lambda开发者指南中的 Lambda 事件筛选。

类型：FilterCriteria

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的FilterCriteria属性。

`FullDocument`

确定 Amazon DocumentDB 在文档更新操作期间向您的事件流发送的内容。如果设置为UpdateLookup，Amazon DocumentDB 将发送描述更改的增量以及整个文档的副本。否则，Amazon DocumentDB 仅发送包含更改的部分文档。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMappingDocumentDBEventSourceConfig数据类型的FullDocument属性。

MaximumBatchingWindowInSeconds

在调用函数之前收集记录的最长时间（以秒为单位）。

类型：整数

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的MaximumBatchingWindowInSeconds属性。

SecretsManagerKmsKeyId

来自 Secrets Manager 的客户管理密钥的Amazon Key Management Service (Amazon KMS)Amazon密钥 ID。当您使用 Secrets Manager 的客户托管密钥时，该密钥具有不包含kms:Decrypt权限的 Lambda 执行角色，则为必填项。

此属性的值是一个 UUID。例如：1abc23d4-567f-8ab9-cde0-1fab234c5d67。

类型：字符串

必需：条件

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

SourceAccessConfigurations

身份验证协议或虚拟主机的数组。使用SourceAccessConfigurations数据类型指定此项。

对于DocumentDB事件源类型，唯一有效的配置类型是BASIC_AUTH。

- BASIC_AUTH— 存储代理凭证的 Secrets Manager 密钥。对于此类型，凭证必须为以下格式：{"username": "your-username", "password": "your-password"}。只允许BASIC_AUTH一个类型的对象。

类型：清单

必需：是

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的SourceAccessConfigurations属性。

StartingPosition

在流中开始读取数据的位置。

- AT_TIMESTAMP— 指定开始读取记录的时间。
- LATEST— 仅读取新记录。
- TRIM_HORIZON— 处理所有可用的记录。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的StartingPosition属性。

`StartingPositionTimestamp`

开始读取的时间（以 Unix 时间秒为单位）。定义`StartingPositionTimestamp`何时指定`StartingPosition`为`AT_TIMESTAMP`。

类型：双精度

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Lambda::EventSourceMapping`资源的`StartingPositionTimestamp`属性。

示例

Amazon DocumentDB ument

```
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

描述DynamoDB事件源类型的对象。有关更多信息，请参阅《Amazon Lambda开发者指南》中的 “Amazon Lambda与 Amazon DynamoDB 一起使用”。

Amazon SAM设置此事件类型时生成AWS::Lambda::EventSourceMapping资源。

语法

要使用以下语Amazon SAM法，请使用以下语法。Amazon Serverless Application Model

YAML

```
  BatchSize: Integer
  BisectBatchOnFunctionError: Boolean
  DestinationConfig: DestinationConfig
  Enabled: Boolean
  FilterCriteria: FilterCriteria
  FunctionResponseTypes: List
  MaximumBatchingWindowInSeconds: Integer
  MaximumRecordAgeInSeconds: Integer
  MaximumRetryAttempts: Integer
  ParallelizationFactor: Integer
```

```
  StartingPosition: String
  StartingPositionTimestamp (p. 170): Double
  Stream: String
  TumblingWindowInSeconds: Integer
```

属性

`BatchSize`

要在单个批次中检索的最大项目数。

类型：整数

必需：否

默认值：100

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Lambda::EventSourceMapping`资源的`BatchSize`属性。

最小值：1

最大值：1000

`BisectBatchOnFunctionError`

函数返回错误，则将批次拆分为两批并重试。

类型：布尔值

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Lambda::EventSourceMapping`资源的`BisectBatchOnFunctionError`属性。

`DestinationConfig`

用于丢弃记录的Amazon Simple Queue Service (Amazon SQS) ation Simple Notification Service (Amazon SNS) 的主题目标。

类型：DestinationConfig

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Lambda::EventSourceMapping`资源的`DestinationConfig`属性。

`Enabled`

禁用事件源映射以暂停轮询和调用。

类型：布尔值

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Lambda::EventSourceMapping`资源的`Enabled`属性。

`FilterCriteria`

用于确定 Lambda 是否应处理事件的标准。有关更多信息，请参阅《Amazon Lambda开发者指南》中的Amazon Lambda事件筛选。

类型：FilterCriteria

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的FilterCriteria属性。

FunctionResponseTypes

当前应用于事件源映射的响应类型列表。有关更多信息，请参阅Amazon Lambda开发者指南中的报告批处理项目故障。

有效值：ReportBatchItemFailures

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的FunctionResponseTypes属性。

MaximumBatchingWindowInSeconds

在调用函数之前收集记录的最长时间（以秒为单位）。

类型：整数

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的MaximumBatchingWindowInSeconds属性。

MaximumRecordAgeInSeconds

Lambda 发送到函数以进行处理的记录的最长期限。

类型：整数

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的MaximumRecordAgeInSeconds属性。

MaximumRetryAttempts

在函数返回错误时重试的最大次数。

类型：整数

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的MaximumRetryAttempts属性。

ParallelizationFactor

每个分区中同时处理的批次数。

类型：整数

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的ParallelizationFactor属性。

StartingPosition

在流中开始读取数据的位置。

- AT_TIMESTAMP— 指定开始读取记录的时间。
- LATEST— 仅读取新记录。
- TRIM_HORIZON— 处理所有可用的记录。

有效值：AT_TIMESTAMP | LATEST | TRIM_HORIZON

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的StartingPosition属性。

StartingPositionTimestamp

开始读取的时间（Unix 时间秒为单位）。定义StartingPositionTimestamp何时指定StartingPosition为AT_TIMESTAMP。

类型：双精度

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的StartingPositionTimestamp属性。

Stream

DynamoDB NS 流的 Amazon 资源名称（ARN）。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的EventSourceArn属性。

TumblingWindowInSeconds

处理窗口的持续时间（以秒为单位）。有效范围为 1 到 900（15 分钟）。

有关更多信息，请参阅《Amazon Lambda开发者指南》中的 Tumbling Windows。

类型：整数

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的TumblingWindowInSeconds属性。

示例

现有 DynamoDB 表的 DynamoDB 事件源

Amazon账户中已存在的 DynamoDB 表的 DynamoDB 事件源。

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
```

在模板中声明的 DynamoDB 表的 DynamoDB 事件

在同一模板文件中声明的 DynamoDB 表的 DynamoDB 事件。

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

描述`EventBridgeRule`事件源类型的对象，它将您的无服务器函数设置为 Amazon EventBridge 规则的目标。有关更多信息，请参阅 Amazon EventBridge？ 在亚马逊 EventBridge 用户指南中。

Amazon SAM设置此事件类型时生成AWS::Events::Rule资源。

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

```
  DeadLetterConfig: DeadLetterConfig (p. 174)
  EventBusName: String
  Input: String
  InputPath: String
  Name: String
  Pattern: EventPattern
  RetryPolicy: RetryPolicy
  State (p. 173): String
  Target: Target (p. 175)
```

属性

`DeadLetterConfig`

配置 Amazon Simple Que Service (Amazon SQS) 队列， EventBridge 将事件发送到目标。例如，当向不存在的 Lambda 函数发送事件时，或者当权限不足以调用 Lambda 函数时 EventBridge ，调用可能会失败。有关更多信息，请参阅 Amazon EventBridge 用户指南中的 Event 重试策略和使用死信队列。

> **Note**
>
> AWS::Serverless::Function (p. 129)资源类型具有类似的数据类型`DeadLetterQueue`，用于处理成功调用目标 Lambda 函数后发生的故障。此类故障的示例包括 Lambda 限制或 Lambda

目标函数返回的错误。有关函数DeadLetterQueue属性的更多信息，请参阅《Amazon Lambda开发者指南》中的Amazon Lambda函数死信队列。

类型：DeadLetterConfig (p. 174)

必需：否

Amazon CloudFormation兼容性：此属性类似于AWS::Events::RuleTarget数据类型的DeadLetterConfig属性。此属性的Amazon SAM版本包括其他子属性，以防你Amazon SAM想为你创建死信队列。

`EventBusName`

要与该规则关联的事件总线。如果省略此属性，则Amazon SAM使用默认事件总线。

类型：字符串

必需：否

默认：默认事件总线

Amazon CloudFormation兼容性：此属性直接传递给AWS::Events::Rule资源的EventBusName属性。

`Input`

传递到目标的有效 JSON 文本。如果使用此属性，则不会将事件文本本身的任何内容传递到目标。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Events::Rule Target资源的Input属性。

`InputPath`

当您不希望传递整个匹配的事件到目标，使用InputPath属性描述将事件的哪个部分传递到目标。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Events::Rule Target资源的InputPath属性。

`Name`

规则的名称。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Events::Rule资源的Name属性。

`Pattern`

描述哪些事件路由到指定目标。有关更多信息，请参阅 Amazon EventBridge 用户指南中的 Events and EventBridge Event Patterns。

类型：EventPattern

必需：是

Amazon CloudFormation兼容性：此属性直接传递给AWS::Events::Rule资源的EventPattern属性。

RetryPolicy

包含有关重试策略设置的信息的 RetryPolicy 对象。有关更多信息，请参阅 Amazon EventBridge 用户指南中的 Event 重试策略和使用死信队列。

类型：RetryPolicy

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Events::RuleTarget数据类型的RetryPolicy属性。

State

规则的状态。

可接受的值：DISABLED | ENABLED

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Events::Rule资源的State 属性。

Target

触发规则时 EventBridge 调用的Amazon资源。您可以使用此属性指定目标的逻辑 ID。如果未指定此属性，则Amazon SAM生成目标的逻辑 ID。

类型：目标 (p. 175)

必需：否

Amazon CloudFormation兼容性：此属性类似于AWS::Events::Rule资源的Targets属性。此属性的Amazon SAM版本仅允许您指定单个目标的逻辑 ID。

示例

EventBridgeRule

以下是EventBridgeRule事件源类型的示例。

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

用于指定Amazon Simple Queue Service (Amazon SQS) 失败后 EventBridge 发送事件的对象。例如，向不存在的 Lambda 函数发送事件，或者权限不足以调用 Lambda 函数时，调用可能会失败。有关更多信息，请参阅 Azon EventBridge 用户指南中的事件重试策略和使用死信队列。

Note

AWS::Serverless::Function (p. 129)资源类型具有相似的数据类型DeadLetterQueue，用于处理成功调用目标 Lambda 函数后发生的故障。此类故障的示例包括 Lambda 限制或 Lambda 目标函数返回的错误。有关函数DeadLetterQueue属性的更多信息，请参阅Amazon Lambda开发者指南中的Amazon Lambda函数死信队列。

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

```
Arn: String
QueueLogicalId: String
Type: String
```

属性

`Arn`

指定作为死信队列的目标Amazon SQS 资源名称（ARN）。

Note

指定TypeArn属性或，但不能同时指定两者。

类型：字符串

必填项：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Events::RuleDeadLetterConfig数据类型的Arn属性。

`QueueLogicalId`

如果已指定，则Amazon SAM创建的死信队列Type的自定义名称。

Note

如果未设置该Type属性，将忽略该属性。

类型：字符串

必填项：否

Amazon CloudFormation兼容性：此属性是独一无二的Amazon SAM，没有Amazon CloudFormation等效属性。

`Type`

队列。设置此属性后，Amazon SAM会自动创建死信队列并附加必要的基于资源的策略，以授予规则资源向队列发送事件的权限。

Note

指定TypeArn属性或，但不能同时指定两者。

有效值：SQS

类型：字符串

必填项：否

Amazon CloudFormation兼容性：此属性是独一无二的Amazon SAM，没有Amazon CloudFormation等效属性。

示例

DeadLetterConfig

DeadLetterConfig

YAML

```
DeadLetterConfig:
  Type: SQS
  QueueLogicalId: MyDLQ
```

Target

配置触发规则时 EventBridge 调用的Amazon资源。

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

```
  Id: String
```

属性

`Id`

目标的逻辑 ID。

的值`Id`可以包含字母数字字符、句点 (`.`)、连字符 (`-`) 和下划线 (`_`)。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Events::RuleTarget`数据类型的`Id`属性。

示例

目标

YAML

```
EBRule:
  Type: EventBridgeRule
  Properties:
```

```
    Target:
      Id: MyTarget
```

HttpApi

用 HTTPAPI 类型描述事件源的对象。

如果 API 上存在指定路径和方法的 OpenAPI 定义，SAM 将为您添加 Lambda 集成和安全部分（如果适用）。

如果 API 上没有指定路径和方法的 OpenAPI 定义，SAM 将为您创建此定义。

语法

要在您的中声明此实体Amazon Serverless Application Model(Amazon SAM) 模板，请使用以下语法。

YAML

```
ApiId: String
Auth: HttpApiFunctionAuth (p. 178)
Method: String
Path: String
PayloadFormatVersion: String
RouteSettings: RouteSettings
TimeoutInMillis: Integer
```

属性

`ApiId`

的标识符AWS::Serverless::HttpApi (p. 208)在此模板中定义的资源。

如果没有定义，则默认值AWS::Serverless::HttpApi (p. 208)资源被创建名为`ServerlessHttpApi`使用生成的 OpenAPI 文档，其中包含由此模板中定义的未指定`ApiId`.

这不能引用AWS::Serverless::HttpApi (p. 208)在另一个模板中定义的资源。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM没有Amazon CloudFormation等效项

`Auth`

此特定 Api + 路径 + 方法的身份验证配置。

对于覆盖 API 很有用`DefaultAuthorizer`或者在没有时在单个路径上设置 auth 配置`DefaultAuthorizer`已指定。

类型：HTTPIFF 函数身份验证 (p. 178)

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM没有Amazon CloudFormation等效项

`Method`

调用此函数的 HTTP 方法。

如果没有`Path`和`Method`，SAM 将创建一个默认 API 路径，该路径将任何未映射到其他终端节点的请求路由到此 Lambda 函数。每个 API 只能存在其中一个默认路径。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM没有Amazon CloudFormation等效项

`Path`

调用此函数的 Uri 路径。必须从开始/.

如果没有`Path`和`Method`，SAM 将创建一个默认 API 路径，该路径将任何未映射到其他终端节点的请求路由到此 Lambda 函数。每个 API 只能存在其中一个默认路径。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM没有Amazon CloudFormation等效项

`PayloadFormatVersion`

指定发送到集成的负载的格式。

注意：PayloadFormatVersion 要求 SAM 修改你的 OpenAPI 定义，因此它只适用于在`DefinitionBody`财产。

类型：字符串

必需：否

默认值：2.0

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM没有Amazon CloudFormation等效项

`RouteSettings`

此 HTTP API 的每路由路由设置。有关路由设置的更多信息，请参阅AWS::ApiGatewayV2::Stage RouteSettings中的API Gateway 开发人员指南.

注意：如果在 HTTPAPI 资源和事件源中都指定了 RouteSettings，Amazon SAM将它们与事件源属性合并优先。

类型：RouteSettings

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`RouteSettings`的财产`AWS::ApiGatewayV2::Stage`资源。

`TimeoutInMillis`

自定义超时值，范围在 50 到 29,000 毫秒之间。

注意：TimeoutInMillis 要求 SAM 修改你的 OpenAPI 定义，因此它只适用于在`DefinitionBody`财产。

类型：整数

必需：否

默认值：5000

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM没有Amazon CloudFormation等效项

示例

默认 HTTPi 事件

使用默认路径的 HTTPi 事件。此 API 上的所有未映射路径和方法都将路由到此终端节点。

YAML

```
Events:
  HttpApiEvent:
    Type: HttpApi
```

httPapi

使用特定路径和方法的 HTTPAPI 事件。

YAML

```
Events:
  HttpApiEvent:
    Type: HttpApi
    Properties:
      Path: /
      Method: GET
```

httPAPI 授权

使用授权方的 HTTPAPI 事件。

YAML

```
Events:
  HttpApiEvent:
    Type: HttpApi
    Properties:
      Path: /authenticated
      Method: GET
      Auth:
        Authorizer: OpenIdAuth
        AuthorizationScopes:
          - scope1
          - scope2
```

HttpApiFunctionAuth

在事件级别配置授权。

为特定 API + 路径 + 方法配置身份验证

语法

要在您的中声明此实体Amazon Serverless Application Model(Amazon SAM) 模板，请使用以下语法。

YAML

```
  AuthorizationScopes: List
  Authorizer: String
```

属性

`AuthorizationScopes`

要应用于此 API、路径和方法的授权范围。

此处列出的作用域将覆盖`DefaultAuthorizer`如果存在。

类型：列表

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项。

`Authorizer`

这些区域有：`Authorizer`对于特定函数。要使用 IAM 授权，请指定`AWS_IAM`然后指定`true`为了`EnableIamAuthorizer`中的`Globals`模板的部分。

如果您已在 API 上指定了全局授权者并希望公开特定的函数，请通过设置`Authorizer`到`NONE`.

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项。

示例

函数身份验证

在函数级别指定授权

YAML

```
Auth:
  Authorizer: OpenIdAuth
  AuthorizationScopes:
    - scope1
    - scope2
```

IAM 授权

指定事件级别的 IAM 授权。使用`AWS_IAM`在活动级别授权，您还必须指定`true`为了`EnableIamAuthorizer`中的`Globals`模板的部分。有关更多信息，请参阅 Amazon SAM模板的全局变量部分 (p. 79)。

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

描述IoTRule事件源类型。

创建AWS::IoT::TopicRule可以使用资源声明Amazon IoT规则。有关更多信息，请参阅Amazon CloudFormation文档

语法

要将此实体声明您的Amazon Serverless Application Model(Amazon SAM）模板，请使用以下语法。

YAML

```
  AwsIotSqlVersion: String
  Sql: String
```

属性

`AwsIotSqlVersion`

评估规则时使用的 SQL 规则引擎的版本。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给AwsIotSqlVersion的财产`AWS::IoT::TopicRule TopicRulePayload`资源。

`Sql`

用于查询主题的 SQL 语句。有关更多信息，请参阅 。Amazon IoTSQL 参考中的Amazon IoT开发人员指南.

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给Sql的财产`AWS::IoT::TopicRule TopicRulePayload`资源。

示例

物联网规则

示例物联网规则

YAML

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

Kinesis

描述Kinesis事件源类型的对象。有关更多信息，请参阅《Amazon Lambda开发者指南》中的 “在 Amazon LambdaAmazon Kinesis 上使用”。

Amazon SAM设置此事件类型时生成AWS::Lambda::EventSourceMapping资源。

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

```
  BatchSize: Integer
  BisectBatchOnFunctionError: Boolean
  DestinationConfig: DestinationConfig
  Enabled: Boolean
  FilterCriteria: FilterCriteria
  FunctionResponseTypes: List
  MaximumBatchingWindowInSeconds: Integer
  MaximumRecordAgeInSeconds: Integer
  MaximumRetryAttempts: Integer
  ParallelizationFactor: Integer
  StartingPosition: String
  StartingPositionTimestamp (p. 183): Double
  Stream: String
  TumblingWindowInSeconds: Integer
```

属性

`BatchSize`

要在单个批次中检索的最大项目数。

类型：整数

必需：否

默认值：100

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Lambda::EventSourceMapping`资源的`BatchSize`属性。

最小值：1

最大值：10000

`BisectBatchOnFunctionError`

如果函数返回错误，则将批并重试。

类型：布尔值

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的BisectBatchOnFunctionError属性。

`DestinationConfig`

Amazon Simple Queue Service (Amazon SQS) 或存放丢弃记录的Simple Notification Service (Amazon SNS) otification Service

类型：DestinationConfig

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的DestinationConfig属性。

`Enabled`

禁用事件源映射以暂停轮询和调用。

类型：布尔值

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的Enabled属性。

`FilterCriteria`

定义用于确定 Lambda 是否应处理事件的标准的对象。有关更多信息，请参阅《Amazon Lambda开发者指南》中的Amazon Lambda事件筛选。

类型：FilterCriteria

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的FilterCriteria属性。

`FunctionResponseTypes`

当前应用于事件源映射的响应类型列表。有关更多信息，请参阅Amazon Lambda开发者指南中的报告批处理项目故障。

有效值：ReportBatchItemFailures

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的FunctionResponseTypes属性。

`MaximumBatchingWindowInSeconds`

在调用函数之前收集记录的最长时间（以秒为单位）。

类型：整数

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的MaximumBatchingWindowInSeconds属性。

`MaximumRecordAgeInSeconds`

Lambda 发送到函数以进行处理的记录的最长期限。

类型：整数

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的MaximumRecordAgeInSeconds属性。

`MaximumRetryAttempts`

在函数返回错误时重试的最大次数。

类型：整数

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的MaximumRetryAttempts属性。

`ParallelizationFactor`

要从每个分区中同时处理的批并数。

类型：整数

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的ParallelizationFactor属性。

`StartingPosition`

在流中开始读取数据的位置。

- AT_TIMESTAMP— 指定开始读取记录的时间。
- LATEST— 仅读取新记录。
- TRIM_HORIZON— 处理所有可用的记录。

有效值：AT_TIMESTAMP | LATEST | TRIM_HORIZON

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的StartingPosition属性。

`StartingPositionTimestamp`

开始读取的时间（以 Unix 时间秒为单位。定义StartingPositionTimestamp何时指定StartingPosition为AT_TIMESTAMP。

类型：双精度

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的StartingPositionTimestamp属性。

`Stream`

数据流的 Amazon Rcce (ARN)。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的EventSourceArn属性。

TumblingWindowInSeconds

处理窗口的持续时间（以秒为单位）。有效范围为 1 到 900（15 分钟）。

有关更多信息，请参阅《Amazon Lambda开发者指南》中的 Tumbling Windows。

类型：整数

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的TumblingWindowInSeconds属性。

示例

Kinesis 事件源

以下是 Kinesis 事件源的示例。

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

描述MQ事件源类型的对象。有关更多信息，请参阅Amazon Lambda开发者指南中的 Lambda 与 Amazon MQ 配合使用。

Amazon Serverless Application Model(Amazon SAM) 在设置此事件类型时生成AWS::Lambda::EventSourceMapping资源。

Note

要在虚拟私有云 (VPC) 中建立 Amazon MQ 队列并连接到公共网络中的 Lambda 函数，您的函数的执行角色必须包含以下权限：

- ec2:CreateNetworkInterface
- ec2:DeleteNetworkInterface
- ec2:DescribeNetworkInterfaces
- ec2:DescribeSecurityGroups
- ec2:DescribeSubnets
- ec2:DescribeVpcs

有关更多信息，请参阅《Amazon Lambda开发者指南》中的执行角色权限。

语法

要在Amazon SAM模板中声明此实体，请使用以下语法：

YAML

```
  BatchSize: Integer
  Broker: String
  DynamicPolicyName (p. 185): Boolean
  Enabled: Boolean
  FilterCriteria: FilterCriteria
  MaximumBatchingWindowInSeconds: Integer
  Queues: List
  SecretsManagerKmsKeyId: String
  SourceAccessConfigurations: List
```

属性

`BatchSize`

要在单个批次中检索的最大项目数。

类型：整数

必需：否

默认值：100

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Lambda::EventSourceMapping`资源的`BatchSize`属性。

最小值：1

最大值：`10000`

`Broker`

Amazon MQ 代理的 Amazon 资源名称（ARN）。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Lambda::EventSourceMapping`资源的`EventSourceArn`属性。

`DynamicPolicyName`

默认情况下，Amazon Identity and Access Management(IAM) 策略名称是`SamAutoGeneratedAMQPolicy`为了向后兼容。指定`true`您的 IAM 策略使用自动生成的名称。此名称将包含Amazon MQ 事件源逻辑 ID。

> **Note**
>
> 使用多个 Amazon MQ 事件源时，请指定`true`以避免 IAM 策略名称重复。

类型：布尔值

必需：否

默认值：`false`

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`Enabled`

如果`true`，则事件源映射处于活动状态。要暂停轮询和调用，请设置为`false`。

类型：布尔值

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Lambda::EventSourceMapping`资源的`Enabled`属性。

`FilterCriteria`

定义用于确定 Lambda 是否应处理事件的条件的对象。有关更多信息，请参阅《Amazon Lambda开发者指南》中的Amazon Lambda事件筛选。

类型：FilterCriteria

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Lambda::EventSourceMapping`资源的`FilterCriteria`属性。

`MaximumBatchingWindowInSeconds`

在调用函数之前收集记录的最长时间（以秒为单位）。

类型：整数

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Lambda::EventSourceMapping`资源的`MaximumBatchingWindowInSeconds`属性。

`Queues`

要使用的 Amazon MQ 代理目标队列的名称。

类型：清单

必需：是

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Lambda::EventSourceMapping`资源的`Queues`属性。

`SecretsManagerKmsKeyId`

来自的客户管理密钥的Amazon Key Management Service (Amazon KMS) 密钥 IDAmazon Secrets Manager。当您使用 Secrets Manager 的客户托管密钥时，该密钥具有不包含`kms:Decrypt`权限的 Lambda 执行角色，则为必填项。

此属性的值是一个 UUID。例如：`1abc23d4-567f-8ab9-cde0-1fab234c5d67`。

类型：字符串

必需：条件

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`SourceAccessConfigurations`

虚拟主机的身份验证协议数组。使用SourceAccessConfigurations数据类型指定此项。

对于MQ事件源类型，唯一有效的配置类型是BASIC_AUTH和VIRTUAL_HOST。

- **BASIC_AUTH**— 存储您的代理凭证的 Secrets Manager 密钥。对于此类型，凭证必须为以下{"username": "your-username", "password": "your-password"}格式：只允许BASIC_AUTH一个类型的对象。
- **VIRTUAL_HOST**— 您的 RabbbMQ 代理中虚拟主机的名称。Lambda 将使用此 Rabbda 的 Rabbda 主作为事件源。只允许VIRTUAL_HOST一个类型的对象。

类型：清单

必需：是

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的SourceAccessConfigurations属性。

示例

Amazon MQ 事件源

以下是 Amazon MQ 代理MQ的事件源类型的示例。

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

描述MSK事件源类型的对象。有关更多信息，请参阅《Amazon Lambda开发者指南》Amazon Lambda中的“在 Amazon MSK 上使用”。

Amazon Serverless Application Model(Amazon SAM) 在设置此事件类型时生成AWS::Lambda::EventSourceMapping资源。

语法

要在Amazon SAM模板中声明此实体，请使用以下语法。

YAML

```
  ConsumerGroupId: String
  FilterCriteria: FilterCriteria
  MaximumBatchingWindowInSeconds: Integer
  SourceAccessConfigurations (p. 188): SourceAccessConfigurations
  StartingPosition: String
  StartingPositionTimestamp (p. 189): Double
  Stream: String
  Topics: List
```

属性

`ConsumerGroupId`

一个字符串，用于配置如何从 Kafka 主题读取事件。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Lambda::EventSourceMapping`资源的`AmazonManagedKafkaConfiguration`属性。

`FilterCriteria`

定义用于确定 Lambda 是否应处理事件的标准的对象。有关更多信息，请参阅《Amazon Lambda开发者指南》中的Amazon Lambda事件筛选。

类型：FilterCriteria

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Lambda::EventSourceMapping`资源的`FilterCriteria`属性。

`MaximumBatchingWindowInSeconds`

在调用函数之前收集记录的最长时间（以秒为单位）。

类型：整数

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Lambda::EventSourceMapping`资源的`MaximumBatchingWindowInSeconds`属性。

`SourceAccessConfigurations`

用于保护与定义事件源的身份验证协议数组 VPC 组件或虚拟化主机。

有效值：`CLIENT_CERTIFICATE_TLS_AUTH`

类型：SourceAccessConfiguration 列表

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Lambda::EventSourceMapping`资源的`SourceAccessConfigurations`属性。

`StartingPosition`

在流中开始读取数据的位置。

- `AT_TIMESTAMP`— 指定开始读取记录的时间。
- `LATEST`— 仅读取新记录。
- `TRIM_HORIZON`— 处理所有可用的记录。

有效值：`AT_TIMESTAMP` | `LATEST` | `TRIM_HORIZON`

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Lambda::EventSourceMapping`资源的`StartingPosition`属性。

`StartingPositionTimestamp`

开始读取的时间（以 Unix 时间秒为单位）。定义`StartingPositionTimestamp`何时指定`StartingPosition`为`AT_TIMESTAMP`。

类型：双精度

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Lambda::EventSourceMapping`资源的`StartingPositionTimestamp`属性。

`Stream`

数据流的 Amazon 资源名称 (ARN)。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Lambda::EventSourceMapping`资源的`EventSourceArn`属性。

`Topics`

Kafka 主题的名称。

类型：清单

必需：是

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Lambda::EventSourceMapping`资源的`Topics`属性。

示例

现有集群的 Amazon MSK 示例

以下是已存在于 Amazon MSK 集群`MSK`的事件源类型的示例Amazon Web Services 账户。

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
```

在同一模板中声明的集群的 Amazon MSK 示例

以下是在同一模板文件中声明的 Amazon MSK 集群`MSK`的事件源类型的示例。

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

描述S3事件源类型。

语法

要在您的Amazon Serverless Application Model(Amazon SAM) 模板，请使用以下语法。

YAML

```
Bucket: String
Events: String | List
Filter: NotificationFilter
```

属性

`Bucket`

S3 存储桶名称。此存储桶必须存在于同一模板中。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性类似于`BucketName`一个的财产`AWS::S3::Bucket`资源。这是 SAM 中的必填字段。此字段仅接受对在此模板中创建的 S3 存储桶的引用

`Events`

调用 Lambda 函数的 Amazon S3 存储桶事件。请参阅Amazon S3 支持的事件类型有关有效值的列表。

类型：字符串 | 列表

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给`Event`的财产`AWS::S3::Bucket LambdaConfiguration`数据类型。

`Filter`

确定哪些 Amazon S3 对象调用 Lambda 函数的筛选规则。有关 Amazon S3 密钥名称筛选的信息，请参阅配置 Amazon S3 事件通知中的Amazon Simple Storage Service 用户指南.

类型：NotificationFilter

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`Filter`的财产`AWS::S3::Bucket LambdaConfiguration`数据类型。

示例

S3 事件

S3 事件示例。

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

描述Schedule事件源类型的对象，它将您的无服务器函数设置为按计划触发的 EventBridge 规则的目标。有关更多信息，请参阅 Amazon 是什么 EventBridge？ 在亚马逊 EventBridge 用户指南中。

Amazon Serverless Application Model(Amazon SAM) 在设置此事件类型时生成AWS::Events::Rule资源。

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

```
  DeadLetterConfig: DeadLetterConfig (p. 193)
  Description: String
  Enabled: Boolean
  Input: String
  Name: String
  RetryPolicy: RetryPolicy
  Schedule: String
  State: String
```

属性

`DeadLetterConfig`

配置 Amazon Simple Queue Service (Amazon SQS) 队列，此队列将在目标调用失败后 EventBridge 发送事件。例如，当向不存在的 Lambda 函数发送事件时，或者当权限不足以调用 Lambda 函数时 EventBridge ，调用可能会失败。有关更多信息，请参阅 Amazon EventBridge 用户指南中的事件重试策略和使用死信队列。

> **Note**
>
> AWS::Serverless::Function (p. 129)资源类型具有相似的数据类型`DeadLetterQueue`，用于处理成功调用目标 Lambda 函数后发生的故障。此类故障的示例包括 Lambda 限制或 Lambda 目标函数返回的错误。有关函数`DeadLetterQueue`属性的更多信息，请参阅《Amazon Lambda开发者指南》中的Amazon Lambda函数死信队列。

类型：DeadLetterConfig (p. 193)

必填项：否

Amazon CloudFormation兼容性：此属性类似于`AWS::Events::RuleTarget`数据类型的`DeadLetterConfig`属性。此属性的Amazon SAM版本包括其他子属性，Amazon SAM以备您想要创建死信队列时使用。

`Description`

规则的描述。

类型：字符串

必填项：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Events::Rule`资源的`Description`属性。

`Enabled`

指示是否启用规则。

要禁用该规则，请将此属性设置为`false`。

> Note
>
> 指定`Enabled`或`State`属性，但不能同时指定两者。

类型：布尔值

必填项：否

Amazon CloudFormation兼容性：此属性类似于`AWS::Events::Rule`资源的`State`属性。如果将此属性设置为 thtrue enAmazon SAM pass`ENABLED`，否则通过`DISABLED`。

`Input`

传递到目标的有效 JSON 文本。如果使用此属性，则不会将事件文本本身的任何内容传递到目标。

类型：字符串

必填项：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Events::Rule Target`资源的`Input`属性。

`Name`

规则的名称。如果不指定名称，则 Amazon CloudFormation 生成一个唯一物理 ID 并将该 ID 用作规则名称。

类型：字符串

必填项：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Events::Rule`资源的`Name`属性。

`RetryPolicy`

包含有关重试策略设置的信息的 `RetryPolicy` 对象。有关更多信息，请参阅 Amazon EventBridge 用户指南中的事件重试策略和使用死信队列。

类型：RetryPolicy

必填项：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Events::RuleTarget`数据类型的`RetryPolicy`属性。

`Schedule`

决定运行规则的时间和频率的计划表达式。有关更多信息，请参阅规则的计划表达式。

类型：字符串

必填项：是

Amazon CloudFormation兼容性：此属性直接传递给AWS::Events::Rule资源的ScheduleExpression属性。

State

规则的状态。

可接受的值：DISABLED | ENABLED

Note

指定Enabled或State属性，但不能同时指定两者。

类型：字符串

必填项：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Events::Rule资源的State属性。

示例

CloudWatch 计划事件

CloudWatch 计划事件示例

YAML

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

DeadLetterConfig

用于指定目标调用失败后 EventBridge 发送事件的 Amazon Simple Queueueue Service (Amazon SQS) 队列。例如，向不存在的 Lambda 函数发送事件，或者权限不足以调用 Lambda 函数时，调用可能会失败。有关更多信息，请参阅 Amazon EventBridge 用户指南中的事件重试策略和使用死信队列。

Note

AWS::Serverless::Function (p. 129)资源类型具有相似的数据类型DeadLetterQueue，用于处理成功调用目标 Lambda 函数后发生的故障。此类故障的示例包括 Lambda 限制或 Lambda 目标函数返回的错误。有关函数DeadLetterQueue属性的更多信息，请参阅Amazon Lambda开发者指南中的Amazon Lambda函数死信队列。

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

```
  Arn: String
  QueueLogicalId: String
  Type: String
```

属性

`Arn`

指定作为死信队列的目标的 Amazon SQS 队列的目标的 Amazon SQS 队列（ARN）。

Note

指定两者`Type`和`Arn`，但不能同时指定两者。

类型：字符串

必填项：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Events::RuleDeadLetterConfig`数据类型的`Arn`属性。

`QueueLogicalId`

如果已指定，则Amazon SAM创建的死信队列`Type`的自定义名称。

Note

如果未设置该`Type`属性，则将忽略该属性。

类型：字符串

必填项：否

Amazon CloudFormation兼容性：此属性是独一无二的Amazon SAM，没有Amazon CloudFormation等效属性。

`Type`

队列。设置此属性后，Amazon SAM会自动创建死信队列并附加必要的基于资源的策略，以授予规则资源向队列发送事件的权限。

Note

指定两者`Type`和`Arn`，但不能同时指定两者。

有效值：`SQS`

类型：字符串

必填项：否

Amazon CloudFormation兼容性：此属性是独一无二的Amazon SAM，没有Amazon CloudFormation等效属性。

示例

DeadLetterConfig

DeadLetterConfig

YAML

```
DeadLetterConfig:
  Type: SQS
  QueueLogicalId: MyDLQ
```

ScheduleV2

描述ScheduleV2事件源类型的对象，它将您的无服务器函数设置为按计划触发的 Amazon S EventBridge cheduler 事件的目标。有关更多信息，请参阅什么是 Amazon S EventBridge 计划程序？ 在《EventBridge 调度程序用户指南》中。

Amazon Serverless Application Model(Amazon SAM) 在设置此事件类型时生成AWS::Scheduler::Schedule资源。

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

```
DeadLetterConfig: DeadLetterConfig (p. 193)
Description: String
EndDate: String
FlexibleTimeWindow: FlexibleTimeWindow (p. 196)
GroupName: String
Input: String
KmsKeyArn: String
Name: String
PermissionsBoundary: String
RetryPolicy: RetryPolicy (p. 197)
RoleArn: String
ScheduleExpression: String
ScheduleExpressionTimezone: String
StartDate: String
State: String
```

属性

`DeadLetterConfig`

配置 Amazon Simple Queue Service (Amazon SQS) 队列，该队列将在目标调用失败后 EventBridge 发送事件。例如，当向不存在的 Lambda 函数发送事件时，或者没有足够的权限调用 Lambda 函数时 EventBridge ，调用可能会失败。有关更多信息，请参阅《调度程序用户指南》中的为 EventBridge 调度器配置死信队列。EventBridge

> Note
>
> 该AWS::Serverless::Function (p. 129)资源类型具有类似的数据类型DeadLetterQueue，用于处理成功调用目标 Lambda 函数后发生的故障。此类故障的示例包括 Lambda 限制或 Lambda 目标函数返回的错误。有关函数DeadLetterQueue属性的更多信息，请参阅《Amazon Lambda开发者指南》中的Amazon Lambda函数死信队列。

类型：DeadLetterConfig (p. 193)

必需：否

Amazon CloudFormation兼容性：此属性类似于AWS::Scheduler::ScheduleTarget数据类型的DeadLetterConfig属性。此属性的Amazon SAM版本包括其他子属性，以防你Amazon SAM想为自己创建死信队列。

`Description`

计划的描述。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Scheduler::Schedule资源的Description属性。

EndDate

日程可以调用其目标的日期，以 UTC 为单位。根据调度的重复表达式，调用可能会在EndDate您指定的日期或之前停止。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Scheduler::Schedule资源的EndDate属性。

FlexibleTimeWindow

允许配置可在其中调用时间表的窗口。

类型：FlexibleTimeWindow

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Scheduler::Schedule资源的FlexibleTimeWindow属性。

GroupName

与该计划关联的计划组的名称。如果未定义，则使用默认组。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Scheduler::Schedule资源的GroupName属性。

Input

传递到目标的有效 JSON 文本。如果使用此属性，则不会将事件文本本身的任何内容传递到目标。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Scheduler::Schedule Target资源的Input属性。

KmsKeyArn

将用于加密客户数据的 KMS 密钥的 ARN。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Scheduler::Schedule资源的KmsKeyArn属性。

Name

计划的名称。如果未指定名称，Amazon SAM将以以下格式生成一个名称，*Function-Logical-IDEvent-Source-Name*并使用该 ID 作为计划名称。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Scheduler::Schedule资源的Name属性。

PermissionsBoundary

用于为角色设置权限边界的策略的 ARN。

Note

如果已定义，PermissionsBoundary则Amazon SAM将对调度程序计划的目标 IAM 角色应用相同的边界。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::IAM::Role资源的PermissionsBoundary属性。

RetryPolicy

包含有关重试策略设置的信息的 RetryPolicy 对象。

类型：RetryPolicy

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Scheduler::ScheduleTarget数据类型的RetryPolicy属性。

RoleArn

计划计划被调用时 EventBridge 计划程序将用于目标的 IAM 角色的 ARN。

类型：RoleArn

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Scheduler::ScheduleTarget数据类型的RoleArn属性。

ScheduleExpression

决定运行计划计划事件的时间和频率的计划表达式。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性直接传递给AWS::Scheduler::Schedule资源的ScheduleExpression属性。

ScheduleExpressionTimezone

评估调度表达式的时区。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Scheduler::Schedule资源的ScheduleExpressionTimezone属性。

`StartDate`

以 UTC 为单位的日期，在此之后计划可以开始调用目标。根据调度的重复表达式，调用可能会在StartDate您指定的时间或之后发生。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Scheduler::Schedule`资源的`StartDate`属性。

`State`

计划程序计划的状态。

可接受的值：`DISABLED | ENABLED`

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Scheduler::Schedule`资源的`State`属性。

示例

定义 ScheduleV2 资源的基本示例

```
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

描述`SelfManagedKafka`事件源类型的对象。有关更多信息，请参阅《Amazon Lambda开发者指南》中的“Amazon Lambda与自管理的 Apache Kafka 一起使用”。

Amazon Serverless Application Model(Amazon SAM) 在设置此事件类型时生成AWS::Lambda::EventSourceMapping资源。

语法

要在Amazon SAM模板中声明此实体，请使用以下语法。

YAML

```
  BatchSize: Integer
  ConsumerGroupId: String
  Enabled: Boolean
  FilterCriteria: FilterCriteria
  KafkaBootstrapServers: List
  SourceAccessConfigurations: SourceAccessConfigurations
  Topics: List
```

属性

`BatchSize`

Lambda 从流中提取并发送到函数的每个批处理中的最大记录数。

类型：整数

必需：否

默认值：100

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Lambda::EventSourceMapping`资源的`BatchSize`属性。

最小值：1

最大值：`10000`

`ConsumerGroupId`

一个字符串，用于配置如何从 Kafka 主题读取事件。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Lambda::EventSourceMapping`资源的`SelfManagedKafkaConfiguration`属性。

`Enabled`

禁用事件源映射以暂停轮询和调用。

类型：布尔值

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Lambda::EventSourceMapping`资源的`Enabled`属性。

`FilterCriteria`

定义用于确定 Lambda 是否应处理事件的标准的对象。有关更多信息，请参阅Amazon Lambda开发者指南中的Amazon Lambda事件筛选。

类型：FilterCriteria

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的FilterCriteria属性。

KafkaBootstrapServers

适用于以下列表 Kafka 代理的 Bootstrap 服务器列表。例如，包括端口broker.example.com:*xxxx*

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

SourceAccessConfigurations

用于保护与定义事件源的身份验证协议数组 VPC 组件或虚拟化主机。

有效值：BASIC_AUTH | CLIENT_CERTIFICATE_TLS_AUTH | SASL_SCRAM_256_AUTH | SASL_SCRAM_512_AUTH | SERVER_ROOT_CA_CERTIFICATE

类型：SourceAccessConfiguration 列表

必需：是

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的SourceAccessConfigurations属性。

Topics

Kafka 主题的名称。

类型：清单

必需：是

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的Topics属性。

示例

自行管理的 Kafka 事件源

以下是SelfManagedKafka事件源类型的示例。

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

描述SNS事件源类型的对象。

设置此事件类型后，SAM 会生成AWS::SNS::Subscription资源

语法

要在请使用以下语法：请使用语法：请Amazon Serverless Application Model使用以下Amazon SAM语法：请使用以下语法：请使用以下语法：

YAML

```
FilterPolicy: SnsFilterPolicy
RedrivePolicy (p. 201): Json
Region: String
SqsSubscription: Boolean | SqsSubscriptionObject (p. 202)
Topic: String
```

属性

`FilterPolicy`

分配给订阅的筛选策略 JSON。有关更多信息，请参阅 Amazon 简单通知服务 API 参考GetSubscriptionAttributes中的 Amazon Simple Notion

类型：SnsFilterPolicy

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::SNS::Subscription`资源的`FilterPolicy`属性。

`RedrivePolicy`

指定后，将无法传输的消息发送到指定的 Amazon SQS 死信队列。由于客户端错误（例如，在无法访问订阅的端点时）或服务器错误（例如，在支持订阅的端点的服务变得不可用时）而无法传输的消息将保留在死信队列中，以进行进一步分析或重新处理。

有关重新驱动策略和死信队列的更多信息，请参阅 Amazon Simple Queue Service 开发人员指南中的Amazon SQS 死信队列。

类型：Json

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::SNS::Subscription`资源的`RedrivePolicy`属性。

`Region`

对于跨区域订阅，为主题所在的区域。

如果没有指定 CloudFormation

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::SNS::Subscription`资源的`Region`属性。

`SqsSubscription`

将此属性设置为 true，或指定在`SqsSubscriptionObject` SQS 队列中启用批处理 SNS 主题通知。将此属性设置为`true`会创建新的 SQS 队列，而指定 a 则`SqsSubscriptionObject`使用现有的 SQS 队列。

类型：布尔值 | SqsSubscriptionObject (p. 202)

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`Topic`

要订阅的主题的 ARN。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性直接传递给`AWS::SNS::Subscription`资源的`TopicArn`属性。

示例

SNS 事件源源示例示例：

SNS 事件源源示例示例：

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

为 SNS 事件指定现有 SQS 队列选项

语法

要在您的中声明此实体Amazon Serverless Application Model(Amazon SAM) 模板，请使用以下语法。

YAML

```
  BatchSize: String
  Enabled: Boolean
  QueueArn: String
  QueuePolicyLogicalId: String
```

```
  QueueUrl: String
```

属性

`BatchSize`

要在单个批次中检索的最大项目数，以供 SQS 队列检索的最大项目数。

类型：字符串

必需：否

默认值：10

Amazon CloudFormation兼容性：该属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`Enabled`

禁用 SQS 事件源映射以暂停轮询和调用。

类型：Boolean

必需：否

默认值：True

Amazon CloudFormation兼容性：该属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`QueueArn`

指定现有的 SQS 队列 ARN。

类型：字符串

必需：是

Amazon CloudFormation兼容性：该属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`QueuePolicyLogicalId`

为AWS::SQS::QueuePolicy资源。

类型：字符串

必需：否

Amazon CloudFormation兼容性：该属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`QueueUrl`

指定与`QueueArn`财产。

类型：字符串

必需：是

Amazon CloudFormation兼容性：该属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

示例

针对 SNS 的现有 SQS 事件

添加现有 SQS 队列以订阅 SNS 主题的示例。

YAML

```
QueuePolicyLogicalId: CustomQueuePolicyLogicalId
QueueArn:
  Fn::GetAtt: MyCustomQueue.Arn
QueueUrl:
  Ref: MyCustomQueue
BatchSize: 5
```

SQS

描述SQS事件源类型的对象。有关更多信息，请参阅 Amazon Lambda 开发人员指南中的将 Amazon Lambda 与 Amazon SQS 结合使用。

设置此事件类型后，SAM 会生成AWS::Lambda::EventSourceMapping资源

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

```
  BatchSize: Integer
  Enabled: Boolean
  FilterCriteria: FilterCriteria
  FunctionResponseTypes: List
  MaximumBatchingWindowInSeconds: Integer
  Queue: String
  ScalingConfig (p. 205): ScalingConfig
```

属性

`BatchSize`

要在单个批次中检索的最大项目数。

类型：整数

必需：否

默认值：10

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Lambda::EventSourceMapping`资源的`BatchSize`属性。

最小值：1

最大值：`10000`

`Enabled`

禁用事件源映射以暂停轮询和调用。

类型：布尔值

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的Enabled属性。

FilterCriteria

定义用于确定 Lambda 是否应处理事件的条件的对象。有关更多信息，请参阅《Amazon Lambda开发者指南》中的Amazon Lambda事件筛选。

类型：FilterCriteria

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的FilterCriteria属性。

FunctionResponseTypes

当前应用于事件源映射的响应类型列表。有关更多信息，请参阅Amazon Lambda开发者指南中的报告批处理项目故障。

有效值：ReportBatchItemFailures

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的FunctionResponseTypes属性。

MaximumBatchingWindowInSeconds

在调用函数之前收集记录的最长时间（以秒为单位）。

类型：整数

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的MaximumBatchingWindowInSeconds属性。

Queue

队列的 ARN。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的EventSourceArn属性。

ScalingConfig

扩展 SQS 轮询器的配置以控制调用速率和设置最大并发调用次数。

类型： ScalingConfig

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::EventSourceMapping资源的ScalingConfig属性。

示例

SQS 活动

SQS 活动

YAML

```
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"]}'
```

带有 SQS 事件且已配置扩展的 Lambda 函数

```
MyFunction:
  Type: AWS::Serverless::Function
  Properties:
    ...
    Events:
      MySQSEvent:
        Type: SQS
          Properties:
            ...
            ScalingConfig:
              MaximumConcurrency: 10
```

FunctionCode

Lambda 函数的部署程序包。

语法

要在您的Amazon Serverless Application Model(Amazon SAM) 模板，请使用以下语法。

YAML

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

属性

`Bucket`

与您的函数处于同一Amazon区域的 Amazon S3 存储桶。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给`S3Bucket`的财产`AWS::Lambda::Function` `Code`数据类型。

`Key`

部署程序包的 Amazon S3 密钥。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给`S3Key`的财产`AWS::Lambda::Function Code`数据类型。

`Version`

对于版本控制的对象，指要使用的部署程序包对象的版本。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`S3ObjectVersion`的财产`AWS::Lambda::Function Code`数据类型。

示例

FunctionCode

示例函数代码

YAML

```
FunctionCode:
  Bucket: mybucket-name
  Key: mykey-name
  Version: 121212
```

FunctionUrlConfig

使用指定的配置参数创建函数 URL。函数 URL 是可用于调用函数的 HTTPS 端点。

默认情况下，函数 URL 使用`$LATEST`您的 Lambda 函数的版本。如果你指定`AutoPublishAlias`对于您的 Lambda 函数，终端节点将连接到指定的函数别名。

有关更多信息，请参阅 。函数 URL中的Amazon Lambda开发人员指南.

语法

要在您的Amazon Serverless Application Model(Amazon SAM) 模板，请使用以下语法。

YAML

```
  AuthType: String
  Cors: Cors
```

属性

`AuthType`

函数 URL 的授权类型。设置为`AWS_IAM`以使用 IAM 授权请求。设置为`NONE`为开放访问提供。

有关更多信息，请参阅 。函数 URL中的Amazon Lambda开发人员指南

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给AuthType的财产AWS::Lambda::Url资源。

Cors

函数 URL 的跨源资源共享 (CORS) 设置。

类型：Cors

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给Cors的财产AWS::Lambda::Url资源。

示例

函数 URL

以下示例使用函数 URL 创建 Lambda 函数。函数 URL 使用 IAM 授权。

YAML

```
HelloWorldFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: hello_world/
    Handler: index.handler
    Runtime: nodejs14.x
    FunctionUrlConfig:
      AuthType: AWS_IAM

Outputs:
  MyFunctionUrlEndpoint:
      Description: "My Lambda Function URL Endpoint"
      Value:
        Fn::GetAtt: HelloWorldFunctionUrl.FunctionUrl
```

AWS::Serverless::HttpApi

创建一个Amazon API Gateway HTTP API，这使您能够创建比 REST API 具有更低延迟和更低成本的 RESTful API 的 API，有关更多信息，请参阅 API Gateway 开发人员指南中的使用 HTTP API 开发人员指南中的使用 HTTP API

我们建议您使用Amazon CloudFormation挂钩或 IAM 策略来验证 API Gateway 资源是否附加了授权者以控制对它们的访问权限。

有关使用Amazon CloudFormation挂钩的更多信息，请参阅 Amazon CloudFormationCLI 用户指南和apigw-enforce-authorizer GitHub 存储库中的注册挂钩。

有关使用 IAM 策略的更多信息，请参阅 API Gateway 开发人员指南中的要求 API 路由获得授权。

Note

当您部署到时Amazon CloudFormation，Amazon SAM会将您的Amazon SAM资源转换为Amazon CloudFormation资源。有关更多信息，请参阅Amazon CloudFormation (p. 261)：

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

```
Type: AWS::Serverless::HttpApi
Properties:
  AccessLogSettings: AccessLogSettings
  Auth: HttpApiAuth (p. 215)
  CorsConfiguration: String | HttpApiCorsConfiguration (p. 222)
  DefaultRouteSettings: RouteSettings
  DefinitionBody: JSON
  DefinitionUri: String | HttpApiDefinition (p. 223)
  Description: String
  DisableExecuteApiEndpoint: Boolean
  Domain: HttpApiDomainConfiguration (p. 224)
  EnableFunctionDefaultPermissions (p. 211): Boolean
  FailOnWarnings: Boolean
  Name (p. 211): String
  RouteSettings: RouteSettings
  StageName: String
  StageVariables: Json
  Tags: Map
```

属性

`AccessLogSettings`

阶段的访问日志记录的设置。

类型：AccessLogSettings

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::ApiGatewayV2::Stage`资源的`AccessLogSettings`属性。

`Auth`

配置授权以控制对您的 API Gateway HTTP API 的访问。

有关更多信息，请参阅《API Gateway 开发人员指南》中的使用 JWT 授权方控制对 HTTP API 的访问。

类型：HttpApiAuth (p. 215)

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`CorsConfiguration`

管理所有 API Gateway HTTP API API 的跨域资源共享 (CORS)。以字符串形式指定要允许的域，或指定一个`HttpApiCorsConfiguration`对象。请注意，CORSAmazon SAM 需要修改你的 OpenAPI 定义，因此 CORS 只有在指定了`DefinitionBody`属性时才起作用。

有关更多信息，请参阅 API Gateway 开发人员指南中的为 HTTP API 配置 CORS 的 CORS 配置 CORS。

Note

如果CorsConfiguration在 OpenAPI 定义和属性级别都设置了，则将两个配置源Amazon SAM合并，属性优先。如果将此属性设置为true，则允许所有来源。

类型：字符串 | HttpApiCorsConfiguration (p. 222)

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`DefaultRouteSettings`

此 HTTP API 的默认路由设置。这些设置适用于所有路由，除非被某些路径的RouteSettings属性所覆盖。

类型：RouteSettings

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::ApiGatewayV2::Stage资源的RouteSettings属性。

`DefinitionBody`

描述你的 HTTP API 的 OpenAPI 定义。如果您未指定 aDefinitionUri 或 aDefinitionBody，则根据您的模板配置DefinitionBody为您Amazon SAM生成。

类型：JSON

必需：否

Amazon CloudFormation兼容性：此属性类似于AWS::ApiGatewayV2::Api资源的Body属性。如果提供了某些属性，则Amazon SAM可以在将内容传递给DefinitionBody之前将其插入或修改Amazon CloudFormation。属性包括相应AWS::Serverless::Function资源EventSource HttpApi的Auth和类型。

`DefinitionUri`

定义 HTTP API API 的 Amazon Simple Storage Service (Amazon S3) URI、本地文件路径或位置对象。此属性引用的Amazon S3 对象必须是有效的 OpenAPI 定义文件。如果您未指定 aDefinitionUri 或 a，DefinitionBody则根据您的模板配置DefinitionBody为您Amazon SAM生成。

如果您提供本地文件路径，则模板必须经过包含sam deploy或sam package命令的工作流程才能正确转换定义。

您引用的外部 OpenApi 定义文件不支持内部函数DefinitionUri。要将 OpenApi 定义导入到模板中，请将该DefinitionBody属性与 Include 转换一起使用。

类型：字符串 | HttpApiDefinition (p. 223)

必需：否

Amazon CloudFormation兼容性：此属性类似于AWS::ApiGatewayV2::Api资源的BodyS3Location属性。嵌套的 Amazon S3 属性的命名方式不同。

`Description`

HTTP API 资源的描述。

指定时Description，Amazon SAM将通过设置description字段来修改 HTTP API 资源的 OpenApi 定义。以下情况将会导致错误：

- 该DefinitionBody属性是使用 Open API 定义中设置的description字段指定的 — 这会导致description字段冲突，但Amazon SAM无法解决。
- DefinitionUri属性已指定 —Amazon SAM 不会修改从 Amazon S3 检索到的 Open API 定义。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`DisableExecuteApiEndpoint`

指定客户端是否可以使用默认execute-api终端节点调用您的 HTTP API API 调用 HTTP API APIhttps://{api_id}.execute-api.{region}.amazonaws.com。默认情况下，客户端可以使用默认 终端节点调用您的 API。如果要求客户端仅使用自定义域名调用您的 API，请禁用默认终端节点。

类型：布尔值

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::ApiGatewayV2::Api资源的DisableExecuteApiEndpoint属性。

`Domain`

为此 API Gateway HTTP API 配置自定义域。

类型：HttpApiDomainConfiguration (p. 224)

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`EnableFunctionDefaultPermissions`

默认情况下，不授予 HTTP API 资源调用 Lambda 授权器的权限。将此属性指定true为在您的 HTTP API 资源和 Lambda 授权者之间自动创建权限。

类型：布尔值

必需：否

默认值：false

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`FailOnWarnings`

指定遇到警告时回滚 HTTP API 创建 (true) 或不回滚 HTTP API 创建 (false) 的 HTTP API 创建过程 ()。默认值为 false。

类型：布尔值

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::ApiGatewayV2::Api资源的FailOnWarnings属性。

`Name`

HTTP API 资源的名称。

当你指定时Name，Amazon SAM将通过设置title字段来修改 HTTP API 资源的 OpenAPI 定义。以下情况将会导致错误：

- 该DefinitionBody属性是使用 Open API 定义中设置的title字段指定的 — 这会导致title字段冲突，但Amazon SAM无法解决。
- DefinitionUri属性已指定 —Amazon SAM 不会修改从 Amazon S3 检索到的 Open API 定义。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`RouteSettings`

此 HTTP API 的每条路由的路由设置。有关更多信息，请参阅 API Gat e way 开发人员指南中的使用 HTTP API 路由。

类型：RouteSettings

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::ApiGatewayV2::Stage资源的RouteSettings属性。

`StageName`

API 阶段的名称。如果未指定名称，则Amazon SAM使用 API Gateway 中的$default阶段。

类型：字符串

必需：否

默认值：$默认

Amazon CloudFormation兼容性：此属性直接传递给AWS::ApiGatewayV2::Stage资源的StageName属性。

`StageVariables`

一个定义阶段变量的映射。变量名称可以包含字母数字和下划线字符。值必须匹配 [A-Za-z0-9-9-._~:/？#&=，] +。

类型：Json

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::ApiGatewayV2::Stage资源的StageVariables属性。

`Tags`

一个映射（字符串到字符串），指定要添加到此 API Gateway 阶段的标签。密钥的长度可以为 1 到 128 个 Unicode 字符并且不能包含前缀aws:。您可以使用以下任一字符：Unicode 字母、数字、空格、_、.、/、=、+ 和 - 的组合。值的长度可以为 1 到 256 个 Unicode 字符之间。

类型：地图

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

附加说明：该Tags属性Amazon SAM需要修改您的 OpenAPI 定义，因此只有在指定了DefinitionBody属性时才添加标签，如果指定了该DefinitionUri属性，则不会添加任何标签。 Amazon SAM自动添加httpapi:createdBy:SAM标签。标签还会添加到AWS::ApiGatewayV2::Stage资源和资源（如果DomainName已指定）。AWS::ApiGatewayV2::DomainName

返回值

Ref

在将此资源的逻辑 ID 传递给内部Ref函数时，会Ref返回底层AWS::ApiGatewayV2::Api资源的 API ID，例如a1bcdef2gh。

有关使用该Ref函数的更多信息，请参阅Ref《Amazon CloudFormation用户指南》。

示例

简单 HttpApi

以下示例显示了设置由 Lambda 函数支持的 HTTP API 终端节点所需的最低限度。此示例使用Amazon SAM创建的默认 HTTP API。

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Description: Amazon 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 使用身份验证

以下示例说明如何在 HTTP API 端点上设置授权。

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 使用 OpenAPI 定义

以下示例说明如何将 OpenAPI 定义（OpenAPI）定义添加到模板。

请注意，对于引用此 HTTP API HttpApi 的事件，请Amazon SAM填写所有缺失的 Lambda 集成。 Amazon SAM还添加了 HttpApi 事件引用的任何缺失路径。

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 使用配置设置

以下示例说明如何将 HTTP API 和阶段配置添加到模板。

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

配置授权来控制对 Amazon API Gateway HTTP API 的访问。

有关配置 HTTP API 访问权限的更多信息，请参阅控制和管理对 API Gateway 中 HTTP API 的访问中的API Gateway 开发人员指南.

语法

要在您的Amazon Serverless Application Model(Amazon SAM) 模板，请使用以下语法。

YAML

```
  Authorizers: OAuth2Authorizer (p. 220) | LambdaAuthorizer (p. 217)
  DefaultAuthorizer: String
```

```
  EnableIamAuthorizer: Boolean
```

属性

`Authorizers`

用于控制对 API Gateway API 访问权限的授权方。

类型：OAth2授权者/授权器 (p. 220)|LambdaAuthorizer (p. 217)

必需：否

默认值：无

Amazon CloudFormation兼容性：该属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

附加说明：Amazon SAM将授权者添加到 OpenAPI 定义中。

`DefaultAuthorizer`

指定用于授权 API 调用 API Gateway 的默认授权方。您可以指定`AWS_IAM`作为默认授权者`EnableIamAuthorizer`设置为`true`. 否则，请指定您在中定义的授权者`Authorizers`.

类型：字符串

必需：否

默认值：无

Amazon CloudFormation兼容性：该属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`EnableIamAuthorizer`

指定是否对 API 路由使用 IAM 授权。

类型：布尔值

必需：否

Amazon CloudFormation兼容性：该属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

示例

OAuth 2.0 授权方

OAuth 2.0 授权方示例

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 授权方

IAM 授权方示例

YAML

```
Auth:
  EnableIamAuthorizer: true
  DefaultAuthorizer: AWS_IAM
```

LambdaAuthorizer

使用以下方配置 Lambda 授权方以控制对 Amazon API Gateway HTTP API 的访问。Amazon Lambdafunction.

有关详细信息和示例，请参阅使用Amazon LambdaHTTP API 的授权方中的API Gateway 开发人员指南.

语法

要在您的版本中申报此实体Amazon Serverless Application Model(Amazon SAM) 模板，请使用以下语法。

YAML

```
  AuthorizerPayloadFormatVersion: String
  EnableSimpleResponses: Boolean
  FunctionArn: String
  FunctionInvokeRole: String
  Identity: LambdaAuthorizationIdentity (p. 219)
```

属性

`AuthorizerPayloadFormatVersion`

指定发送到 HTTP API Lambda 授权方的负载的格式。对于 HTTP API Lambda 授权方必须指定。

这将传递至`authorizerPayloadFormatVersion`的部分`x-amazon-apigateway-authorizer`中的`securitySchemes`OpenAPI 定义的部分。

有效值：`1.0` 或 `2.0`

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`EnableSimpleResponses`

指定 Lambda 授权方是否以简单格式返回响应。默认情况下，Lambda 授权方必须返回Amazon Identity and Access Management(IAM) 策略。如果启用，Lambda 授权方可以返回布尔值，而不是 IAM 策略。

这将传递至`enableSimpleResponses`的部分`x-amazon-apigateway-authorizer`中的`securitySchemes`OpenAPI 定义的部分。

类型：Boolean

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`FunctionArn`

提供 API 授权的 Lambda 函数的 Amazon 资源名称 (ARN)。

这将传递至`authorizerUri`的部分`x-amazon-apigateway-authorizer`中的`securitySchemes`OpenAPI 定义的部分。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`FunctionInvokeRole`

具有 API Gateway 所需凭证的 IAM 角色的 ARN 以调用授权方函数。如果您的函数的基于资源的策略未授权 API Gateway，请指定此参数`lambda:InvokeFunction`权限。

这将传递至`authorizerCredentials`的部分`x-amazon-apigateway-authorizer`中的`securitySchemes`OpenAPI 定义的部分。

有关更多信息，请参阅 。创建 Lambda 授权方中的API Gateway 开发人员指南.

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`Identity`

指定`IdentitySource`在收到的授权人请求中。

这将传递至`identitySource`的部分`x-amazon-apigateway-authorizer`中的`securitySchemes`OpenAPI 定义的部分。

类型：lambda 授权身份 (p. 219)

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

示例

lambdaA授权器

Lambda 授权方示例

YAML

```
Auth:
  Authorizers:
```

```
    MyLambdaAuthorizer:
      AuthorizerPayloadFormatVersion: 2.0
      FunctionArn:
        Fn::GetAtt:
          - MyAuthFunction
          - Arn
      FunctionInvokeRole:
        Fn::GetAtt:
          - LambdaAuthInvokeRole
          - Arn
      Identity:
        Headers:
          - Authorization
```

LambdaAuthorizationIdentity

使用属性可用于在对 Lambda 授权者的传入请求中指定 IdentitySource。有关身份源的更多信息，请参阅身份来源中的API Gateway 开发人员指南.

语法

要在您的中声明此实体Amazon Serverless Application Model(Amazon SAM) 模板，请使用以下语法。

YAML

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

属性

`Context`

将给定的上下文字符串转换为格式的映射表达式列表`$context.contextString`.

类型：List

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`Headers`

将标题转换为格式的映射表达式列表`$request.header.name`.

类型：List

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`QueryStrings`

将给定的查询字符串转换为格式的映射表达式列表`$request.querystring.queryString`.

类型：List

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`ReauthorizeEvery`

生存时间 (TTL) 期间（以秒为单位），用于指定 API Gateway 缓存授权方结果的时长。如果您指定一个大于 0 的值，API Gateway 将缓存授权方响应。最大值为 3600 秒（1 小时）。

类型：整数

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`StageVariables`

将给定阶段变量转换为格式的映射表达式列表`$stageVariables.stageVariable`.

类型：List

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

示例

lambda 请求身份

Lambda 请求身份示例

YAML

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

OAuth2Authorizer

OAuth 2.0 授权者的定义，也称为 JSON Web Token（JWT）令牌。

有关更多信息，请参阅《API Gateway 开发人员指南》中的使用 JWT 授权方控制对 HTTP API 的访问。

语法

要在Amazon Serverless Application Model（Amazon SAM）模板中声明此实体，请使用以下语法。

YAML

```
  AuthorizationScopes: List
  IdentitySource: String
  JwtConfiguration: Map
```

属性

`AuthorizationScopes`

此授权者的授权范围列表。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`IdentitySource`

此授权者的身份源表达式。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`JwtConfiguration`

此授权者的 JWT 配置。

这将传递到`jwtConfiguration` OpenAPI 定义`securitySchemes`部分`x-amazon-apigateway-authorizer`中的 a 部分。

> **Note**
>
> 属`issuer`性和`audience`不区分大小写，可以像在 OpenAPI 中一样使用小写字母，也可以`Audience`像中一样使用大写字母`Issuer`AWS::ApiGatewayV2::Authorizer。

类型：地图

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

示例

OAuth 2.0 授权者

OAuth 2.0 授权器示例

YAML

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

```
  DefaultAuthorizer: OAuth2Authorizer
```

HttpApiCorsConfiguration

为 HTTP API 管理跨源资源共享 (CORS)。将允许的域指定为字符串，或者指定带有其他 Cors 配置的字典。注意：Cora 要求 SAM 修改你的 OpenAPI 定义，因此它只适用于`DefinitionBody`财产。

有关 CORS 的更多信息，请参阅为 HTTP API 配置 CORS中的API Gateway 开发人员指南.

注意：如果在 OpenAPI 和属性级别都设置了 HTTPiCORS 配置，Amazon SAM将它们与优先属性合并。

语法

要在您的中声明此实体Amazon Serverless Application Model(Amazon SAM) 模板，请使用以下语法。

YAML

```
  AllowCredentials: Boolean
  AllowHeaders: List
  AllowMethods: List
  AllowOrigins: List
  ExposeHeaders: List
  MaxAge: Integer
```

属性

`AllowCredentials`

指定是否在 CORS 请求中包含凭证。

类型：Boolean

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`AllowHeaders`

表示允许的标头集合。

类型：List

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`AllowMethods`

表示允许的 HTTP 方法集合。

类型：List

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`AllowOrigins`

表示允许的源集合。

类型：List

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`ExposeHeaders`

表示公开的标头集合。

类型：List

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`MaxAge`

浏览器应缓存预检请求结果的秒数。

类型：整数

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

示例

httpPicors配置

HTTP API CORS 配置示例。

YAML

```
CorsConfiguration:
  AllowOrigins:
    - "https://example.com"
  AllowHeaders:
    - x-apigateway-header
  AllowMethods:
    - GET
  MaxAge: 600
  AllowCredentials: true
```

HttpApiDefinition

定义 API 的 OpenAPI 文档。

语法

要在您的Amazon Serverless Application Model(Amazon SAM) 模板，请使用以下语法。

YAML

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

属性

`Bucket`

存储 OpenAPI 文件的 Amazon S3 存储桶的名称。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给`Bucket`的财产`AWS::ApiGatewayV2::Api BodyS3Location`数据类型。

`Key`

OpenAPI 文件的 Amazon S3 密钥。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给`Key`的财产`AWS::ApiGatewayV2::Api BodyS3Location`数据类型。

`Version`

对于版本控制的对象，则为 OpenAPI 文件的版本。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`Version`的财产`AWS::ApiGatewayV2::Api BodyS3Location`数据类型。

示例

示例定义示例

示例 API 定义

YAML

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

HttpApiDomainConfiguration

为 API 配置自定义域。

语法

要在您的中声明此实体Amazon Serverless Application Model(Amazon SAM) 模板，请使用以下语法。

YAML

```
  BasePath: List
```

```
  CertificateArn: String
  DomainName: String
  EndpointConfiguration: String
  MutualTlsAuthentication: MutualTlsAuthentication
  OwnershipVerificationCertificateArn: String
  Route53: Route53Configuration (p. 227)
  SecurityPolicy: String
```

属性

`BasePath`

要使用 Amazon API Gateway 域名配置的基本路径列表。

类型：列表

必需：否

默认值:/

Amazon CloudFormation兼容性：此属性类似于`ApiMappingKey`的财产`AWS::ApiGatewayV2::ApiMapping`资源。Amazon SAM创建多个`AWS::ApiGatewayV2::ApiMapping`资源，此属性中指定的每个值一个。

`CertificateArn`

的 Amazon 资源名称 (ARN)Amazon此域名终端节点的托管证书。Amazon Certificate Manager是唯一受支持的源。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给`CertificateArn`的财产`AWS::ApiGateway2::DomainName DomainNameConfiguration`资源。

`DomainName`

API Gateway API 的自定义域名。不支持大写字母。

Amazon SAM生成`AWS::ApiGatewayV2::DomainName`设置此属性时的资源。有关此方案的信息，请参阅指定了 DomainName 属性 (p. 269). 有关生成的信息Amazon CloudFormation资源，请参阅Amazon CloudFormation (p. 261).

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给`DomainName`的财产`AWS::ApiGateway2::DomainName`资源。

`EndpointConfiguration`

定义要映射到自定义域的 API Gateway 终端节点的类型。此属性的价值决定了`CertificateArn`属性被映射在Amazon CloudFormation.

HTTP API 的唯一有效值为`REGIONAL`.

类型：字符串

必需：否

默认值：`REGIONAL`

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`MutualTlsAuthentication`

自定义域名的相互传输层安全性 (TLS) 身份验证配置。

类型：MutualTls身份验证

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`MutualTlsAuthentication`的财产`AWS::ApiGatewayV2::DomainName`资源。

`OwnershipVerificationCertificateArn`

ACM 颁发的用于验证自定义域所有权的公有证书的 ARN。仅当您配置双向 TLS 并指定导入的 ACM 或私有 CA 证书 ARN 时才需要`CertificateArn`.

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`OwnershipVerificationCertificateArn`的财产`AWS::ApiGatewayV2::DomainName DomainNameConfiguration`数据类型。

`Route53`

定义亚马逊路线 53 配置。

类型：Route53 配置 (p. 227)

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`SecurityPolicy`

此域名的安全策略的 TLS 版本。

HTTP API 的唯一有效值为`TLS_1_2`.

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`SecurityPolicy`的财产`AWS::ApiGatewayV2::DomainName DomainNameConfiguration`数据类型。

示例

DomainName

DomainName示例

YAML

```
Domain:
  DomainName: www.example.com
  CertificateArn: arn-example
  EndpointConfiguration: REGIONAL
  Route53:
```

```
    HostedZoneId: Z1PA6795UKMFR9
  BasePath:
    - foo
    - bar
```

Route53Configuration

为 API 配置 Route53 记录集。

语法

要在Amazon Serverless Application Model（Amazon SAM）中声明此实体，请使用以下语法。

YAML

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

属性

`DistributionDomainName`

配置 API 自定义域名的自定义分配。

类型：字符串

必需：否

默认：使用 API Gateway 发行版。

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Route53::RecordSetGroup` `AliasTarget`资源的`DNSName`属性。

附加说明：CloudFront 发行版的域名。

`EvaluateTargetHealth`

当 EvaluateTargetHealth 为时，别名记录继承所引用Amazon资源（例如托管区域中的 ElastLlong 负载均衡器或其他记录）的运行状况。

类型：布尔值

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Route53::RecordSetGroup` `AliasTarget`资源的`EvaluateTargetHealth`属性。

附加说明：当别名目标是，您不能将设置 EvaluateTargetHealth CloudFront 为。

`HostedZoneId`

要在其中创建记录的托管区的 ID。

指定 `HostedZoneName` 或 `HostedZoneId`，但不能同时指定两者。如果您拥有多个使用相同域名的托管区域，则必须使用 `HostedZoneId` 指定托管区。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Route53::RecordSetGroup RecordSet资源的HostedZoneId属性。

HostedZoneName

要在其中创建记录的托管区的名称。您必须包含结尾圆点（例如 www.example.com.）作为 HostedZoneName 的一部分。

指定 HostedZoneName 或 HostedZoneId，但不能同时指定两者。如果您拥有多个使用相同域名的托管区域，则必须使用 HostedZoneId 指定托管区。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Route53::RecordSetGroup RecordSet资源的HostedZoneName属性。

IpV6

设置此属性后，将Amazon SAM创建AWS::Route53::RecordSet资源并将所提供资源的 Type 设置AAAA为 HostedZone。

类型：布尔值

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

示例

Route 53

此示例演示示示示示示示示示示示示示示示示示

YAML

```
Domain:
  DomainName: www.example.com
  CertificateArn: arn-example
  EndpointConfiguration: EDGE
  Route53:
    HostedZoneId: Z1PA6795UKMFR9
    EvaluateTargetHealth: true
    DistributionDomainName: xyz
```

AWS::Serverless::LayerVersion

创建包含 Lamb LayerVersion da 函数所需的库或运行时代码的 Lambda。

该AWS::Serverless::LayerVersion (p. 228)资源还支持Metadata资源属性，因此您可以指示Amazon SAM构建应用程序中包含的图层。有关建筑层的更多信息，请参阅建筑层 (p. 354)。

重要说明：自从发布UpdateReplacePolicy资源属性以来Amazon CloudFormation，AWS::Lambda::LayerVersion（推荐）提供的好处与AWS::Serverless::LayerVersion (p. 228)。

转换无服务器 LayerVersion 时，SAM 还会转换资源的逻辑 ID，这样 LayerVersions 在资源更新 CloudFormation 时不会自动删除旧的 ID。

Note

当您部署到时Amazon CloudFormation，Amazon SAM会将您的Amazon SAM资源转换为Amazon CloudFormation资源。有关更多信息，请参阅Amazon CloudFormation (p. 261)：

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

```
Type: AWS::Serverless::LayerVersion
Properties:
  CompatibleArchitectures: List
  CompatibleRuntimes: List
  ContentUri: String | LayerContent (p. 231)
  Description: String
  LayerName: String
  LicenseInfo: String
  RetentionPolicy: String
```

属性

`CompatibleArchitectures`

指定层版本支持的指令集架构。

有关此属性的更多信息，请参阅Amazon Lambda开发者指南中的 Lambda 指令集架构。

有效值：`x86_64`、`arm64`

类型：清单

必需：否

默认值：`x86_64`

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Lambda::LayerVersion`资源的`CompatibleArchitectures`属性。

`CompatibleRuntimes`

与此兼容的运行时列表 LayerVersion。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Lambda::LayerVersion`资源的`CompatibleRuntimes`属性。

`ContentUri`

Amazon S3 URI、本地文件夹的路径或层代码的 LayerContent 对象。

如果提供了 Amazon S3 URI 或 LayerContent 对象，则引用的Amazon S3 对象必须是包含 Lambda 层内容的有效 ZIP 存档。

如果提供了本地文件夹的路径，则要正确转换内容，模板必须经过包括sam deploy (p. 412)或山姆·布莱德 (p. 408)在内的工作流程sam package (p. 431)。默认情况下，相对路径是相对于Amazon SAM模板的位置进行解析的。

类型：字符串 | LayerContent (p. 231)

必需：是

Amazon CloudFormation兼容性：此属性类似于AWS::Lambda::LayerVersion资源的Content属性。嵌套的 Amazon S3 属性的命名方式不同。

`Description`

此层的描述。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::LayerVersion资源的Description属性。

`LayerName`

层的名称或 Amazon 资源名称（ARN）。

类型：字符串

必需：否

默认：资源逻辑 ID

Amazon CloudFormation兼容性：此属性类似于AWS::Lambda::LayerVersion资源的LayerName属性。如果您没有指定名称，则将使用资源的逻辑 ID 来作为名称。

`LicenseInfo`

有关此许可证的信息 LayerVersion。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Lambda::LayerVersion资源的LicenseInfo属性。

`RetentionPolicy`

指定更新后 LayerVersion 是保留还是删除您的旧版本。

有效值：`Retain` 或 `Delete`

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

附加说明：指定时Retain，Amazon SAM将DeletionPolicy: Retain向转换后的AWS::Lambda::LayerVersion资源添加一个资源属性 (p. 260)。

返回值

Ref

当向Ref内部函数提供此资源的逻辑 ID 时，它将返回底层 Lambda 的资源ARN LayerVersion。

有关使用该Ref函数的更多信息，请参阅Ref《Amazon CloudFormation用户指南》。

示例

LayerVersionExample

的示例 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

一个 ZIP 归档，其中包含Lambda 层.

语法

要在您的中声明此实体Amazon Serverless Application Model(Amazon SAM) 模板，请使用以下语法。

YAML

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

属性

`Bucket`

层存档的 Amazon S3 存储桶。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给S3Bucket的财产AWS::Lambda::LayerVersion Content数据类型。

`Key`

层存档的 Amazon S3 密钥。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给S3Key的财产AWS::Lambda::LayerVersion Content数据类型。

`Version`

对于进行版本控制的对象，则为要使用的层存档对象版本。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给S3ObjectVersion的财产AWS::Lambda::LayerVersion Content数据类型。

示例

图层内容

层内容示例

YAML

```
LayerContent:
  Bucket: mybucket-name
  Key: mykey-name
  Version: 121212
```

AWS::Serverless::SimpleTable

使用单一属性主键创建 DynamoDB 表。当只需要通过主键访问数据时，它很有用。

要使用 DynamoDB 的更高级功能，请改用AWS::DynamoDB::Table资源。

> **Note**
>
> 当您部署到时Amazon CloudFormation，Amazon SAM会将您的Amazon SAM资源转换为Amazon CloudFormation资源。有关更多信息，请参阅Amazon CloudFormation (p. 261)：

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

```
Type: AWS::Serverless::SimpleTable
Properties:
  PrimaryKey: PrimaryKeyObject (p. 234)
  ProvisionedThroughput: ProvisionedThroughput
  SSESpecification: SSESpecification
  TableName: String
  Tags: Map
```

属性

`PrimaryKey`

用作表主键的属性名称和类型。如果未提供，则主键将为`String`，其值为`id`。

> **Note**
>
> 创建此资源后，无法修改此属性的值。

类型：PrimaryKeyObject (p. 234)

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`ProvisionedThroughput`

读取和写入吞吐量配置信息。

如果`ProvisionedThroughput`未指定，则`BillingMode`将指定为`PAY_PER_REQUEST`。

类型：ProvisionedThroughput

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::DynamoDB::Table`资源的`ProvisionedThroughput`属性。

`SSESpecification`

指定用于启用服务器端加密的设置。

类型：SSESpecification

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::DynamoDB::Table`资源的`SSESpecification`属性。

`TableName`

DynamoDB 表的名称。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::DynamoDB::Table`资源的`TableName`属性。

`Tags`

一个映射（字符串到字符串），指定要添加的标签 SimpleTable。有关标签的有效密钥和值的详细信息，请参阅Amazon CloudFormation用户指南中的资源标签。

类型：地图

必需：否

Amazon CloudFormation兼容性：此属性类似于`AWS::DynamoDB::Table`资源的`Tags`属性。SAM 中的 Tags 属性由 Key: Value 对组成； CloudFormation 它由标签对象列表组成。

返回值

Ref

当该资源的逻辑 ID 提供给 Ref 内部函数时，将返回底层 DynamoDB 表的资源名称。

有关使用该`Ref`函数的更多信息，请参阅`Ref`《Amazon CloudFormation用户指南》。

示例

SimpleTableExample

的示例 SimpleTable

YAML

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

PrimaryKeyObject

描述主键属性的对象。

语法

要在您的Amazon Serverless Application Model(Amazon SAM) 模板，请使用以下语法。

YAML

```
  Name: String
  Type: String
```

属性

Name

主键的属性名称。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给AttributeName的财产AWS::DynamoDB::Table AttributeDefinition数据类型。

附加说明：此属性也被传递给AttributeName的财产AWS::DynamoDB::Table KeySchema数据类型。

Type

主键的数据类型。

有效值：String、Number、Binary

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给AttributeType的财产AWS::DynamoDB::Table AttributeDefinition数据类型。

示例

PrimaryKey

主键示例。

YAML

```
Properties:
```

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

AWS::Serverless::StateMachine

创建Amazon Step Functions状态机，您可以使用它来协调Amazon Lambda函数和其他Amazon资源，以形成复杂而强大的工作流程。

有关 Step Functions 的更多信息，请参阅 Amazon Step Functions 开发人员指南。

> Note
>
> 当您部署到时Amazon CloudFormation，Amazon SAM会将您的Amazon SAM资源转换为Amazon CloudFormation资源。有关更多信息，请参阅Amazon CloudFormation (p. 261)：

语法

要在Amazon Serverless Application Model（Amazon SAM）模板中声明此实体，请使用以下语法（）。

YAML

```
Type: AWS::Serverless::StateMachine
Properties:
  Definition: Map
  DefinitionSubstitutions: Map
  DefinitionUri: String | S3Location
  Events: EventSource (p. 239)
  Logging: LoggingConfiguration
  Name: String
  PermissionsBoundary: String
  Policies: String | List | Map
  RolePath: String
  Role: String
  Tags: Map
  Tracing: TracingConfiguration
  Type: String
```

属性

`Definition`

状态机定义是一个对象，其中的对象格式与 Amazon SAM 模板文件的格式相匹配，例如 JSON 或 YAML。状态机定义遵循 Amazon 状态语言。

有关行内状态机定义的示例，请参见示例 (p. 238)。

您必须提供 `aDefinition` 或 `aDefinitionUri`。

类型：地图

必需：条件

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`DefinitionSubstitutions`

指定状态机定义中占位符变量的映射。 string-to-string 这使您可以将运行时获得的值（例如，从内部函数中）。

类型：地图

必需：否

Amazon CloudFormation兼容性：此属性类似于AWS::StepFunctions::StateMachine资源的DefinitionSubstitutions属性。如果在行内状态机定义中指定了任何内部函数，则向该属性Amazon SAM添加条目以将其注入状态机定义。

`DefinitionUri`

状态机的 Amazon Simple Storage Service (Amazon S3) 的 Amazon S3)。

如果您提供本地文件路径，则模板必须通过包含sam deploy或sam package命令的工作流程才能正确转换定义。要执行此操作，必须使用 CLI 的 0.52.0 或更高版本的Amazon SAM CLI 的 0.52.0 或更高版本。

您必须提供 aDefinition 或 aDefinitionUri。

类型：字符串 | S3Location

必需：条件

Amazon CloudFormation兼容性：此属性直接传递给AWS::StepFunctions::StateMachine资源的DefinitionS3Location属性。

`Events`

指定触发此状态机的事件。事件由一种类型和一组依赖于该类型的属性组成。

类型：EventSource (p. 239)

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`Logging`

定义记录哪些执行历史事件以及它们的记录位置。

类型：LoggingConfiguration

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::StepFunctions::StateMachine资源的LoggingConfiguration属性。

`Name`

状态机的名称。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::StepFunctions::StateMachine资源的StateMachineName属性。

`PermissionsBoundary`

这个状态机的 ARN。只有在为你生成角色时，此属性才有效。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::IAM::Role资源的PermissionsBoundary属性。

`Policies`

此状态机的执行角色需要的一个或多个策略。

此属性接受一个字符串或字符串列表。该属性可以是Amazon托管Amazon Identity and Access Management (IAM) 策略的名称、Amazon SAM策略模板或一个或多个格式化为地图的内联策略文档。

如果设置了该Role属性，将忽略该属性。

类型：字符串 | 列表 | 地图

必需：条件

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`Role`

用作此状态机的 IAAM 角色的 IAAM 角色的 ARN。

类型：字符串

必需：条件

Amazon CloudFormation兼容性：此属性直接传递给AWS::StepFunctions::StateMachine资源的RoleArn属性。

`RolePath`

状态机的 IAM 执行角色的路径。

在为您生成角色时使用此属性。在使用Role属性指定角色时不要使用。

类型：字符串

必需：条件

Amazon CloudFormation兼容性：此属性直接传递给AWS::IAM::Role资源的Path属性。

`Tags`

一个 string-to-string 地图，用于指定添加到状态机的标签和相应的执行角色。有关标签的有效键和值的信息，请参阅AWS::StepFunctions::StateMachine资源的 Tags 属性。

类型：地图

必需：否

Amazon CloudFormation兼容性：此属性类似于AWS::StepFunctions::StateMachine资源的Tags属性。 Amazon SAM自动向该资源以及为其生成的默认角色添加stateMachine:createdBy:SAM标签。

`Tracing`

选择Amazon X-Ray是否启用状态机的。有关将 X-Ray 与步进函数结合使用的更多信息，请参阅Amazon Step Functions开发人员指南中的Amazon X-Ray和St ep Functions。

类型：TracingConfiguration

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::StepFunctions::StateMachine资源的TracingConfiguration属性。

Type

The type of the state machine.

有效值：STANDARD 或 EXPRESS

类型：字符串

必需：否

默认值：STANDARD

Amazon CloudFormation兼容性：此属性直接传递给AWS::StepFunctions::StateMachine资源的StateMachineType属性。

返回值

Ref

当向 Ref 内部函数提供此资源的逻辑 ID 时，Ref 将返回底层AWS::StepFunctions::StateMachine资源的 Amazon 资源名称（ARN）。

有关使用该Ref函数的更多信息，请参阅Ref《Amazon CloudFormation用户指南》。

Fn::GetAtt

Fn::GetAtt 返回一个此类型指定属性的值。以下为可用属性和示例返回值。

有关使用的更多信息Fn::GetAtt，请参阅Fn::GetAttAmazon CloudFormation用户指南中的。

Name

返回状态HelloWorld-StateMachine。

示例

状态机定义文件

以下是使用定义定义定义定义的状态机的。该my_state_machine.asl.json文件必须使用亚马逊州语言编写。

在此示例中，DefinitionSubstitution条目允许状态机包含在Amazon SAM模板文件中声明的资源。

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
```

行内状态机定义

以下是内联状态机定义的。

在此示例中，Amazon SAM模板文件是用 YAML 编写的，因此状态机定义也在 YAML 中。要在 JSON 中声明内联状态机定义，请在 JSON 中编写您的Amazon SAM模板文件。

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

描述触发状态机的事件来源的对象。每个事件都由一种类型和一组依赖于该类型的属性组成。有关适用于每个事件源的属性信息，请参阅与该类型的子主题对应的副主题。

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

```
  Properties: Schedule (p. 251) | ScheduleV2 (p. 256) | CloudWatchEvent (p. 246)
| EventBridgeRule (p. 247) | Api (p. 240)
  Type: String
```

属性

`Properties`

描述此事件映射属性的对象。属性集必须符合定义的属性Type。

类型：日程 (p. 251) | s cheduleV2 (p. 256) | CloudWatchEvent (p. 246)| EventBridgeRule (p. 247)| Ap i (p. 240)

必需：是

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

`Type`

事件类型。

有效值：`ApiSchedule`、`ScheduleV2`、`CloudWatchEvent`、`EventBridgeRule`

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

示例

API

以下是该API类型的示例事件。

YAML

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

Api

描述Api事件源类型的对象。如果定义了AWS::Serverless::Api (p. 83)资源，则路径和方法值必须对应于 API 的 OpenAPI 定义中的操作。

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

```
  Auth: ApiStateMachineAuth (p. 242)
  Method: String
  Path: String
  RestApiId: String
  UnescapeMappingTemplate (p. 241): Boolean
```

属性

`Auth`

此 API、路径和方法的授权配置。

当未`DefaultAuthorizer`指定路径时，使用此属性覆盖 API 对单个路径的`DefaultAuthorizer`设置，或覆盖默认`ApiKeyRequired`设置。

类型：ApiStateMachineAuth (p. 242)

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`Method`

调用此函数的 HTTP 方法。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`Path`

调用此函数的 URI 路径。该值必须以开头/。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`RestApiId`

`RestApi`资源的标识符，它必须包含具有给定路径和方法的操作。通常，将其设置为引用此模板中定义的AWS::Serverless::Api (p. 83)资源。

如果您未定义此属性，则使用生成的`OpenApi`文档Amazon SAM创建默认AWS::Serverless::Api (p. 83)资源。该资源包含所有路径和方法的组合，这些路径和方法由同一模板中的`Api`事件定义，这些事件未指定`RestApiId`。

此属性无法引用在其他模板中定义的AWS::Serverless::Api (p. 83)资源。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

`UnescapeMappingTemplate`

通过替换`\'`为，在传递给状态机的输入上取消单引号的转义。`'`当您的输入包含单引号时使用。

> **Note**
>
> 如果设置为，`False`并且您的输入包含单引号，则会出现错误。

类型：布尔值

必需：否

默认值：False

Amazon CloudFormation兼容性：此属性是唯一的Amazon SAM，没有Amazon CloudFormation等效属性。

示例

ApiEvent

以下是该`Api`类型事件的示例。

YAML

```
Events:
  ApiEvent:
    Type: Api
    Properties:
```

```
      Path: /path
      Method: get
      RequestParameters:
        - method.request.header.Authorization
```

ApiStateMachineAuth

在事件级别为特定 API、路径和方法配置授权。

语法

要在您的中声明此实体Amazon Serverless Application Model(Amazon SAM) 模板，请使用以下语法。

YAML

```
ApiKeyRequired: Boolean
AuthorizationScopes: List
Authorizer: String
ResourcePolicy: ResourcePolicyStatement (p. 243)
```

属性

`ApiKeyRequired`

需要此 API、路径和方法的 API 密钥。

类型：Boolean

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`AuthorizationScopes`

要应用于此 API、路径和方法的授权范围。

您指定的作用域将覆盖`DefaultAuthorizer`属性（如果你已经指定了它）。

类型：List

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`Authorizer`

这些区域有：`Authorizer`对于特定的状态机。

如果您已经为 API 指定了全局授权者并希望将此状态机公开，请通过设置设置覆盖全局授权者`Authorizer`到`NONE`.

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`ResourcePolicy`

为此 API 和路径配置资源策略。

类型：资源策略声明 (p. 243)

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

示例

State Machine Auth

以下示例指定状态机级别的授权。

YAML

```
Auth:
  ApiKeyRequired: true
  Authorizer: NONE
```

ResourcePolicyStatement

为 API 的所有方法和路径配置资源策略。有关资源策略的更多信息，请参阅《API Gateway 开发者指南》中的 “使用 API Gateway 资源策略控制 API 的访问”。

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

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

属性

`AwsAccountBlacklist`

要屏蔽的Amazon账户。

类型：字符串列表

必需：否

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

`AwsAccountWhitelist`

允许的Amazon账户。有关此属性的示例用法，请参阅本页底部的示例。

类型：字符串列表

必需：否

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

`CustomStatements`

适用于此 API 的自定义资源策略声明列表。有关此属性的示例用法，请参阅本页底部的示例。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

`IntrinsicVpcBlacklist`

要屏蔽的虚拟私有云 (VPC) 列表，其中每个 VPC 都指定为引用，例如动态引用或`Ref`内部函数。有关此属性的示例用法，请参阅本页底部的示例。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

`IntrinsicVpcWhitelist`

允许的 VPC 列表，其中每个 VPC 都指定为引用，例如动态引用或`Ref`内部函数。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

`IntrinsicVpceBlacklist`

要屏蔽的 VPC 终端节点列表，其中每个 VPC 终端节点都指定为引用，例如动态引用或`Ref`内部函数。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

`IntrinsicVpceWhitelist`

允许的 VPC 终端节点列表，其中每个 VPC 终端节点都指定为引用，例如动态引用或`Ref`内部函数。有关此属性的示例用法，请参阅本页底部的示例。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

`IpRangeBlacklist`

要屏蔽的 IP 地址或地址范围。有关此属性的示例用法，请参阅本页底部的示例。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

`IpRangeWhitelist`

允许的 IP 地址或地址范围。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

`SourceVpcBlacklist`

要屏蔽的源 VPC 或 VPC 终端节点。源 VPC 名称必须以开头`"vpc-"`，源 VPC 终端节点名称必须以开头`"vpce-"`。有关此属性的示例用法，请参阅本页底部的示例。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

`SourceVpcWhitelist`

允许的源 VPC 或 VPC 终端节点。源 VPC 名称必须以开头`"vpc-"`，源 VPC 终端节点名称必须以开头`"vpce-"`。

类型：清单

必需：否

Amazon CloudFormation兼容性：此属性是独有的Amazon SAM，没有Amazon CloudFormation等效属性。

示例

资源策略示例

以下示例屏蔽了两个 IP 地址和一个源 VPC，并允许一个Amazon账户。

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

描述CloudWatchEvent事件源类型的对象。

Amazon Serverless Application Model(Amazon SAM) 在设置此事件类型时生成AWS::Events::Rule资源。

重要说明：EventBridgeRule (p. 247)是首选使用的事件源类型，而不是CloudWatchEvent。EventBridgeRule并CloudWatchEvent使用相同的底层服务、API 和Amazon CloudFormation资源。但是，只Amazon SAM会添加对新功能的支持EventBridgeRule。

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

```
  EventBusName: String
  Input: String
  InputPath: String
  Pattern: EventPattern
```

属性

EventBusName

要与该规则关联的事件总线。如果省略此属性，则Amazon SAM使用默认事件总线。

类型：字符串

必填项：否

默认值：默认事件总线

Amazon CloudFormation兼容性：此属性直接传递给AWS::Events::Rule资源的EventBusName属性。

Input

传递到目标的有效 JSON 文本。如果使用此属性，则不会将事件文本本身的任何内容传递到目标。

类型：字符串

必填项：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Events::Rule Target资源的Input属性。

`InputPath`

当您不希望将整个匹配的事件传递到目标时，使用该`InputPath`属性描述要传递事件的哪个部分。

类型：字符串

必填项：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Events::Rule Target`资源的`InputPath`属性。

`Pattern`

描述哪些事件路由到指定目标。有关更多信息，请参阅 Amazon EventBridge 用户指南 EventBridge中的事件和事件模式。

类型：EventPattern

必填项：是

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Events::Rule`资源的`EventPattern`属性。

示例

CloudWatchEvent

以下是`CloudWatchEvent`事件源类型的示例。

YAML

```
CWEvent:
  Type: CloudWatchEvent
  Properties:
    Input: '{"Key": "Value"}'
    Pattern:
      detail:
        state:
          - running
```

EventBridgeRule

描述`EventBridgeRule`事件源类型的对象，它将您的状态机设置为 Amazon EventBridge 规则的目标。有关更多信息，请参 EventBridge阅 Amazon Amazon Amazon Ama 在亚马逊 EventBridge 用户指南中。

Amazon SAM设置此事件类型时生成AWS::Events::Rule资源。

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

```
  DeadLetterConfig: DeadLetterConfig (p. 249)
  EventBusName: String
  Input: String
  InputPath: String
  Name: String
  Pattern: EventPattern
  RetryPolicy: RetryPolicy
  Target (p. 249): Target (p. 251)
```

属性

`DeadLetterConfig`

配置 Amazon Simple Queue Service (Amazon SQS) Amazon Service Amazon Service (Amazon S EventBridge 例如，当向不存在的 Lambda 函数发送事件时，或者当权限不足以调用 Lambda 函数时 EventBridge ，调用可能会失败。有关更多信息，请参阅 Amazon Amazon Amazon Amazon Amazon Amazon Amazon Amaz EventBridge on Amazon

类型：DeadLetterConfig (p. 249)

必需：否

Amazon CloudFormation兼容性：此属性类似于`AWS::Events::RuleTarget`数据类型的`DeadLetterConfig`属性。此属性的Amazon SAM版本包括其他子属性，以防你Amazon SAM想为你创建死信队列。

`EventBusName`

要与该规则关联的事件总线。如果省略此属性，则Amazon SAM使用默认事件总线。

类型：字符串

必需：否

默认：默认事件总线

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Events::Rule`资源的`EventBusName`属性。

`Input`

传递到目标的有效 JSON 文本。如果使用此属性，则不会将事件文本本身的任何内容传递到目标。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Events::Rule Target`资源的`Input`属性。

`InputPath`

当您不希望传递整个匹配的事件时，使用`InputPath`属性描述将事件的哪个部分传递到目标。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Events::Rule Target`资源的`InputPath`属性。

`Name`

规则的名称。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Events::Rule`资源的`Name`属性。

`Pattern`

描述哪些事件路由到指定目标。有关更多信息，请参阅 Amazon Amazon Amazon Amazon Amazon Amaz EventBridge on Amazon EventBridge

类型：EventPattern

必需：是

Amazon CloudFormation兼容性：此属性直接传递给AWS::Events::Rule资源的EventPattern属性。

RetryPolicy

包含有关重试策略设置的信息的 RetryPolicy 对象。有关更多信息，请参阅 Amazon Amazon Amazon Amazon Amazon Amazon Amazon Amaz EventBridge on Amazon

类型：RetryPolicy

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Events::RuleTarget数据类型的RetryPolicy属性。

Target

触发规则时 EventBridge 调用的Amazon资源。您可以使用此属性指定目标的逻辑 ID。如果未指定此属性，则Amazon SAM生成目标的逻辑 ID。

类型：目标 (p. 251)

必需：否

Amazon CloudFormation兼容性：此属性类似于AWS::Events::Rule资源的Targets属性。此属性的Amazon SAM版本仅允许您指定单个目标的逻辑 ID。

示例

EventBridgeRule

以下是EventBridgeRule事件源类型的示例。

YAML

```
EBRule:
  Type: EventBridgeRule
  Properties:
    Input: '{"Key": "Value"}'
    Pattern:
      detail:
        state:
          - terminated
```

DeadLetterConfig

用于指定Amazon Simple Queue Service (Amazon SQS) 失败后 EventBridge 发送事件的对象。例如，当向不存在的状态机发送事件或权限不足以调用状态机时，调用可能会失败。有关更多信息，请参阅 Amazon EventBridge 用户指南中的事件重试策略和使用死信队列。

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

```
  Arn: String
```

```
  QueueLogicalId: String
  Type: String
```

属性

`Arn`

指定作为死信队列的目标的 Amazon SQS 队列。

Note

`Type`指定`Arn`两者。

类型：字符串

必填项：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Events::RuleDeadLetterConfig`数据类型的`Arn`属性。

`QueueLogicalId`

如果已指定，则Amazon SAM创建的死信队列`Type`的自定义名称。

Note

如果未设置该`Type`属性，则将忽略该属性。

类型：字符串

必填项：否

Amazon CloudFormation兼容性：此属性是独一无二的Amazon SAM，没有Amazon CloudFormation等效属性。

`Type`

队列。设置此属性后，Amazon SAM会自动创建死信队列并附加必要的基于资源的策略，以授予规则资源向队列发送事件的权限。

Note

`Type`指定`Arn`两者。

有效值：`SQS`

类型：字符串

必填项：否

Amazon CloudFormation兼容性：此属性是独一无二的Amazon SAM，没有Amazon CloudFormation等效属性。

示例

DeadLetterConfig

DeadLetterConfig

YAML

```
DeadLetterConfig:
  Type: SQS
```

```
  QueueLogicalId: MyDLQ
```

Target

配置触发规则时 EventBridge 调用的Amazon资源。

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

```
  Id: String
```

属性

`Id`

目标的逻辑 ID。

的值`Id`可以包含字母数字字符、句点 (`.`)、连字符 (`-`) 和下划线 (`_`)。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Events::RuleTarget`数据类型的`Id`属性。

示例

目标

YAML

```
EBRule:
  Type: EventBridgeRule
  Properties:
    Target:
      Id: MyTarget
```

Schedule

描述`Schedule`事件源类型的对象，它将您的状态机设置为按计划触发的 EventBridge 规则的目标。有关更多信息，请参阅 Amazon Eveue Eve E EventBridge v 在亚马逊 EventBridge 用户指南中。

Amazon Serverless Application Model(Amazon SAM) 在设置此事件类型时生成AWS::Events::Rule资源。

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

```
  DeadLetterConfig: DeadLetterConfig (p. 254)
  Description: String
  Enabled: Boolean
  Input: String
  Name: String
```

```
  RetryPolicy: RetryPolicy
  Schedule: String
  State: String
  Target (p. 253): Target (p. 255)
```

属性

`DeadLetterConfig`

配置 Amazon Simple Queue Service (Amazon SQS) 队列，以获取目标调用失败后 EventBridge 发送事件。例如，当向不存在的 Lambda 函数发送事件时，或者当权限不足以调用 Lambda 函数时 EventBridge ，调用可能会失败。有关更多信息，请参阅 Amazon Schedu le Evice Evice Evice Evic e Evice (A EventBridge mazon E v

类型：DeadLetterConfig (p. 254)

必需：否

Amazon CloudFormation兼容性：此属性类似于`AWS::Events::RuleTarget`数据类型的`DeadLetterConfig`属性。此属性的Amazon SAM版本包括其他子属性，以防你Amazon SAM想为你创建死信队列。

`Description`

规则的描述。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Events::Rule`资源的`Description`属性。

`Enabled`

指示是否启用规则。

要禁用该规则，请将此属性设置为`false`。

> **Note**
>
> 指定`Enabled`或`State`属性，但不能同时指定两者。

类型：布尔值

必需：否

Amazon CloudFormation兼容性：此属性类似于`AWS::Events::Rule`资源的`State`属性。如果将此属性设置为`true`，则Amazon SAM通过`ENABLED`，否则通过`DISABLED`。

`Input`

传递到目标的有效 JSON 文本。如果使用此属性，则不会将事件文本本身的任何内容传递到目标。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Events::Rule Target`资源的`Input`属性。

`Name`

规则的名称。如果不指定名称，则 Amazon CloudFormation 生成一个唯一物理 ID 并将该 ID 用作规则名称。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Events::Rule资源的Name属性。

RetryPolicy

包含有关重试策略设置的信息的 RetryPolicy 对象。有关更多信息，请参阅 Amazon Schedu le Evice Evice Evice Evic e Evice (A EventBridge mazon E v

类型：RetryPolicy

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Events::RuleTarget数据类型的RetryPolicy属性。

Schedule

决定运行规则的时间和频率的计划表达式。有关更多信息，请参阅规则的计划表达式。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性直接传递给AWS::Events::Rule资源的ScheduleExpression属性。

State

规则的状态。

可接受的值：DISABLED | ENABLED

Note

指定Enabled或State属性，但不能同时指定两者。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Events::Rule资源的State属性。

Target

触发规则时 EventBridge 调用的Amazon资源。您可以使用此属性指定目标的逻辑 ID。如果未指定此属性，则Amazon SAM生成目标的逻辑 ID。

类型：目标 (p. 251)

必需：否

Amazon CloudFormation兼容性：此属性类似于AWS::Events::Rule资源的Targets属性。此属性的Amazon SAM版本仅允许您指定单个目标的逻辑 ID。

示例

CloudWatch 计划事件

CloudWatch 计划事件示例

YAML

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

DeadLetterConfig

用于指定目标调Amazon Simple Queue Service (Amazon SQS) 目标调用失败后 EventBridge 发送事件。例如，当向不存在的状态机发送事件或权限不足以调用状态机时，调用可能会失败。有关更多信息，请参阅Azon U zon EventBridge Uzon SQueue ueueueueue。

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

```
  Arn: String
  QueueLogicalId: String
  Type: String
```

属性

`Arn`

指定作为死信队列队列队列队列队列队列的目标信队列队列Amazon SQS 列的目标信队列队列的目标信队列的目标信队列的目标信队列

> **Note**
>
> 指定两者`Type`Arn，但不能同时指定两者。

类型：字符串

必填项：否

Amazon CloudFormation兼容性：此属性直接传递给`AWS::Events::RuleDeadLetterConfig`数据类型的`Arn`属性。

`QueueLogicalId`

如果已指定，则Amazon SAM创建的死信队列`Type`的自定义名称。

> **Note**
>
> 如果未设置该`Type`属性，将忽略该属性。

类型：字符串

必填项：否

Amazon CloudFormation兼容性：此属性是独一无二的Amazon SAM，没有Amazon CloudFormation等效属性。

`Type`

队列的类型。设置此属性后，Amazon SAM会自动创建死信队列并附加必要的基于资源的策略，以授予规则资源向队列发送事件的权限。

Note

指定两者TypeArn，但不能同时指定两者。

有效值：SQS

类型：字符串

必填项：否

Amazon CloudFormation兼容性：此属性是独一无二的Amazon SAM，没有Amazon CloudFormation等效属性。

示例

DeadLetterConfig

DeadLetterConfig

YAML

```
DeadLetterConfig:
  Type: SQS
  QueueLogicalId: MyDLQ
```

Target

配置触发规则时 EventBridge 调用的Amazon资源。

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

```
  Id: String
```

属性

`Id`

目标的逻辑 ID。

的值Id可以包含字母数字字符、句点 (.)、连字符 (-) 和下划线 (_)。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性直接传递给AWS::Events::RuleTarget数据类型的Id属性。

示例

目标

YAML

```
EBRule:
```

```
  Type: Schedule
  Properties:
    Target:
      Id: MyTarget
```

ScheduleV2

描述ScheduleV2事件源类型的对象，它将您的状态机设置为按计划触发的 Amazon S EventBridge cheduler 事件的目标。有关更多信息，请参阅什么是 Amazon S EventBridge chedule？ 在《EventBridge 调度程序用户指南》中。

Amazon Serverless Application Model(Amazon SAM) 在设置此事件类型时生成AWS::Scheduler::Schedule资源。

语法

要在Amazon Serverless Application Model (Amazon SAM) 模板中声明此实体，请使用以下语法。

YAML

```
DeadLetterConfig (p. 256): DeadLetterConfig (p. 254)
Description: String
EndDate: String
FlexibleTimeWindow: FlexibleTimeWindow (p. 257)
GroupName: String
Input: String
KmsKeyArn: String
Name: String
PermissionsBoundary: String
RetryPolicy: RetryPolicy (p. 258)
RoleArn: String
ScheduleExpression: String
ScheduleExpressionTimezone: String
StartDate: String
State: String
```

属性

`DeadLetterConfig`

配置 Amazon Simple Queue Service (Amazon SQS) 队列，该队列将在目标调用失败后 EventBridge 发送事件。例如，当向不存在的 Lambda 函数发送事件时，或者没有足够的权限调用 Lambda 函数时 EventBridge ，调用可能会失败。有关更多信息，请参阅《调度程序用户指南》中的为 EventBridge 调度器配置死信队列。EventBridge

类型：DeadLetterConfig (p. 254)

必需：否

Amazon CloudFormation兼容性：此属性类似于AWS::Scheduler::ScheduleTarget数据类型的DeadLetterConfig属性。此属性的Amazon SAM版本包括其他子属性，以防你Amazon SAM想为自己创建死信队列。

`Description`

计划的描述。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Scheduler::Schedule资源的Description属性。

EndDate

日程可以调用其目标的日期（以 UTC 为单位）。根据调度的重复表达式，调用可能会在EndDate您指定的日期或之前停止。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Scheduler::Schedule资源的EndDate属性。

FlexibleTimeWindow

允许配置一个窗口，在该窗口中可以调用时间表。

类型：FlexibleTimeWindow

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Scheduler::Schedule资源的FlexibleTimeWindow属性。

GroupName

与此计划的关联计划的计划组的名称。如果未定义，则使用默认组。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Scheduler::Schedule资源的GroupName属性。

Input

传递到目标的有效 JSON 文本。如果使用此属性，则不会将事件文本本身的任何内容传递到目标。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Scheduler::Schedule Target资源的Input属性。

KmsKeyArn

将用于加密客户数据的 KMS 密钥的 ARN。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Scheduler::Schedule资源的KmsKeyArn属性。

Name

计划的名称。如果未指定名称，Amazon SAM将生成一个格式的名称，*StateMachine-Logical-IDEvent-Source-Name*并使用该 ID 作为计划的名称。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Scheduler::Schedule资源的Name属性。

PermissionsBoundary

用于为角色设置权限边界的策略的 ARN。

Note

如果已定义，PermissionsBoundary则Amazon SAM将对调度程序计划的目标 IAM 角色应用相同的边界。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::IAM::Role资源的PermissionsBoundary属性。

RetryPolicy

包含有关重试策略设置的信息的 RetryPolicy 对象。

类型：RetryPolicy

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Scheduler::ScheduleTarget数据类型的RetryPolicy属性。

RoleArn

计划启动计划时， EventBridge 计划程序将用于目标的 IAM 角色的 ARN。

类型：RoleArn

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Scheduler::ScheduleTarget数据类型的RoleArn属性。

ScheduleExpression

决定运行计划的计划的时间和频率的计划表达式。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性直接传递给AWS::Scheduler::Schedule资源的ScheduleExpression属性。

ScheduleExpressionTimezone

评估调度表达式的时区。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Scheduler::Schedule资源的ScheduleExpressionTimezone属性。

StartDate

以 UTC 为单位的日期，在此日期之后，计划可以开始调用目标。根据调度的重复表达式，调用可能会在StartDate您指定的时间或之后发生。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Scheduler::Schedule资源的StartDate属性。

State

计划的状态。

可接受的值：DISABLED | ENABLED

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性直接传递给AWS::Scheduler::Schedule资源的State属性。

示例

定义 ScheduleV2 资源的基本示例

```
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
```

有关所有Amazon资源和属性类型Amazon CloudFormation及Amazon SAM支持的参考信息，请参阅Amazon CloudFormation用户指南中的Amazon资源和属性类型参考。

资源属性

资源属性是可以添加到的属性Amazon SAM和Amazon CloudFormation用于控制其他行为和关系的资源。有关资源属性的更多信息，请参阅。资源属性引用中的Amazon CloudFormation用户指南.

Amazon SAM支持资源属性的子集，这些属性由Amazon CloudFormation. 在受支持的资源属性中，有些属性仅复制到生成的基础Amazon CloudFormation相应的资源Amazon SAM资源，有些被复制到所有生成Amazon CloudFormation来自相应的资源Amazon SAM资源。有关 的更多信息Amazon CloudFormation从相应生成的资源Amazon SAM资源，请参阅。Amazon CloudFormation (p. 261).

下表汇总了资源属性支持的方式。Amazon SAM，取决于异常 (p. 260)下面列出了。

资源属性	目标生成的资源
DependsOn Metadata [1、2]	BaseAmazon CloudFormation仅限生成的资源。有关之间映射的信息Amazon SAM资源和基础Amazon CloudFormation资源，请参阅。生成的Amazon CloudFormation资源方案 (p. 262).
Condition DeletionPolicy UpdateReplacePolicy	生成的所有Amazon CloudFormation来自相应的资源Amazon SAM资源。有关生成的方案的信息Amazon CloudFormation资源，请参阅。生成的Amazon CloudFormation资源方案 (p. 262).

备注:

1. 有关如何使用的更多信息Metadata资源属性带AWS::Serverless::Function资源类型，请参阅。构建自定义运行时 (p. 355).
2. 有关如何使用的更多信息Metadata资源属性带AWS::Serverless::LayerVersion资源类型，请参阅。建筑层 (p. 354).

异常

前面描述的资源属性规则有许多例外情况：

- 适用于AWS::Lambda::LayerVersion，Amazon SAM只有自定义字段RetentionPolicy设置DeletionPolicy对于生成的Amazon CloudFormation资源的费用。这优先级高于DeletionPolicy本身。如果两者都不设置，那么默认情况下DeletionPolicy设置为Retain.
- 适用于AWS::Lambda::Version，如果DeletionPolicy未指定，默认值为Retain.
- 对于那样的场景DeploymentPreferences是为无服务器函数指定的，资源属性不会复制到以下生成的Amazon CloudFormation资源：
  - AWS::CodeDeploy::Application
  - AWS::CodeDeploy::DeploymentGroup
  - 这些区域有：AWS::IAM::Role被命名CodeDeployServiceRole是为此场景创建的
- 如果您的Amazon SAM模板包含多个函数，其中包含隐式创建的 API 事件源，然后这些函数将共享生成的AWS::ApiGateway::RestApi资源。在这种情况下，如果函数具有不同的资源属性，那么对于生成的AWS::ApiGateway::RestApi资源，Amazon SAM根据以下优先级列表复制资源属性：
  - UpdateReplacePolicy:

1. Retain
2. Snapshot
3. Delete

- DeletionPolicy:
  1. Retain
  2. Delete

内部函数

内部函数是内部函数，以便为仅在运行时可用的属性分配值。有关内部函数的更多信息，请参阅固有功能参考中的Amazon CloudFormation用户指南.

Amazon CloudFormation

当Amazon Serverless Application Model (Amazon SAM) 处理您的Amazon SAM模板文件时，它会生成一个或多个Amazon CloudFormation资源。Amazon SAM生成的Amazon CloudFormation资源集因您指定的方案而异。场景是在模板文件中指定的Amazon SAM资源和属性的组合。您可以在模板文件中的其他地方引用生成Amazon CloudFormation的资源，这与引用在模板文件中明确声明的资源类似。

例如，如果您在Amazon SAM模板文件中指定了`AWS::Serverless::Function`资源，则Amazon SAM始终会生成`AWS::Lambda::Function`基础资源。如果您还指定了可选`AutoPublishAlias`属性，则Amazon SAM还会生成`AWS::Lambda::Alias`和`AWS::Lambda::Version`资源。

本节列出了场景及其生成Amazon CloudFormation的资源，并说明了如何在Amazon SAM模板文件中引用生成Amazon CloudFormation的资源。

引用生成Amazon CloudFormation的资源

您可以通过两种方式在Amazon SAM模板文件中引用生成Amazon CloudFormation的资源，即按可引用属性`LogicalId`或按可引用属性引用生成的资源。

通过以下方式引用生成Amazon CloudFormation的资源 LogicalId

Amazon CloudFormationAmazon SAM`LogicalId`生成的 Amazon SAM使用模板文件`LogicalIds`中的Amazon SAM资源来构造它生成Amazon CloudFormation的资源。`LogicalIds`您可以使用生成的Amazon CloudFormation资源在模板文件中访问该资源的属性，就像使用已明确声明的Amazon CloudFormation资源一样。`LogicalId`有关`LogicalIds`中Amazon CloudFormation和Amazon SAM模板的更多信息，请参阅《Amazon CloudFormation用户指南》中的资源。

> Note
>
> 一些`LogicalIds`生成的资源包含一个唯一的哈希值，以避免命名空间冲突。`LogicalIds` 只有在使用、或其中一个Amazon SDK 创建堆栈后Amazon Web Services Management ConsoleAmazon CLI，才能检索它们。我们不建议通过引用这些资源，`LogicalId`因为哈希值可能会发生变化。

通过可引用属性引用生成Amazon CloudFormation的资源

对于某些生成的资源，Amazon SAM提供该Amazon SAM资源的可引用属性。您可以使用此属性在Amazon SAM模板文件中引用生成的Amazon CloudFormation资源及其属性。

> Note
>
> 并非所有生成的Amazon CloudFormation资源都具有可引用的属性。`LogicalId`

生成的Amazon CloudFormation资源方案

下表汇总了构成生成Amazon SAM资源的方案的Amazon CloudFormation资源和属性。场景列中的主题提供了有关为该场景Amazon SAM生成的其他Amazon CloudFormation资源的详细信息。

Amazon SAM 资源	基础Amazon CloudFormation 资源	场景
AWS::Serverless::Api (p. 263)	AWS::ApiGateway::RestApi	• 指定了 DomainName 属性 (p. 263) • 指定了 UsagePlan 属性 (p. 263)
Amazon# # #### # (p. 264)	AWS::CloudFormation::Stack	• 除了生成基础Amazon CloudFormation资源外，此无服务器资源没有其他场景。
AWS::Serverless::Function (p. 265)	AWS::Lambda::Function	• 指定了 AutoPublishAlias 属性 (p. 265) • 未指定角色属性 (p. 266) • 指定了 DeploymentPreference 属性 (p. 266) • 指定了 Api 事件源 (p. 266) • 指定了 HTTPAPI 事件源 (p. 266) • 指定了流媒体事件源 (p. 267) • 指定了事件桥（或事件总线）事件源 (p. 267) • 指定了 iotRule 事件源 (p. 267) • 为 Amazon SNS 事件指定了 onSuccess（或 onFailure）属性 (p. 267) • 为 Amazon SQS 事件指定了 onSuccess（或 onFailure）属性 (p. 268)
AWS::Serverless::HttpApi (p. 268)	AWS::ApiGatewayV2::Api	• 指定了 StageName 属性 (p. 268) • StageName 属性是不指定的 (p. 269) • 指定了 DomainName 属性 (p. 269)
AWS::Serverless::LayerVersion (p. 269)	AWS::Lambda::LayerVersion	• 除了生成基础Amazon CloudFormation资源外，此无服务器资源没有其他场景。
AWS::Serverless::SimpleTable (p. 269)	AWS::DynamoDB::Table	• 除了生成基础Amazon CloudFormation资源外，此无服务器资源没有其他场景。
AWS::Serverless::StateMachine (p. 270)	AWS::StepFunctions::StateMachine	• 未指定角色属性 (p. 270) • 指定了 Api 事件源 (p. 270) • 指定了事件桥（或事件总线）事件源 (p. 270)

主题

- Amazon CloudFormation指定 Amazon# Serverless# Api 时生成的资源 (p. 263)
- Amazon CloudFormation指定 Amazon# Serverless# 应用程序时生成的资源 (p. 264)
- Amazon CloudFormation指定时生成的资源AWS::Serverless::Connector (p. 264)
- Amazon CloudFormation生成的资源时间AWS::Serverless::Function已指定 (p. 265)
- Amazon CloudFormation指定 Amazon# Serverless# HTTPAPI 时生成的资源 (p. 268)
- Amazon CloudFormation当时生成的资源AWS::Serverless::LayerVersion已指定 (p. 269)
- Amazon CloudFormation当时生成的资源AWS::Serverless::SimpleTable已指定 (p. 269)

- Amazon CloudFormation生成的资源AWS::Serverless::StateMachine已指定 (p. 270)

Amazon CloudFormation指定 Amazon# Serverless# Api 时生成的资源

当您时AWS::Serverless::Api已指定，Amazon Serverless Application Model(Amazon SAM)始终生成AWS::ApiGateway::RestApi基础Amazon CloudFormation资源。此外，它还总是生成AWS::ApiGateway::Stage和AWS::ApiGateway::Deployment资源。

AWS::ApiGateway::RestApi

LogicalId: *<api-LogicalId>*

可参考的属性：N/A（你必须使用LogicalId引用此Amazon CloudFormation资源)

AWS::ApiGateway::Stage

LogicalId: *<api-LogicalId><stage-name>*Stage

*<stage-name>*是的字符串StageName属性将设置为。例如，如果您设置StageName到Gamma，LogicalId是*MyRestApiGamma*Stage.

可参考的属性：*<api-LogicalId>*.Stage

AWS::ApiGateway::Deployment

LogicalId: *<api-LogicalId>*Deployment*<sha>*

*<sha>*是创建堆栈时生成的唯一哈希值。例如：*MyRestApi*Deployment*926eeb5ff1*。

可参考的属性：*<api-LogicalId>*.Deployment

除此之外Amazon CloudFormation资源，什么时候AWS::Serverless::Api已指定，Amazon SAM生成额外的Amazon CloudFormation用于以下场景的资源。

方案

- 指定了 DomainName 属性 (p. 263)
- 指定了 UsagePlan 属性 (p. 263)

指定了 DomainName 属性

当您时DomainName的财产Domain的财产AWS::Serverless::Api已指定，Amazon SAM生成AWS::ApiGateway::DomainName Amazon CloudFormation资源。

AWS::ApiGateway::DomainName

LogicalId: ApiGatewayDomainName*<sha>*

*<sha>*是创建堆栈时生成的唯一哈希值。例如：ApiGatewayDomainName*926eeb5ff1*。

可参考的属性：*<api-LogicalId>*.DomainName

指定了 UsagePlan 属性

当您时UsagePlan的财产Auth的财产AWS::Serverless::Api已指定，Amazon SAM生成以下内容Amazon CloudFormation资

源：AWS::ApiGateway::UsagePlan、AWS::ApiGateway::UsagePlanKey, 和AWS::ApiGateway::ApiKey.

AWS::ApiGateway::UsagePlan

LogicalId: *<api-LogicalId>*UsagePlan

可参考的属性：*<api-LogicalId>*.UsagePlan

AWS::ApiGateway::UsagePlanKey

LogicalId: *<api-LogicalId>*UsagePlanKey

可参考的属性：*<api-LogicalId>*.UsagePlanKey

AWS::ApiGateway::ApiKey

LogicalId: *<api-LogicalId>*ApiKey

可参考的属性：*<api-LogicalId>*.ApiKey

Amazon CloudFormation指定 Amazon# Serverless# 应用程序时生成的资源

当AWS::Serverless::Application已指定，Amazon Serverless Application Model(Amazon SAM) 会生成AWS::CloudFormation::Stack基础Amazon CloudFormation资源。

AWS::CloudFormation::Stack

LogicalId: *<application-LogicalId>*

可参考的属性：N/A（你必须使用LogicalId引用Amazon CloudFormation资源)

Amazon CloudFormation指定时生成的资源 AWS::Serverless::Connector

Note

当您通过 embeddeConnectors d 属性定义连接器时，它首先会转换为AWS::Serverless::Connector资源，然后再生成这些资源。

在Amazon SAM模板中指定AWS::Serverless::Connector资源时，Amazon SAM会根据需要生成以下Amazon CloudFormation资源。

AWS::IAM::ManagedPolicy

LogicalId:*<connector-LogicalId>*Policy

可引用属性：N/A（要引用此Amazon CloudFormation资源，必须使用LogicalId。）

AWS::SNS::TopicPolicy

LogicalId:*<connector-LogicalId>*TopicPolicy

可引用属性：N/A（要引用此Amazon CloudFormation资源，必须使用LogicalId。）

AWS::SQS::QueuePolicy

LogicalId:*<connector-LogicalId>*QueuePolicy

可引用属性：N/A（要引用此Amazon CloudFormation资源，必须使用LogicalId。）

AWS::Lambda::Permission

LogicalId:*<connector-LogicalId>*<*permission*>LambdaPermission

*<permission>*是Permissions属性指定的权限。例如，Write。

可引用属性：N/A（要引用此Amazon CloudFormation资源，必须使用LogicalId。）

Amazon CloudFormation生成的资源时间AWS::Serverless::Function已指定

当您时AWS::Serverless::Function已指定，Amazon Serverless Application Model(Amazon SAM) 始终创建AWS::Lambda::Function基础Amazon CloudFormation资源。

AWS::Lambda::Function

LogicalId: *<function-LogicalId>*

可参考的属性：N/A（你必须使用LogicalId引用这个Amazon CloudFormation资源)

除此之外Amazon CloudFormation资源，什么时候AWS::Serverless::Function已指定，Amazon SAM还会生成Amazon CloudFormation用于以下场景的资源。

方案

- 指定了 AutoPublishAlias 属性 (p. 265)
- 未指定角色属性 (p. 266)
- 指定了 DeploymentPreference 属性 (p. 266)
- 指定了 Api 事件源 (p. 266)
- 指定了 HTTPAPI 事件源 (p. 266)
- 指定了流媒体事件源 (p. 267)
- 指定了事件桥（或事件总线）事件源 (p. 267)
- 指定了 iotrRule 事件源 (p. 267)
- 为 Amazon SNS 事件指定了 onSuccess（或 onFailure）属性 (p. 267)
- 为 Amazon SQS 事件指定了 onSuccess（或 onFailure）属性 (p. 268)

指定了 AutoPublishAlias 属性

当您时AutoPublishAlias一个的财产AWS::Serverless::Function已指定，Amazon SAM生成以下内容Amazon CloudFormation资源：AWS::Lambda::Alias和AWS::Lambda::Version.

AWS::Lambda::Alias

LogicalId: *<function-LogicalId>*Alias*<alias-name>*

*<alias-name>*是字符串AutoPublishAlias设置为。例如，如果您设置AutoPublishAlias到live，LogicalId是：*myFunction*别名*##*.

可参考的属性：*<function-LogicalId>*.Alias

AWS::Lambda::Version

LogicalId: *<function-LogicalId>*Version*<sha>*

*<sha>*是创建堆栈时生成的唯一哈希值。例如，*myFunction*版本*926eeb5ff1*.

可参考的属性：*<function-LogicalId>*.Version

未指定角色属性

当您时Role一个的财产AWS::Serverless::Function是不指定，Amazon SAM生成AWS::IAM::Role Amazon CloudFormation资源。

AWS::IAM::Role

LogicalId: *<function-LogicalId>*Role

可参考的属性：N/A（你必须使用LogicalId引用这个Amazon CloudFormation资源）

指定了 DeploymentPreference 属性

当您时DeploymentPreference一个的财产AWS::Serverless::Function已指定，Amazon SAM生成以下资源：Amazon CloudFormation资源：AWS::CodeDeploy::Application和AWS::CodeDeploy::DeploymentGroup. 此外，如果Role的财产DeploymentPreference对象是不指定，Amazon SAM还会生成AWS::IAM::Role Amazon CloudFormation资源。

AWS::CodeDeploy::Application

LogicalId: ServerlessDeploymentApplication

可参考的属性：N/A（你必须使用LogicalId引用这个Amazon CloudFormation资源）

AWS::CodeDeploy::DeploymentGroup

LogicalId: *<function-LogicalId>*DeploymentGroup

可参考的属性：N/A（你必须使用LogicalId引用这个Amazon CloudFormation资源）

AWS::IAM::Role

LogicalId: CodeDeployServiceRole

可参考的属性：N/A（你必须使用LogicalId引用这个Amazon CloudFormation资源）

指定了 Api 事件源

当您时Event一个的财产AWS::Serverless::Function设置为Api，但是RestApiId属性是不指定，Amazon SAM生成AWS::ApiGateway::RestApi Amazon CloudFormation资源。

AWS::ApiGateway::RestApi

LogicalId: ServerlessRestApi

可参考的属性：N/A（你必须使用LogicalId引用这个Amazon CloudFormation资源）

指定了 HTTPAPI 事件源

当您时Event一个的财产AWS::Serverless::Function设置为HttpApi，但是ApiId属性是不指定，Amazon SAM生成AWS::ApiGatewayV2::Api Amazon CloudFormation资源。

AWS::ApiGatewayV2::Api

LogicalId: ServerlessHttpApi

可参考的属性：N/A（你必须使用LogicalId引用这个Amazon CloudFormation资源）

指定了流媒体事件源

当您时Event一个的财产AWS::Serverless::Function被设置为其中一种流媒体类型，Amazon SAM生成AWS::Lambda::EventSourceMapping Amazon CloudFormation资源。这适用于以下类型：DynamoDB、Kinesis、MQ、MSK, 和SQS.

AWS::Lambda::EventSourceMapping

LogicalId: *<function-LogicalId><event-LogicalId>*

可参考的属性：N/A（你必须使用LogicalId引用这个Amazon CloudFormation资源）

指定了事件桥（或事件总线）事件源

当您时Event一个的财产AWS::Serverless::Function被设置为其中一种事件桥（或事件总线）类型，Amazon SAM生成AWS::Events::Rule Amazon CloudFormation资源。这适用于以下类型：EventBridgeRule、Schedule, 和CloudWatchEvents.

AWS::Events::Rule

LogicalId: *<function-LogicalId><event-LogicalId>*

可参考的属性：N/A（你必须使用LogicalId引用这个Amazon CloudFormation资源）

指定了 iotrRule 事件源

当您时Event一个的财产AWS::Serverless::Function设置为 ioTRULE，Amazon SAM生成AWS::IoT::TopicRule Amazon CloudFormation资源。

AWS::IoT::TopicRule

LogicalId: *<function-LogicalId><event-LogicalId>*

可参考的属性：N/A（你必须使用LogicalId引用这个Amazon CloudFormation资源）

为 Amazon SNS 事件指定了 onSuccess（或 onFailure）属性

当您时OnSuccess（或OnFailure) 的属性DestinationConfig的财产EventInvokeConfig一个的财产AWS::Serverless::Function已指定，目标类型为SNS但目的地 ARN 是不指定，Amazon SAM生成以下内容Amazon CloudFormation资源：AWS::Lambda::EventInvokeConfig和AWS::SNS::Topic.

AWS::Lambda::EventInvokeConfig

LogicalId: *<function-LogicalId>*EventInvokeConfig

可参考的属性：N/A（你必须使用LogicalId引用这个Amazon CloudFormation资源）

AWS::SNS::Topic

LogicalId：*<function-LogicalId>*OnSuccessTopic（或*<function-LogicalId>*OnFailureTopic）

可参考的属性：*<function-LogicalId>*.DestinationTopic

如果两者都OnSuccess和OnFailure是为 Amazon SNS 事件指定的，要区分生成的资源，您必须使用LogicalId.

为 Amazon SQS 事件指定了 onSuccess（或 onFailure）属性

当您时OnSuccess（或OnFailure) 的属性DestinationConfig的财产EventInvokeConfig一个的财产AWS::Serverless::Function已指定，目标类型为SQS但目的地 ARN 是不指定，Amazon SAM生成以下内容Amazon CloudFormation资源：AWS::Lambda::EventInvokeConfig和AWS::SQS::Queue.

AWS::Lambda::EventInvokeConfig

LogicalId: *<function-LogicalId>*EventInvokeConfig

可参考的属性：N/A（你必须使用LogicalId引用这个Amazon CloudFormation资源)

AWS::SQS::Queue

LogicalId：*<function-LogicalId>*OnSuccessQueue（或*<function-LogicalId>*OnFailureQueue）

可参考的属性：*<function-LogicalId>*.DestinationQueue

如果两者都OnSuccess和OnFailure是针对 Amazon SQS 事件指定的，为了区分生成的资源，必须使用LogicalId.

Amazon CloudFormation指定 Amazon# Serverless# HTTPAPI 时生成的资源

当您时AWS::Serverless::HttpApi已指定，Amazon Serverless Application Model(Amazon SAM) 生成AWS::ApiGatewayV2::Api基础Amazon CloudFormation资源。

AWS::ApiGatewayV2::Api

LogicalId: *<httpapi-LogicalId>*

可参考属性：N/A（你必须使用LogicalId引用此Amazon CloudFormation资源)

此外Amazon CloudFormation资源，何时AWS::Serverless::HttpApi已指定，Amazon SAM还会生成Amazon CloudFormation以下场景的资源：

方案

- 指定了 StageName 属性 (p. 268)
- StageName 属性是不指定的 (p. 269)
- 指定了 DomainName 属性 (p. 269)

指定了 StageName 属性

当您时StageName的财产AWS::Serverless::HttpApi已指定，Amazon SAM生成AWS::ApiGatewayV2::Stage Amazon CloudFormation资源。

AWS::ApiGatewayV2::Stage

LogicalId: *<httpapi-LogicalId><stage-name>*Stage

*<stage-name>*是的字符串StageName属性设置为。例如，如果您设置StageName到Gamma，LogicalId是：*myhttPapigamma*阶段。

可参考属性：*<httpapi-LogicalId>*.Stage

StageName 属性是不指定的

当您时StageName的财产AWS::Serverless::HttpApi是不指定，Amazon SAM生成AWS::ApiGatewayV2::Stage Amazon CloudFormation资源。

AWS::ApiGatewayV2::Stage

LogicalId: *<httpapi-LogicalId>*ApiGatewayDefaultStage

可参考属性：*<httpapi-LogicalId>*.Stage

指定了 DomainName 属性

当您时DomainName的财产Domain的财产AWS::Serverless::HttpApi已指定，Amazon SAM生成AWS::ApiGatewayV2::DomainName Amazon CloudFormation资源。

AWS::ApiGatewayV2::DomainName

LogicalId: ApiGatewayDomainNameV2*<sha>*

*<sha>*是创建堆栈时生成的唯一哈希值。例如，ApiGatewayDomainNameV2*926eeb5ff1*.

可参考属性：*<httpapi-LogicalId>*.DomainName

Amazon CloudFormation当时生成的资源 AWS::Serverless::LayerVersion已指定

当AWS::Serverless::LayerVersion已指定，Amazon Serverless Application Model(Amazon SAM) 生成AWS::Lambda::LayerVersion基础Amazon CloudFormation资源。

AWS::Lambda::LayerVersion

LogicalId: *<layerversion-LogicalId>*

可参考的属性：N/A（你必须使用LogicalId引用这个Amazon CloudFormation资源)

Amazon CloudFormation当时生成的资源 AWS::Serverless::SimpleTable已指定

当AWS::Serverless::SimpleTable已指定，Amazon Serverless Application Model(Amazon SAM) 生成AWS::DynamoDB::Table基础Amazon CloudFormation资源。

AWS::DynamoDB::Table

LogicalId: *<simpletable-LogicalId>*

可参考的属性：N/A（你必须使用LogicalId来引用这个Amazon CloudFormation资源)

Amazon CloudFormation生成的资源AWS::Serverless::StateMachine已指定

当您时AWS::Serverless::StateMachine已指定，Amazon Serverless Application Model(Amazon SAM) 生成AWS::StepFunctions::StateMachine基础Amazon CloudFormation资源。

AWS::StepFunctions::StateMachine

LogicalId: *<statemachine-LogicalId>*

可参考的属性：N/A（你必须使用LogicalId引用这个Amazon CloudFormation资源)

此外Amazon CloudFormation资源，什么时候AWS::Serverless::StateMachine已指定，Amazon SAM还会生成Amazon CloudFormation将用于以下方案的资源：

方案

- 未指定角色属性 (p. 270)
- 指定了 Api 事件源 (p. 270)
- 指定了事件桥（或事件总线）事件源 (p. 270)

未指定角色属性

当您时Role一个的财产AWS::Serverless::StateMachine是不已指定，Amazon SAM生成AWS::IAM::Role Amazon CloudFormation资源。

AWS::IAM::Role

LogicalId: *<statemachine-LogicalId>*Role

可参考的属性：N/A（你必须使用LogicalId引用这个Amazon CloudFormation资源)

指定了 Api 事件源

当您时Event一个的财产AWS::Serverless::StateMachine设置为Api，但是RestApiId属性是不已指定，Amazon SAM生成AWS::ApiGateway::RestApi Amazon CloudFormation资源。

AWS::ApiGateway::RestApi

LogicalId: ServerlessRestApi

可参考的属性：N/A（你必须使用LogicalId引用这个Amazon CloudFormation资源)

指定了事件桥（或事件总线）事件源

当您时Event一个的财产AWS::Serverless::StateMachine被设置为其中一种事件桥（或事件总线）类型，Amazon SAM生成AWS::Events::Rule Amazon CloudFormation资源。这适用于以下类型：EventBridgeRule、Schedule, 和CloudWatchEvents.

AWS::Events::Rule

LogicalId: *<statemachine-LogicalId><event-LogicalId>*

可参考的属性：N/A（你必须使用LogicalId引用这个Amazon CloudFormation资源)

API Gateway 扩展

API Gateway 扩展是基于 OpenAPI 规范的扩展，支持Amazon特定于的授权和 API Gateway 特定的 API 集成。有关 API Gateway 扩展的更多信息，请参阅基于 OpenAPI 的 API Gateway 扩展.

Amazon SAM支持 API Gateway 扩展的一部分。查看哪些 API Gateway 扩展支持Amazon SAM，请参阅下表。

API Gateway 扩展	支持Amazon SAM
x-amazon-apigateway-any-method 对象	是
x-amazon-apigateway-api-key-source 属性	否
x-amazon-apigateway-auth 对象	是
x-amazon-apigateway-authorizer 对象	是
x-amazon-apigateway-authtype 属性	是
x-amazon-apigateway-binary-media-types 属性	是
x-amazon-apigateway-documentation 对象	否
x-amazon-apigateway-endpoint-configuration 对象	否
x-amazon-apigateway-gateway-responses 对象	是
x-amazon-apigateway-gateway-responses.gatewayResponse 对象	是
x-amazon-apigateway-gateway-responses.responseParameters 对象	是
x-amazon-apigateway-gateway-responses.responseTemplates 对象	是
x-amazon-apigateway-integration 对象	是
x-amazon-apigateway-integration.requestTemplates 对象	是
x-amazon-apigateway-integration.requestParameters 对象	否
x-amazon-apigateway-integration.responses 对象	是
x-amazon-apigateway-integration.response 对象	是
x-amazon-apigateway-integration.responseTemplates 对象	是
x-amazon-apigateway-integration.responseParameters 对象	是
x-amazon-apigateway-request-validator 属性	否
x-amazon-apigateway-request-validators 对象	否
x-amazon-apigateway-request-validators.requestValidator 对象	否

创作无服务器应用程序

使用编写无服务器应用程序时Amazon SAM，可以构造一个Amazon SAM模板来声明和配置应用程序的组件。

本节包含有关验证Amazon SAM模板和使用依赖关系构建应用程序的主题。它还包含有关在某些用例中使用的主题，例如使用 Lambda 层、使用嵌套应用程序、控制 API Gateway API 的访问权限、使用 Step Functions 编排Amazon资源以及对应用程序进行代码签名。Amazon SAM

主题

管理访问权限

为了使您的Amazon资源相互交互，必须在您的资源之间配置适当的访问和权限，需要配置Amazon Identity and Access Management (IAM) 用户、角色和策略才能以安全的方式完成交互。要了解更多信息，请参阅《Amazon CloudFormation用户指南》中的Amazon Identity and Access Management “使用控制访问权限”。

Amazon Serverless Application Model(Amazon SAM) 提供了两个选项，可简化无服务器应用程序的访问和权限管理。

1. Amazon SAM连接器
2. Amazon SAM策略模板

Amazon SAM连接器

连接器是在两个资源之间配置权限的一种方式。你可以通过描述它们在你的Amazon SAM模板中应该如何相互交互来做到这一点。它们可以使用`Connectors`资源属性或`AWS::Serverless::Connector`资源类型进行定义。连接器支持在资源组合之间调配`Read`和`Write`访问数据和事件。Amazon要了解Amazon SAM连接器的更多信息，请参阅使用Amazon SAM连接器管理资源权限 (p. 273)。

Amazon SAM策略模板

Amazon SAM策略模板是预定义的权限集，您可以将其添加到Amazon SAM模板中，以管理Amazon Lambda函数、Amazon Step Functions状态机及其与之交互的资源之间的访问和权限。要了解Amazon SAM策略模板的更多信息，请参阅Amazon SAM策略模板 (p. 278)。

Amazon CloudFormation机制

Amazon CloudFormation机制包括配置 IAM 用户、角色和策略以管理您的Amazon资源之间的权限。要了解更多信息，请参阅 使用Amazon CloudFormation机制管理权限 (p. 320)。

最佳实践

在您的无服务器应用程序中，您可以使用多种方法在资源之间配置权限。因此，您可以为每种情况选择最佳选项，并在整个应用程序中同时使用多个选项。选择最适合您的选择时，有几件事情需要考虑：

- Amazon SAM连接器和策略模板都降低了促进Amazon资源之间安全交互所需的 IAM 专业知识。在支持时使用连接器和策略模板。
- Amazon SAM连接器提供了一种简单直观的简写语法来定义Amazon SAM模板中的权限，并且需要最少的 IAM 专业知识。当Amazon SAM连接器和策略模板都支持时，请使用Amazon SAM连接器。
- Amazon SAM连接器可以在支持的Amazon SAM源和目标资源之间预置`Read`和`Write`访问数据和事件。有关受支持资源的列表，请参阅Amazon SAM连接器参考 (p. 451)。如果支持，请使用Amazon SAM连接器。
- 虽然Amazon SAM策略模板仅限于您的 Lambda 函数、Step Functions 状态机及其与之交互的Amazon资源之间的权限，但策略模板确实支持所有 CRUD 操作。如果支持并且您的场景有Amazon SAM策略模板可用，请使用Amazon SAM策略模板。有关可用策略模板的列表，请参阅Amazon SAM策略模板 (p. 278)。
- 对于所有其他场景，或者需要精细度时，请使用Amazon CloudFormation机制。

使用Amazon SAM连接器管理资源权限

主题

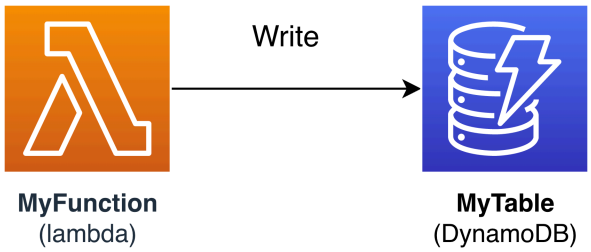
什么是Amazon SAM连接器？

连接器是一种Amazon Serverless Application Model (Amazon SAM) 抽象资源类型，标识为`AWS::Serverless::Connector`，它在您的无服务器应用程序资源之间提供简单且范围明确的权限。通过将`Connectors`资源属性嵌入到源资源中来使用该属性。然后，定义您的目标资源并描述数据或事件应如何在这些资源之间流动。 Amazon SAM然后制定必要的访问策略以促进所需的交互。

```
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>
  ...
```

连接器的示例

在此示例中，我们使用连接器将数据从Amazon Lambda函数写入 Amazon DynamoDB 表。




```
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
```

`Connectors`资源属性嵌入在 Lambda 函数源资源中。使用`Id`属性将 DynamoDB 表定义为目标资源。连接器将在这两种资源之间配置`Write`权限。

当您将Amazon SAM模板部署到时Amazon CloudFormation，Amazon SAM将自动制定此连接正常运行所需的必要访问策略。

源资源和目标资源之间支持的连接

连接器支持`Read`源和目标资源连接的特定组合之间的`Write`数据和事件权限类型。例如，`Write`连接器支持`AWS::ApiGateway::RestApi`源资源和`AWS::Lambda::Function`目标资源之间的连接。

可以使用支持的属性的组合来定义源和目标资源。属性要求将取决于您建立的连接以及资源的定义位置。

> **Note**
>
> 连接器可以在支持的无服务器和非无服务器资源类型之间配置权限。

有关支持的资源连接及其属性要求的列表，请参阅连接器支持的源和目标资源类型 (p. 451)。

使用连接器

定义读取和写入权限

Read并且可以在单个连接器中配置Write权限：

```
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
```

使用其他支持的属性定义资源

对于源资源和目标资源，在同一个模板中定义时，使用Id属性。或者，Qualifier可以添加以缩小已定义资源的范围。当资源不在同一个模板中时，请组合使用支持的属性。

- 有关源资源和目标资源支持的属性组合的列表，请参阅连接器支持的源和目标资源类型 (p. 451)。
- 有关可用于连接器的属性的描述，请参见AWS::Serverless::Connector (p. 120)。

当您使用以外的属性定义源资源时Id，请使用该SourceReference属性。

```
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>
```

以下是使用缩小Amazon API Gateway 资源范围的示例：Qualifier

```
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
  ...
```

以下是一个示例，使用支持的Arn和组合Type来定义另一个模板中的目标资源：

```
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
  ...
```

从单一来源创建多个连接器

在源资源中，您可以定义多个连接器，每个连接器具有不同的目标资源。

```
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
...
```

创建多目的地连接器

在源资源中，您可以定义具有多个目标资源的单个连接器。以下是连接到 Amazon Simple Storage Service (Amazon S3) 存储桶和 DynLambda oDB 表的 Lambddddddda 源资源的示例：

```
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
```

使用连接器定义资源属性

可以为资源定义资源属性以指定其他行为和关系。要了解有关资源属性的更多信息，请参阅Amazon CloudFormation用户指南中的资源属性参考。

您可以通过在与连接器属性相同的级别上定义资源属性来向嵌入式连接器添加资源属性。在部署时转换Amazon SAM模板时，属性将传递到生成的资源。

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Connectors:
      MyConn:
        DeletionPolicy: Retain
        DependsOn: AnotherFunction
        Properties:
          ...
```

连接器的工作原理

> **Note**
>
> 本节介绍连接器如何在后台配置必要的资源。使用连接器时，这会自动发生在你身上。

首先，将嵌入Connectors资源属性转换为AWS::Serverless::Connector资源类型。它的逻辑 ID 会自动创建为 *<source-resource-logical-id ><embedded-connector-logical-id >*。

例如，这是嵌入式连接器：

```
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
```

这将生成以下AWS::Serverless::Connector资源：

```
Transform: AWS::Serverless-2016-10-31
Resources:
  ...
  MyFunctionMyConn:
    Type: AWS::Serverless::Connector
    Properties:
      Source:
        Id: MyFunction
      Destination:
        Id: MyTable
      Permissions:
        - Read
        - Write
```

Note

您也可以使用此语法在Amazon SAM模板中定义连接器。当您的源资源是在与连接器不同的模板上定义时，建议使用此方法。

接下来，将自动制定此连接所需的访问策略。有关连接器生成的资源的更多信息，请参阅Amazon CloudFormation指定时生成的资源AWS::Serverless::Connector (p. 264)。

Amazon SAM连接器的好处

通过在资源之间自动制定相应的访问策略，Connectors 使您能够编写无服务器应用程序并专注于应用程序架构，而无需在Amazon授权功能、策略语言和特定服务安全设置方面的专业知识。因此，连接器对于可能不熟悉无服务器开发的开发人员或希望提高开发速度的经验丰富的开发人员来说是一个很大的好处。

了解更多信息

有关使用Amazon SAM连接器的更多信息，请参阅AWS::Serverless::Connector (p. 120)。

提供反馈

要提供有关连接器的反馈，请在serverless-application-model Amazon GitHub存储库中提交新问题。

Amazon SAM策略模板

Amazon Serverless Application Model(Amazon SAM) 允许您从策略模板列表中进行选择，将您的 Lambda 函数和Amazon Step Functions状态机的权限范围限定为应用程序使用的资源。

Amazon SAM使用策略模板Amazon Serverless Application Repository的应用程序不需要任何特殊的客户确认即可从中部署应用程序Amazon Serverless Application Repository。

如果要请求添加新的策略模板，请执行以下操作：

1. 针对Amazon SAM GitHub 项目`develop`分支中的 policy_templates.json 源文件提交拉取请求。你可以在 GitHub 网站上的 p olicy_templates.json 中找到源文件。
2. 在Amazon SAM GitHub 项目中提交问题，包括提交请求的原因和请求链接。使用此链接提交新问题 Amazon Serverless Application Model：问题。

语法

对于您在模板文件中指定的每个策略Amazon SAM模板，必须始终指定一个包含策略模板占位符值的对象。如果策略模板不需要任何占位符值，则必须指定一个空对象。

YAML

```
MyFunction:
  Type: AWS::Serverless::Function
  Properties:
    Policies:
      - PolicyTemplateName1:        # Policy template with placeholder value
          Key1: Value1
      - PolicyTemplateName2: {}     # Policy template with no placeholder value
```

示例

示例 1：具有占位符值的策略模板

以下示例显示 SQSPollerPolicy (p. 317) 策略模板期待 QueueName 作为资源。该Amazon SAM模板检索“MyQueue” Amazon SQS 队列的名称，您可以在同一个应用程序中创建该队列或请求将其作为应用程序的参数。

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

示例 2：不具有占位符值的策略模板

以下示例包含 CloudWatchPutMetricPolicy (p. 289) 策略模板，该模板没有占位符值。

> **Note**
>
> 即使没有占位符值，也必须指定一个空对象，否则会导致错误。

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

```
Policies:
  - CloudWatchPutMetricPolicy: {}
```

策略模板表

下表列出了可用的策略模板。

策略模板	描述		
AcmGetCertificatePolicy (p. 286)	授予从中读取证书的权限Amazon Certificate Manager。		
AMIDescribePolicy (p. 286)	授予权限以描述Amazon 系统映像 (AMI) 的权限。		
AthenaQueryPolicy (p. 286)	授予执行 Athena 查询的权限。		
AWSSecretsManagerGetSecretValuePolicy (p. 287)	授予获取指定Amazon Secrets Manager密钥的密钥值的权限。		
AWSSecretsManagerRotationPolicy (p. 287)	授予轮换机密的权限Amazon Secrets Manager。		
CloudFormationDescribeStacksPolicy (p. 288)	授予描述Amazon CloudFormation堆栈的权限。		
CloudWatchDashboardPolicy (p. 288)	授予在 CloudWatch 仪表板上运行指标的权限。		
CloudWatchDescribeAlarmHistoryPolicy (p. 288)	授予描述 CloudWatch 警报历史记录的权限。		
CloudWatchPutMetricPolicy (p. 289)	授予向发送指标的权限 CloudWatch。		
CodeCommitCrudPolicy (p. 290)	授予在特定CodeCommit存储库中创建/读取/更新/删除对象的权限。		
CodeCommitReadPolicy (p. 290)	授予读取特定 CodeCommit 存储库中对象的权限。		
CodePipelineLambdaExecutionPolicy (p. 290)	为调用的 Lambda 函数向 CodePipeline 提供报告任务状态的权限。		
CodePipelineReadOnlyPolicy (p. 291)	授予权限以获取有关 CodePipeline 管道的详细信息。		
ComprehendBasicAccessPolicy (p. 292)	允许检测实体、关键短语、语言和情绪。		
CostExplorerReadOnlyPolicy (p. 292)	为只读 Cost Explorer API 提供账单历史记录的只读权限。		
DynamoDBBackupFullAccessPolicy (p. 292)	为表的 DynamoDB 按需备份提供读取和写入权限。		
DynamoDBCrudPolicy (p. 293)	授予 Amazon DynamoDB 表的创建、读取、更新和删除权限。		
DynamoDBReadPolicy (p. 294)	授予 DynamoDB 表的只读权限。		
DynamoDBReconfigurePolicy (p. 294)	授予权限以重新配置 DynamoDB 表。		

策略模板	描述		
DynamoDBRestoreFromBackupPolicy (p. 295)	授予权限以从备份中恢复 DynamoDB 表。		
DynamoDBStreamReadPolicy (p. 296)	授予描述和读取 DynamoDB 流和记录的权限。		
DynamoDBWritePolicy (p. 296)	授予 DynamoDB 表的只写权限。		
EC2CopyImagePolicy (p. 297)	授予复制 Amazon EC2 映像的权限。		
EC2DescribePolicy (p. 297)	授予权限以描述亚马逊Elastic Compute Cloud (Amazon EC2) 实例。		
EcsRunTaskPolicy (p. 298)	授予权限以启动任务定义。		
EFSWriteAccessPolicy (p. 298)	授予权限以挂载具有写入 Amazon EFS 文件系统的权限。		
EKSDescribePolicy (p. 299)	授予描述或列出 Amazon EKS 集群的权限。		
ElasticMapReduceAddStepPolicy (p. 299)	授予权限以将新步骤添加到运行的集群中。		
ElasticMapReduceCancelStepsPolicy (p. 299)	授予权限以取消运行的集群中的一个或多个待处理步骤。		
ElasticMapReduceModifyInstanceFleetPolicy (p. 299)	授予列出集群中实例队列的详细信息和修改容量的权限。		
ElasticMapReduceModifyInstanceGroupsPolicy (p. 300)	授予列出集群中实例组的详细信息和修改设置的权限。		
ElasticMapReduceSetTerminationProtectionPolicy (p. 300)	授予权限以为集群设置终止保护。		
ElasticMapReduceTerminateJobFlowsPolicy (p. 301)	授予权限以关闭集群的权限。		
ElasticsearchHttpPostPolicy (p. 301)	向亚马逊 OpenSearch 服务授予 POST 权限。		
EventBridgePutEventsPolicy (p. 302)	授予向发送事件的权限 EventBridge。		
FilterLogEventsPolicy (p. 302)	授予筛选来自指定 CloudWatch 日志组的日志事件的权限。		
FirehoseCrudPolicy (p. 303)	授予权限以创建、写入、更新和删除 Kinesis Data Firehose 传输流。		
FirehoseWritePolicy (p. 303)	授予权限以写入 Kinesis Data Firehose 传输流。		
KinesisCrudPolicy (p. 304)	授予创建、发布和删除Amazon Kinesis 直播的权限。		
KinesisStreamReadPolicy (p. 304)	授予权限以列出和阅读Amazon Kinesis Kinesis 流。		
KMSDecryptPolicy (p. 305)	授予使用Amazon Key Management Service (Amazon KMS) 密钥解密的权限。		
KMSEncryptPolicy (p. 305)	授予使用Amazon Key Management Service (Amazon KMS) 密钥加密的权限。		

策略模板	描述		
LambdaInvokePolicy (p. 310)	授予调用Amazon Lambda函数、别名或版本的权限。		
MobileAnalyticsWriteOnlyAccessPolicy (p. 311)	授予只写权限，以放置所有应用程序资源的事件数据。		
OrganizationsListAccountsPolicy (p. 312)	授予列出子账户名和 ID 的只读权限。		
PinpointEndpointAccessPolicy (p. 312)	授予权限以获取和更新 Amazon Pinpinpoint 应用程序的终端节点。		
PollyFullAccessPolicy (p. 313)	授予对Amazon Polly 词典资源的完全访问权限。		
RekognitionDetectOnlyPolicy (p. 313)	授予检测人脸、标签和文字的权限。		
RekognitionFacesManagementPolicy (p. 314)	授予在Amazon Rekognition 集合中添加、删除和搜索人脸的权限。		
RekognitionFacesPolicy (p. 314)	授予比较和检测人脸和标签的权限。		
RekognitionLabelsPolicy (p. 315)	授予检测对象和审核标签的权限。		
RekognitionNoDataAccessPolicy (p. 315)	授予比较和检测人脸和标签的权限。		
RekognitionReadPolicy (p. 316)	授予列出和搜索面孔的权限。		
RekognitionWriteOnlyAccessPolicy (p. 316)	授予创建集合和索引人脸的权限。		
Route53ChangeResourceRecordSetsPolicy (p. 317)	授予更改Route 53中的资源记录集的权限。		
S3CrudPolicy (p. 317)	授予创建、读取、更新和删除权限，以对Amazon S3 存储桶中的对象进行操作。		
S3FullAccessPolicy (p. 318)	授予对Amazon S3 存储桶中的对象进行操作的完全访问权限。		
S3ReadPolicy (p. 319)	授予只读Amazon Simp存储服务 (Amazon S3) 存储桶中的对象的只读权限。		
S3WritePolicy (p. 319)	授予写入 Amazon S3 存储桶的权限。		
SageMakerCreateEndpointConfigPolicy (p. 320)	授予权限以在中创建终端节点配置SageMaker。		
SageMakerCreateEndpointPolicy (p. 321)	授予在中创建端点的权限 SageMaker。		
ServerlessRepoReadWriteAccessPolicy (p. 321)	授予在Amazon Serverless Application Repository服务中创建和列出应用程序的权限。		
SESBulkTemplatedCrudPolicy (p. 314)	授予发送电子邮件、模板化电子邮件、模板化批量电子邮件和验证身份的权限。		
SESBulkTemplatedCrudPolicy_v2 (p. 314)	授予发送Amazon SES 电子邮件、模板化电子邮件和模板化批量电子邮件以及验证身份的权限。		
SESCrudPolicy (p. 315)	授予发送电子邮件和验证身份的权限。		

策略模板	描述		
SESEmailTemplateCrudPolicy (p. 315)	授予创建、获取、列出、更新和删除 Amazon SES 电子邮件模板的权限。		
SESSendBounceePolicy (p. 316)	SendBounce 授予权限以获取Amazon SSimple Email Service (Amazon SES) 身份。		
SNSCrudPolicy (p. 317)	授予创建、发布和订阅 Amazon SNS 主题的权限。		
SNSPublishMessagePolicy (p. 318)	授予权限以将消息发布到 Amazon SimpSimple Notification Service (Amazon SNS) 主题。		
SQSPollerPolicy (p. 319)	授予权限以轮询AAmazon Simple Queue Service (Amazon SQS) 队列。		
SQSSendMessagePolicy (p. 321)	授予权限以将消息发送到 Amazon SQS 队列。		
SSMParameterReadPolicy (p. 323)	授予访问来自 Amazon EC2 Systems Manager (SSM) 参数存储的参数以在此账户中加载密钥的权限。在参数名称没有斜杠前缀时使用。		
SSMParameterWithSlashPrefixReadPolicy (p. 326)	授予访问来自 Amazon EC2 Systems Manager (SSM) 参数存储的参数以在此账户中加载密钥的权限。在参数名称有斜杠前缀时使用。		
StepFunctionsExecutionPolicy (p. 327)	授予权限以开始执行 Step Functions 状态机。		
TextractDetectAnalyzePolicy (p. 329)	允许使用工马逊 Textract 检测和分析文档。		
TextractGetResultPolicy (p. 330)	允许从工马逊 Textract 获取检测和分析文档。		
TextractPolicy (p. 332)	提供对Amazon Textract的完全访问权限		
VPCAccessPolicy (p. 334)	提供创建、删除、描述和分离弹性网络接口的权限。		

问题排查

SAM CLI 错误：“必须为策略模板'<policy-template-name >'指定有效的参数值”

执行 `sam build` 时，您会看到以下错误：

```
"Must specify valid parameter values for policy template '<policy-template-name>'"
```

这意味着您在声明没有任何占位符值的策略模板时没有传递空对象。

要解决此问题，请像以下示例一样声明策略CloudWatchPutMetricPolicy (p. 289)。

```
MyFunction:
  Policies:
    - CloudWatchPutMetricPolicy: {}
```

策略模板列表

以下是可用的策略模板以及应用于每个模板的权限。 Amazon Serverless Application Model(Amazon SAM)自动在占位符项目（例如Amazon区域和账户 ID）中填充相应的信息。

主题

AcmGetCertificatePolicy

授予从中读取证书的权限Amazon Certificate Manager。

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

```
    "Effect": "Allow",
    "Action": [
      "acm:GetCertificate"
    ],
    "Resource": {
      "Fn::Sub": [
        "${certificateArn}",
        {
          "certificateArn": {
            "Ref": "CertificateArn"
          }
        }
      ]
    }
  }
]
```

AMIDescribePolicy

授予权限以描述Amazon 系统映像 (AMI)。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ec2:DescribeImages"
    ],
    "Resource": "*"
  }
]
```

AthenaQueryPolicy

授予执行 Athena 查询的权限。

```
"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

授予获取指定Amazon Secrets Manager密钥的密钥值的权限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "secretsmanager:GetSecretValue"
    ],
    "Resource": {
      "Fn::Sub": [
        "${secretArn}",
        {
          "secretArn": {
            "Ref": "SecretArn"
          }
        }
      ]
    }
  }
]
```

AWSSecretsManagerRotationPolicy

授予轮换机密的权限Amazon 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

授予描述Amazon CloudFormation堆栈的权限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "cloudformation:DescribeStacks"
    ],
    "Resource": {
      "Fn::Sub": "arn:${AWS::Partition}:cloudformation:${AWS::Region}:
${AWS::AccountId}:stack/*"
    }
  }
]
```

CloudWatchDashboardPolicy

授予在 CloudWatch 仪表板上运行指标的权限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "cloudwatch:GetDashboard",
      "cloudwatch:ListDashboards",
      "cloudwatch:PutDashboard",
      "cloudwatch:ListMetrics"
    ],
    "Resource": "*"
  }
]
```

CloudWatchDescribeAlarmHistoryPolicy

授予描述亚马逊 CloudWatch 警报历史记录的权限。

```
"Statement": [
```

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

CloudWatchPutMetricPolicy

授予向发送指标的权限 CloudWatch。

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

CodePipelineLambdaExecutionPolicy

为调用的 Lambda 函数Amazon CodePipeline提供报告任务状态的权限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "codepipeline:PutJobSuccessResult",
      "codepipeline:PutJobFailureResult"
    ],
    "Resource": "*"
  }
]
```

CodePipelineReadOnlyPolicy

授予权限以获取有关 CodePipeline 管道的详细信息。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "codepipeline:ListPipelineExecutions"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:codepipeline:${AWS::Region}:${AWS::AccountId}:
${pipelinename}",
        {
          "pipelinename": {
            "Ref": "PipelineName"
          }
        }
      ]
    }
  }
```

```
]
```

CodeCommitCrudPolicy

授予权限以在特定 CodeCommit存储库中创建、读取、更新和删除对象。

```
"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

授予读取特定 CodeCommit 存储库中对象的权限。

```
"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

允许检测实体、关键短语、语言和情绪。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "comprehend:BatchDetectKeyPhrases",
      "comprehend:DetectDominantLanguage",
      "comprehend:DetectEntities",
      "comprehend:BatchDetectEntities",
      "comprehend:DetectKeyPhrases",
      "comprehend:DetectSentiment",
      "comprehend:BatchDetectDominantLanguage",
      "comprehend:BatchDetectSentiment"
    ],
    "Resource": "*"
  }
]
```

CostExplorerReadOnlyPolicy

为只读Amazon Cost Explorer（Cost Explorer）API 提供账单历史记录的只读权限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ce:GetCostAndUsage",
      "ce:GetDimensionValues",
      "ce:GetReservationCoverage",
      "ce:GetReservationPurchaseRecommendation",
      "ce:GetReservationUtilization",
      "ce:GetTags"
    ],
    "Resource": "*"
  }
]
```

DynamoDBBackupFullAccessPolicy

为表的 DynamoDB 按需备份提供读取和写入权限。

```
"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

授予 Amazon DynamoDB 表的创建、读取、更新和删除权限。

```
"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

对 DynamoDB 表授予只读权限。

```
"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

授予权限以重新配置 DynamoDB 表。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "dynamodb:UpdateTable"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
${tableName}",
        {
          "tableName": {
            "Ref": "TableName"
          }
        }
      ]
    }
  }
]
```

DynamoDBRestoreFromBackupPolicy

授予权限以从备份中还原 DynamoDB 表。

```
"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

授予描述和读取 DynamoDB 流和记录的权限。

```
"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

对 DynamoDB 表授予只写权限。

```
"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

授予复制Amazon EC2 映像的权限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ec2:CopyImage"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:ec2:${AWS::Region}:${AWS::AccountId}:image/${imageId}",
        {
          "imageId": {
            "Ref": "ImageId"
          }
        }
      ]
    }
  }
]
```

EC2DescribePolicy

授予权限以描述Amazon Elastic Compute Cloud (Amazon EC2) 实例。

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

```
]
```

EcsRunTaskPolicy

授予权限以启动任务定义的新任务。

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

EFSWriteAccessPolicy

授予权限以挂载具有写入权限的 Amazon EFS 文件系统。

```
"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

授予权限以描述或列出 Amazon Elastic Kubernetes Service (Amazon ES) 集群

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "eks:DescribeCluster",
      "eks:ListClusters"
    ],
    "Resource": "*"
  }
]
```

ElasticMapReduceAddJobFlowSteps政策

授予权限以将新步骤添加到运行的集群中。

```
"Statement": [
  {
    "Action": "elasticmapreduce:AddJobFlowSteps",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:${AWS::AccountId}:cluster/
${clusterId}",
        {
          "clusterId": {
            "Ref": "ClusterId"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

ElasticMapReduceCancelStepsPolicy

授予权限以取消运行的集群中的一个或多个待处理步骤。

```
"Statement": [
  {
    "Action": "elasticmapreduce:CancelSteps",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:${AWS::AccountId}:cluster/
${clusterId}",
        {
          "clusterId": {
            "Ref": "ClusterId"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
```

```
]
```

ElasticMapReduceModifyInstanceFleetPolicy

授予列出集群内实例队列的详细信息和修改容量的权限。

```
"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

授予列出集群内实例组的详细信息和修改设置的权限。

```
"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

授予权限以为集群设置终止保护。

```
"Statement": [
  {
    "Action": "elasticmapreduce:SetTerminationProtection",
    "Resource": {
      "Fn::Sub": [
```

```
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:${AWS::AccountId}:cluster/
${clusterId}",
        {
          "clusterId": {
            "Ref": "ClusterId"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

ElasticMapReduceTerminateJobFlowsPolicy

授予权限以关闭集群的权限。

```
"Statement": [
  {
    "Action": "elasticmapreduce:TerminateJobFlows",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:${AWS::AccountId}:cluster/
${clusterId}",
        {
          "clusterId": {
            "Ref": "ClusterId"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

ElasticsearchHttpPostPolicy

向亚马逊 OpenSearch 服务授予 POST 和 PUT 权限。

```
"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

授予向亚马逊发送事件的权限 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

授予筛选来自指定 CloudWatch 日志组的日志事件的权限。

```
"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

授予权限以创建、写入、更新和删除 Kinesis Data Firehose 传输流。

```
"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

授予权限以写入 Kinesis Data Firehose 传输流。

```
"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

授予创建、发布和删除Amazon Kinesis 直播的权限。

```
"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

授予权限以列出和阅读 Amazon Kinesis 流。

```
"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

授予使用Amazon Key Management Service (Amazon KMS) 密钥解密的权限。请注意，它keyId必须是Amazon KMS密钥 ID，而不是密钥别名。

```
"Statement": [
  {
    "Action": "kms:Decrypt",
    "Effect": "Allow",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:kms:${AWS::Region}:${AWS::AccountId}:key/${keyId}",
        {
          "keyId": {
            "Ref": "KeyId"
          }
```

```
        }
      ]
    }
  }
]
```

KMSEncryptPolicy

授予使用密Amazon KMS钥加密的权限。请注意，keyId 必须是Amazon KMS密钥 ID，而不是密钥别名。

```
"Statement": [
  {
    "Action": "kms:Encrypt",
    "Effect": "Allow",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:kms:${AWS::Region}:${AWS::AccountId}:key/${keyId}",
        {
          "keyId": {
            "Ref": "KeyId"
          }
        }
      ]
    }
  }
]
```

LambdaInvokePolicy

授予调用Amazon Lambda函数、别名或版本的权限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "lambda:InvokeFunction"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:lambda:${AWS::Region}:${AWS::AccountId}:function:
${functionName}*",
        {
          "functionName": {
            "Ref": "FunctionName"
          }
        }
      ]
    }
  }
]
```

MobileAnalyticsWriteOnlyAccessPolicy

授予只写权限，以放置所有应用程序资源的事件数据。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "mobileanalytics:PutEvents"
    ],
```

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

OrganizationsListAccountsPolicy

授予列出子账户名和 ID 的只读权限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "organizations:ListAccounts"
    ],
    "Resource": "*"
  }
]
```

PinpointEndpointAccessPolicy

授予权限以获取和更新 Amazon Pinpoint 应用程序的终端节点。

```
"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

授予对Amazon Polly 词典资源的完全访问权限。

```
"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

授予检测人脸、标签和文字的权限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "rekognition:DetectFaces",
      "rekognition:DetectLabels",
      "rekognition:DetectModerationLabels",
      "rekognition:DetectText"
    ],
    "Resource": "*"
  }
]
```

RekognitionFacesManagementPolicy

授予在Amazon Rekognition 集合中添加、删除和搜索人脸的权限。

```
"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

授予比较和检测人脸和标签的权限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "rekognition:CompareFaces",
      "rekognition:DetectFaces"
    ],
    "Resource": "*"
  }
]
```

RekognitionLabelsPolicy

授予检测对象和审核标签的权限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "rekognition:DetectLabels",
      "rekognition:DetectModerationLabels"
    ],
    "Resource": "*"
  }
]
```

RekognitionNoDataAccessPolicy

授予比较和检测人脸和标签的权限。

```
"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

授予列出和搜索面孔的权限。

```
"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

授予创建集合和索引人脸的权限。

```
"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

授予更改 Route 53 中的资源记录集的权限。

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

```
    "Effect": "Allow",
    "Action": [
      "route53:ChangeResourceRecordSets"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:route53:::hostedzone/${HostedZoneId}",
        {
          "HostedZoneId": {
            "Ref": "HostedZoneId"
          }
        }
      ]
    }
  }
]
```

S3CrudPolicy

授予创建、读取、更新和删除权限，以对 Amazon S3 存储桶中的对象进行操作。

```
"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

授予对 Amazon S3 存储桶中的对象进行操作的完全访问权限。

```
"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

授予权限以读取 Amazon Simple Storage Service (Amazon S3) 存储桶中的对象

```
"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

授予权限以将对象写入 Amazon S3 存储桶。

```
"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

授予权限以在中创建终端节点配置 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

授予在中创建端点的权限 SageMaker。

```
"Statement": [
  {
    "Action": [
      "sagemaker:CreateEndpoint"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:sagemaker:${AWS::Region}:${AWS::AccountId}:endpoint/
${endpointName}",
        {
          "endpointName": {
            "Ref": "EndpointName"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

ServerlessRepoReadWriteAccessPolicy

授予在Amazon Serverless Application Repository (Amazon SAM) 服务中创建和列出应用程序的权限。

```
"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

授予发送 Amazon SES 电子邮件、模板化电子邮件和模板化批量电子邮件以及验证身份的权限。

Note

该ses:SendTemplatedEmail操作需要模板 ARN。请改用SESBulkTemplatedCrudPolicy_v2。

```
"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

授予发送 Amazon SES 电子邮件、模板化电子邮件和模板化批量电子邮件以及验证身份的权限。

```
"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

授予发送电子邮件和验证身份的权限。

```
"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

授予创建、获取、列出、更新和删除 Amazon SES 电子邮件模板的权限。

```
"Statement": [
```

```
  {
    "Effect": "Allow",
    "Action": [
      "ses:CreateTemplate",
      "ses:GetTemplate",
      "ses:ListTemplates",
      "ses:UpdateTemplate",
      "ses:DeleteTemplate",
      "ses:TestRenderTemplate"
    ],
    "Resource": "*"
  }
]
```

SESSendBouncePolicy

SendBounce 授予权限以使用Amazon SSimple Email Service (Amazon SES)

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ses:SendBounce"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:ses:${AWS::Region}:${AWS::AccountId}:identity/
${identityName}",
        {
          "identityName": {
            "Ref": "IdentityName"
          }
        }
      ]
    }
  }
]
```

SNSCrudPolicy

授予创建、发布和订阅 Amazon SNS 主题的权限。

```
"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

授予权限以将消息发布到 Amazon SimpSimple Notification Service (Amazon SNS) 主题。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "sns:Publish"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:sns:${AWS::Region}:${AWS::AccountId}:${topicName}",
        {
          "topicName": {
            "Ref": "TopicName"
          }
        }
      ]
    }
  }
]
```

SQSPollerPolicy

授予权限以轮询亚马逊简单队列Service (Amazon QS) 队列。

```
"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

授予权限以将消息发送到 Amazon SQS 队列。

```
"Statement": [
  {
    "Effect": "Allow",
```

```
    "Action": [
      "sqs:SendMessage*"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:sqs:${AWS::Region}:${AWS::AccountId}:${queueName}",
        {
          "queueName": {
            "Ref": "QueueName"
          }
        }
      ]
    }
  }
]
```

SSMParameterReadPolicy

授予访问来自 Amazon EC2 Systems Manager (SSM) 参数存储的参数以在此账户中加载密钥的权限。在参数名称没有斜杠前缀时使用。

> **Note**
>
> 如果您不使用默认密钥，则还需要该KMSDecryptPolicy策略。

```
"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

授予访问来自 Amazon EC2 Systems Manager (SSM) 参数存储的参数以在此账户中加载密钥的权限。在参数名有斜杠前缀时使用。

> **Note**
>
> 如果您不使用默认密钥，则还需要该KMSDecryptPolicy策略。

```
"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

授予权限以开始执行 Step Functions 状态机。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "states:StartExecution"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:states:${AWS::Region}:${AWS::AccountId}:stateMachine:
${stateMachineName}",
        {
          "stateMachineName": {
            "Ref": "StateMachineName"
          }
        }
      ]
    }
  }
]
```

TextractDetectAnalyzePolicy

允许使用亚马逊 Textract 检测和分析文档。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "textract:DetectDocumentText",
```

```
      "textract:StartDocumentTextDetection",
      "textract:StartDocumentAnalysis",
      "textract:AnalyzeDocument"
    ],
    "Resource": "*"
  }
]
```

TextractGetResultPolicy

允许从亚马逊 Textract 获取检测和分析文档。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "textract:GetDocumentTextDetection",
      "textract:GetDocumentAnalysis"
    ],
    "Resource": "*"
  }
]
```

TextractPolicy

授予 Amazon Textract 的完全访问权限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "textract:*"
    ],
    "Resource": "*"
  }
]
```

VPCAccessPolicy

提供创建、删除、描述和分离弹性网络接口的权限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ec2:CreateNetworkInterface",
      "ec2:DeleteNetworkInterface",
      "ec2:DescribeNetworkInterfaces",
      "ec2:DetachNetworkInterface"
    ],
    "Resource": "*"
  }
]
```

使用Amazon CloudFormation机制管理权限

要控制对Amazon资源的访问权限，Amazon Serverless Application Model(Amazon SAM) 可以使用与相同的机制Amazon CloudFormation。有关更多信息，请参阅《Amazon CloudFormation用户指南》Amazon Identity and Access Management中的控制访问权限。

授予用户管理无服务器应用程序的权限有三个主要选项。每个选项都为用户提供不同级别的访问控制。

- 授予管理员权限。
- 附加必要的Amazon托管策略。
- 授予特定Amazon Identity and Access Management (IAM) 权限。

根据您选择的选项，用户只能管理包含他们有权访问的Amazon资源的无服务器应用程序。

以下部分对每个选项进行了详述。

授予管理员权限

如果您向用户授予管理员权限，则他们可以管理包含任意Amazon资源组合的无服务器应用程序。这是最简单的选项，但它也为用户授予了最广泛的权限，因此他们能够执行影响最大的操作。

有关向用户授加管理员权限的更多信息，请参阅 IAM 用户指南中的创建您的第一个 IAM 管理员用户和组。

附上必要的Amazon托管策略

您可以使用Amazon托管策略向用户授予部分权限，而不是授予完全的管理员权限。如果使用此选项，请确保Amazon托管策略集涵盖用户管理的无服务器应用程序所需的所有操作和资源。

例如，以下Amazon托管策略足以部署示例 Hello World 应用程序 (p. 23)：

- AWSCloudFormationFullAccess
- IAMFullAccess
- AWSLambda_FullAccess
- 亚马逊 APIGatewayAdministrator
- 亚马逊 S3FullAccess
- AmazoneC2ContainerRegistryFullAccess

有关向 IAM 用户附加策略的信息，请参阅 I AM 用户指南中的更改 IAM 用户的权限。

授予特定的 IAM 权限

要实现最精细的访问控制级别，您可以使用策略语句向用户授予特定的 IAM 权限。如果使用此选项，请确保策略声明包含用户管理的无服务器应用程序所需的所有操作和资源。

使用此选项的最佳做法是拒绝用户创建角色（包括 Lambda 执行角色）的权限，这样他们就无法向自己授予升级权限。因此，作为管理员，您必须首先创建一个 Lambda 执行角色，该角色将在用户要管理的无服务器应用程序中指定。有关创建 Lambda 执行角色的信息，请参阅在 IAM 控制台中创建执行角色。

对于示例 Hello World 应用程序 (p. 23)，足以运行该应用程序。AWSLambdaBasicExecutionRole创建 Lambda 执行角色后，修改示例 Hello World 应用程序的Amazon SAM模板文件，为`AWS::Serverless::Function`资源添加以下属性：

```
  Role: lambda-execution-role-arn
```

修改后的 Hello World 应用程序到位后，以下策略声明为用户授予足够的权限来部署、更新和删除应用程序：

```
{
```

```
"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

本节中的示例策略语句授予您足够的权限来部署、更新和删除示例 Hello World 应用程序 (p. 23)。如果您向应用程序添加其他资源类型，则需要更新政策声明以包括以下内容：

1. 允许您的应用程序调用服务操作。
2. 服务主体，如果服务操作需要的话。

例如，如果您添加 Step Functions 工作流，则可能需要为此处列出的操作添加权限和`states.amazonaws.com`服务主体。

有关 IAM 策略的更多信息，请参阅 IAM 用户指南中的管理 IAM 策略。

使用调度器 EventBridge调度事件

什么是亚马逊 EventBridge 日程安排器？

Amazon S EventBridge cheduler 是一项计划服务，允许您在所有Amazon服务中创建、启动和管理数千万个事件和任务。要了解有关亚马逊 EventBridge 日程安排器的更多信息，请参阅什么是亚马逊 EventBridge 日程安排器？ 在《EventBridge 调度程序用户指南》中。

主题

- EventBridge 中的调度器支持Amazon SAM (p. 324)
- 在中创建 EventBridge 调度器事件Amazon SAM (p. 324)
- 示例 (p. 325)
- 了解更多信息 (p. 326)

EventBridge 中的调度器支持Amazon SAM

Amazon Serverless Application Model(Amazon SAM) 模板规范提供了一种简单、简短的语法，您可以使用该语法通过 S EventBridge cheduler 为Amazon Lambda和调度事件Amazon Step Functions。

在中创建 EventBridge 调度器事件Amazon SAM

在Amazon SAM模板中将该`ScheduleV2`属性设置为事件类型以定义 S EventBridge cheduler 事件。此属性支持`AWS::Serverless::Function`和`AWS::Serverless::StateMachine`资源类型。

```
MyFunction:
  Type: AWS::Serverless::Function
  Properties:
    Events:
      CWSchedule:
        Type: ScheduleV2
```

```
        Properties:
          Schedule: 'rate(1 minute)'
          Name: TestScheduleV2Function
          Description: Test schedule event

MyStateMachine:
  Type: AWS::Serverless::StateMachine
  Properties:
    Events:
      CWSchedule:
        Type: ScheduleV2
        Properties:
          Schedule: 'rate(1 minute)'
          Name: TestScheduleV2StateMachine
          Description: Test schedule event
```

EventBridge 调度器事件调度还支持未处理事件的死信队列 (DLQ)。有关死信队列的更多信息，请参阅《调度程序用户指南》中的为 EventBridge 调度器配置死信队列。EventBridge

指定 DLQ ARN 后，为调度器计划Amazon SAM配置权限以向 DLQ 发送消息。如果未指定 DLQ ARN，则Amazon SAM将创建 DLQ 资源。

示例

使用以下方式定义 EventBridge 调度器事件的基本示例Amazon SAM

```
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"}'
```

了解更多信息

要了解有关定义 SScheduleV2 EventBridge cheduler 属性的更多信息，请参阅：

- ScheduleV2 (p. 195)为AWS::Serverless::Function了。
- ScheduleV2 (p. 256)为AWS::Serverless::StateMachine了。

正在验证Amazon SAM模板文件

使用验证模板sam validate (p. 440). 目前，此命令验证提供的模板是否有效的 JSON/YAML。和大多数人一样Amazon SAMCLI 命令，它查找template.[yaml|yml]默认情况下文件在您当前的工作目录中。您可以使用指定不同的模板文件-t要么--template选项。

例如：

```
sam validate
<path-to-file>/template.yml is a valid SAM Template
```

> **Note**
>
> 这些区域有：sam validate命令需要Amazon要配置的凭据。有关更多信息，请参阅配置和凭证文件。

使用图层

使用Amazon SAM，您可以在无服务器应用程序中包含图层。有关层的更多信息，请参阅Amazon Lambda层中的Amazon Lambda开发人员指南.

本主题提供有关以下内容的信息：

- 在应用程序中包含图层
- 图层在本地缓存方式

有关构建自定义图层的信息，请参阅建筑层 (p. 354).

在应用程序中包含图层

要在您的应用程序中包含层，请使用Layers的财产AWS::Serverless::Function (p. 129)资源类型。

以下是一个示例Amazon SAM带有包含图层的 Lambda 函数的模板：

```
ServerlessFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: .
    Handler: my_handler
    Runtime: Python3.7
    Layers:
        - <LayerVersion ARN>
```

图层在本地缓存方式

使用sam local命令时，函数的图层包将被下载并缓存在本地主机上。

下表显示了不同操作系统的默认缓存目录位置。

OS	位置
Windows 7	C:\Users\<user>\AppData\Roaming\AWS SAM
Windows 8	C:\Users\<user>\AppData\Roaming\AWS SAM
Windows 10	C:\Users\<user>\AppData\Roaming\AWS SAM
macOS	~/.aws-sam/layers-pkg
Unix	~/.aws-sam/layers-pkg

缓存软件包后，Amazon SAMCLI 将图层叠加到用于调用函数的 Docker 映像上。这些区域有：Amazon SAMCLI 生成它构建的映像的名称以及缓存中保存的 LayerVersion。您可以在以下章节中找到有关该架构的更多详细信息。

要检查叠加的图层，请执行以下命令以在要检查的图像中启动 bash 会话：

```
docker run -it --entrypoint=/bin/bash samcli/lambda:<Tag following the schema outlined in
 Docker Image Tag Schema> -i
```

层缓存目录名称架构

鉴于模板中定义的 layerVersionARN，Amazon SAMCLI 从 ARN 中提取 LayerName 和版本。它会创建一个目录以将图层内容放置在名为LayerName-Version-<first 10 characters of sha256 of ARN>.

例如：

```
ARN = arn:aws:lambda:us-west-2:111111111111:layer:myLayer:1
Directory name = myLayer-1-926eeb5ff1
```

Docker 映像标签架构

要计算唯一的图层哈希值，请将所有唯一图层名称与 '-' 的分隔符组合起来，取 SHA256 哈希值，然后取前 10 个字符。

例如：

```
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
```

唯一名称的计算方式与图层缓存目录名称架构相同：

```
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
```

要计算唯一的图层哈希值，请将所有唯一图层名称与 '-' 的分隔符组合起来，取 sha256 哈希，然后取前 25 个字符：

```
myLayer-1-926eeb5ff1-mySecondLayer-1-6bc1022bdf = 2dd7ac5ffb30d515926aef
```

然后将此值与函数的运行时间和架构结合起来，并使用 '-' 的分隔符：

```
python3.7-x86_64-2dd7ac5ffb30d515926aefffd
```

使用嵌套应用

无服务器应用程序可能包含一个或多个嵌套应用. 您可以将嵌套应用程序部署为独立工件或作为较大应用程序的组件进行部署。

随着无服务器架构的增长，常见的模式会出现在多个应用程序模板中定义相同的组件。现在，您可以将常见模式分开为专用应用程序，然后将它们嵌套为新的或现有的应用程序模板的一部分。使用嵌套应用程序，您可以更加专注于应用程序独有的业务逻辑。

要在无服务器应用程序中定义嵌套应用程序，请使用AWS::Serverless::Application (p. 116)资源类型。

您可以从以下两个来源定义嵌套应用程序：

- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的Amazon Serverless Application Repository应用程序— 您可以通过使用可供您的帐户使用的应用程序来定义嵌套应用程序Amazon Serverless Application Repository. 这些可能是私人的您账户中的应用程序，私下共享使用你的帐户或应用程序公开共享中的Amazon Serverless Application Repository. 有关不同部署权限级别的更多信息，请参阅应用程序部署权限和发布应用程序中的Amazon Serverless Application Repository开发人员指南.
- 一个本地应用— 您可以使用存储在本地文件系统上的应用程序来定义嵌套应用程序。

有关如何使用的详细信息，请参阅下面几节。Amazon SAM在无服务器应用程序中定义这两种类型的嵌套应用程序。

> **Note**
>
> 可以嵌套在无服务器应用程序中的最大数量为 200 个。
> 嵌套应用程序可以拥有的最大参数数量为 60 个。

从中定义嵌套应用程序Amazon Serverless Application Repository

您可以使用在Amazon Serverless Application Repository. 您还可以使用Amazon Serverless Application Repository. 要查看嵌套应用程序的详细信息Amazon Serverless Application Repository，您可以使用Amazon开发工具包，Amazon CLI或者 Lambda 控制台。

定义托管在Amazon Serverless Application Repository在您的无服务器应用程序中Amazon SAM模板，请使用复制为 SAM 资源每个详情页面上的按钮Amazon Serverless Application Repository应用程序. 为此，请按照以下步骤操作：

1. 请确保您已登录到Amazon Web Services Management Console.
2. 找到要嵌套到Amazon Serverless Application Repository通过使用中的步骤浏览、搜索和部署应用程序的 部分Amazon Serverless Application Repository开发人员指南.
3. 选择复制为 SAM 资源按钮。您正在查看的应用程序的 SAM 模板部分现在位于剪贴板中。
4. 将 SAM 模板部分粘贴到`Resources:`要嵌套在此应用程序中的应用程序的 SAM 模板文件的部分。

以下是托管在中的嵌套应用程序的示例 SAM 模板部分Amazon 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
```

如果没有必要的参数设置，则可以省略`Parameters:`模板的部分。

> **Important**
>
> 包含托管在Amazon Serverless Application Repository继承嵌套应用程序的共享限制。例如，假设一个应用程序是公开共享的，但它包含一个只与私下共享的嵌套应用程序Amazon创建父应用程序的账户。在这种情况下，如果你Amazon帐户没有部署嵌套应用程序的权限，您无法部署父应用程序。有关部署应用程序的权限的更多信息，请参阅应用程序部署权限和发布应用程序中的Amazon Serverless Application Repository开发人员指南.

从本地文件系统中定义嵌套应用程序

您可以使用存储在本地文件系统上的应用程序来定义嵌套应用程序。您可以通过指定路径来指定路径。Amazon SAM存储在您的本地文件系统上的模板文件。

以下是嵌套本地应用程序的 SAM 模板部分示例：

```
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
```

如果没有参数设置，则可以省略`Parameters:`模板的部分。

部署嵌套应用

您可以使用Amazon SAMCLI 命令`sam deploy`. 有关更多信息，请参阅 部署无服务器应用程序 (p. 366)。

> Note
>
> 在部署包含嵌套应用程序的应用程序时，必须确认这一点。您可以通过将 CAPABILITY_AUTO_EXPAND 传递给CreateCloudFormationChangeSet API，或者使用`aws serverlessrepo create-cloud-formation-change-set` Amazon CLI命令。有关确认嵌套应用程序的更多信息，请参在部署应用程序时确认 IAM 角色、资源策略和嵌套应用程序中的Amazon Serverless Application Repository开发人员指南.

控制对 API Gateway API 的访问

要控制谁可以访问您的 Amazon API Gateway，您可以在Amazon SAMTemplate

Amazon SAM支持多种用于控制对 API Gateway API 的访问的机制。受支持的机制集不同`AWS::Serverless::HttpApi`和`AWS::Serverless::Api`资源类型。

下表汇总了每种资源类型支持的机制。

控制访问的机制	AWS::Serverless::HttpApi	AWS::Serverless::Api
Lambda 授权方	✓	✓
IAM 权限		✓
Amazon Cognito 用户池	✓ *	✓
API 密钥		✓
资源策略		✓
OAuth 2.0/JWT 授权方	✓	

* 您可以使用 Amazon Cognito 作为 JSON Web Token (JWT) 颁发者`AWS::Serverless::HttpApi`资源类型。

- Lambda 授权方— Lambda 授权者（以前称为自定义授权方）是您为控制对 API 的访问而提供的 Lambda 函数。调用 API 时，将使用客户端应用程序提供的请求上下文或授权令牌调用此 Lambda 函数。Lambda 函数会响应调用者是否有权执行请求的操作。

  无论是`AWS::Serverless::HttpApi`和`AWS::Serverless::Api`资源类型支持 Lambda 授权方。

有关 Lambda 授权方`AWS::Serverless::HttpApi`，请参阅使用Amazon LambdaHTTP API 的授权方中的API Gateway 开发人员指南. 有关 Lambda 授权方`AWS::Serverless::Api`，请参阅使用 API Gateway Lambda 授权方中的API Gateway 开发人员指南.

有关两种资源类型的 Lambda 授权者的示例，请参阅Lambda 授权方 (p. 332).

- IAM 权限— 您可以使用下列方式控制谁可以调用 API。Amazon Identity and Access Management(IAM) 权限. 调用 API 的用户必须使用 IAM 凭证进行身份验证。只有在附加到代表 API 调用者的 IAM 用户、包含该用户的 IAM 组或该用户担任的 IAM 角色的 IAM 策略时，对 API 的调用才会成功。

  只有`AWS::Serverless::Api`资源类型支持 IAM 权限。

  有关更多信息，请参阅 。使用 IAM 权限控制对 API 的访问中的API Gateway 开发人员指南. 有关示例，请参阅 IAM 权限示例 (p. 334)。
- Amazon Cognito 用户池— Amazon Cognito 用户池是 Amazon Cognito 中的用户目录。API 的客户端必须首先将用户登录到用户池，然后获取该用户的身份或访问令牌。然后，客户端使用返回的令牌之一调用你的 API。只有在所需的令牌有效时，API 调用才会成功。

  这些区域有：`AWS::Serverless::Api`资源类型支持 Amazon Cognito 用户池。这些区域有：`AWS::Serverless::HttpApi`资源类型支持将 Amazon Cognito 用作 JWT 颁发者。

  有关更多信息，请参阅《API Gateway 开发人员指南》中的使用 Amazon Cognito 用户池作为授权方控制对 REST API 的访问。有关示例，请参阅 Amazon Cognito 用户池示例 (p. 335)。
- API 密钥— API 密钥是字母数字字符串值，可将它分发给应用程序开发人员（要向其授予对您的 API 的访问

  只有`AWS::Serverless::Api`资源类型支持 API 密钥。

  有关 API 密钥的更多信息，请参阅。创建和使用带 API 密钥的使用计划中的API Gateway 开发人员指南. 有关 API 密钥的示例，请参阅API 密钥 (p. 336).
- 资源策略— 资源策略是您可以附加到 API Gateway API 的 JSON 策略文档。使用资源策略控制指定的委托人（通常是 IAM 用户或角色）能否调用 API。

  只有`AWS::Serverless::Api`资源类型支持资源策略，作为控制对 API Gateway API 访问的机制。

  有关资源策略的更多信息，请参阅。使用 API Gateway 资源策略控制对 API 的访问中的API Gateway 开发人员指南. 有关资源策略的示例，请参阅。资源策略示例 (p. 336).
- OAuth 2.0/JWT 授权方— 你可以使用 JWT 作为OpenID Connect (OIDC)和OAuth 2.0框架以控制对 API 的访问。API Gateway 将验证客户端通过 API 请求提交的 JWT，并根据令牌中的令牌中的令牌验证和（可选）作用域来允许或拒绝请求。

  只有`AWS::Serverless::HttpApi`资源类型支持 OAuth 2.0/JWT 授权者。

  有关更多信息，请参阅《API Gateway 开发人员指南》中的使用 JWT 授权方控制对 HTTP API 的访问。有关示例，请参阅 OAuth 2.0/JWT 授权方示例 (p. 337)。

选择控制访问的机制

您选择用于控制 API Gateway API 访问权限的机制取决于几个因素。例如，如果您的绿地项目未设置授权或访问控制，那么 Amazon Cognito 用户池可能是您的最佳选择。这是因为在设置用户池时，您还会自动设置身份验证和访问控制。

但是，如果您的应用程序已经设置了身份验证，那么使用 Lambda 授权者可能是您的最佳选择。这是因为您可以调用现有的身份验证服务并根据响应返回策略文档。此外，如果您的应用程序需要用户池不支持的自定义身份验证或访问控制逻辑，那么 Lambda 授权者可能是您的最佳选择。

选择使用哪种机制后，请参阅中的相应部分示例 (p. 332)了解如何使用Amazon SAM将应用程序配置为使用该机制。

自定义错误响应

您可以使用Amazon SAM以自定义某些 API Gateway 错误响应的内容。只有`AWS::Serverless::Api`资源类型支持自定义 API Gateway 响应。

有关 API Gateway 响应的更多信息，请参阅。API Gateway 中的网关响应中的API Gateway 开发人员指南. 有关自定义响应的示例，请参阅。自定义响应示例 (p. 337).

示例

Lambda 授权方

这些区域有：`AWS::Serverless::Api`资源类型支持两种类型的 Lambda 授权方：`TOKEN`和授权方`REQUEST`授权方。这些区域有：`AWS::Serverless::HttpApi`仅支持资源类型`REQUEST`授权方。以下是每种类型的示例。

LambdaTOKEN授权方AWS::Serverless::Api)

您可以通过定义 Lambda 来控制对 API 的访问。`TOKEN`您的授权方Amazon SAMTemplate 要执行此操作，请使用ApiAuth (p. 92)数据类型。

以下是示例：Amazon SAMLambda 的模板部分`TOKEN`授权方

```
Resources:
  MyApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      Auth:
        DefaultAuthorizer: MyLambdaTokenAuthorizer
        Authorizers:
          MyLambdaTokenAuthorizer:
            FunctionArn: !GetAtt MyAuthFunction.Arn

  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: ./src
      Handler: index.handler
      Runtime: nodejs12.x
      Events:
        GetRoot:
          Type: Api
          Properties:
            RestApiId: !Ref MyApi
```

```
            Path: /
            Method: get

  MyAuthFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: ./src
      Handler: authorizer.handler
      Runtime: nodejs12.x
```

有关 Lambda 授权方的更多信息，请参阅。使用 API Gateway Lambda 授权方中的API Gateway 开发人员指南.

LambdaREQUEST授权方AWS::Serverless::Api ）

您可以通过定义 Lambda 来控制对 API 的访问。REQUEST您的授权方Amazon SAMTemplate 要执行此操作，请使用ApiAuth (p. 92)数据类型。

以下是示例：Amazon SAMLambda 的模板部分REQUEST授权方

```
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
```

有关 Lambda 授权方的更多信息，请参阅。使用 API Gateway Lambda 授权方中的API Gateway 开发人员指南.

Lambda 授权方示例 (AWS::Serverless::HttpApi ）

您可以 Lambda 过在Amazon SAMTemplate 要执行此操作，请使用HttpApiAuth (p. 215)数据类型。

以下是示例：Amazon SAMLambda 授权者的模板部分：

```
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 权限示例

您可以通过在自己的中定义 IAM 权限来控制对 API 的访问。Amazon SAMTemplate 要执行此操作，请使用ApiAuth (p. 92)数据类型。

以下是示例：Amazon SAMIAM 权限的模板部分：

```
Resources:
  MyApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      Auth:
        DefaultAuthorizer: AWS_IAM

  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: .
      Handler: index.handler
      Runtime: nodejs12.x
      Events:
```

```
        GetRoot:
          Type: Api
          Properties:
            RestApiId: !Ref MyApi
            Path: /
            Method: get
```

有关 IAM 权限的更多信息，请参阅。针对调用 API 的访问控制中的API Gateway 开发人员指南.

Amazon Cognito 用户池示例

您可以通过在自己的中定义 Amazon Cognito 用户池来控制对 API 的访问权限Amazon SAMTemplate 要执行此操作，请使用ApiAuth (p. 92)数据类型。

以下是示例：Amazon SAM用户池的模板部分：

```
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
```

有关 Amazon Cognito 用户池的更多信息，请参阅。使用 Amazon Cognito 用户池作为授权方控制对 REST API 的访问中的API Gateway 开发人员指南.

API 密钥

您可以通过下列方式控制对 API 的访问：Amazon SAMTemplate 要执行此操作，请使用ApiAuth (p. 92)数据类型。

以下是示例：Amazon SAMAPI 密钥的模板部分：

```
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
```

有关 API 密钥的更多信息，请参阅。创建和使用带 API 密钥的使用计划中的API Gateway 开发人员指南.

资源策略示例

您可以在Amazon SAMTemplate 要执行此操作，请使用ApiAuth (p. 92)数据类型。

以下是示例：Amazon SAM资源策略的模板部分：

```
Resources:
  ExplicitApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      EndpointConfiguration: PRIVATE
      Auth:
        ResourcePolicy:
          CustomStatements: {
              Effect: 'Allow',
              Action: 'execute-api:Invoke',
              Resource: ['execute-api:/*/*/*'],
              Principal: '*'
            }
  MinimalFunction:
    Type: 'AWS::Serverless::Function'
    Properties:
      CodeUri: s3://sam-demo-bucket/hello.zip
      Handler: hello.handler
```

```
      Runtime: python2.7
      Events:
        AddItem:
          Type: Api
          Properties:
            RestApiId:
              Ref: ExplicitApi
            Path: /add
            Method: post
```

有关资源策略的更多信息，请参阅。使用 API Gateway 资源策略控制对 API 的访问中的API Gateway 开发人员指南.

OAuth 2.0/JWT 授权方示例

您可以使用 JWT 来控制对 API 的访问。OpenID Connect (OIDC)和OAuth 2.0框架。要执行此操作，请使用HttpApiAuth (p. 215)数据类型。

以下是示例：Amazon SAMOAuth 2.0/JWT 授权方的模板部分：

```
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
```

有关 OAuth 2.0/JWT 授权方的更多信息，请参阅。使用 JWT 授权方控制对 HTTP API 的访问中的API Gateway 开发人员指南.

自定义响应示例

你可 API Gateway 过在你的Amazon SAMTemplate 要执行此操作，请使用网关响应对象数据类型。

以下是示例：Amazon SAMAPI Gateway 响应的模板部分：

```
Resources:
  MyApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      GatewayResponses:
        DEFAULT_4XX:
          ResponseParameters:
            Headers:
              Access-Control-Expose-Headers: "'WWW-Authenticate'"
              Access-Control-Allow-Origin: "'*'"

  GetFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.get
      Runtime: nodejs12.x
      InlineCode: module.exports = async () => throw new Error('Check out the response
 headers!')
      Events:
        GetResource:
          Type: Api
          Properties:
            Path: /error
            Method: get
            RestApiId: !Ref MyApi
```

有关 API Gateway 响应的更多信息，请参阅。API Gateway 中的网关响应中的API Gateway 开发人员指南.

使用编排Amazon资源Amazon Step Functions

您可以使用Amazon Step Functions来协调Amazon Lambda函数和其他Amazon资源，以形成复杂而强大的工作流程。

> Note
>
> 要管理包含 Step Functions 状态机的Amazon SAM模板，必须使用 0.52.0 或更高版本的Amazon SAM CLI。要检查使用的是哪个版本，请执行以下命令`sam --version`。

Step Functions 基于任务和状态机的概念。您可以使用基于 JSON 的亚马逊状态语言定义状态机。Ste p Functions 控制台显示状态机结构的图形视图，因此您可以直观地检查状态机的逻辑并监视执行情况。

利用Amazon Serverless Application Model (Amazon SAM) 中的 Step Functions 支持，您可以执行以下操作：

- 直接在Amazon SAM模板中或在单独的文件中定义状态机
- 通过Amazon SAM策略模板、内联策略或托管策略创建状态机执行角色
- 通过 API Gateway 或 Amazon EventBridge 事件、在Amazon SAM模板中按计划触发状态机执行，或者直接调用 API
- 使用可用的Amazon SAM策略模板来创建常见的 Step Functions 开发模式。

示例

以下来自Amazon SAM模板文件的示例片段在定义文件中定义了 Step Functions 状态机。请注意，该`my_state_machine.asl.json`文件必须使用亚马逊州语言编写。

```
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
      ...
```

要下载包含 Step Functions 状态机的示例Amazon SAM应用程序，请参阅《Amazon Step Functions开发人员指南》Amazon SAM中的 “使用创建 Step Functions 状态机”。

更多信息

要了解有关 Step Functions 及其与结合使用的更多信息Amazon SAM，请参阅以下内容：

- Amazon Step Functions 的工作原理
- Amazon Step Functions 和 Amazon Serverless Application Model
- 教程：使用创建 Step Functions 状态机Amazon SAM
- Amazon SAM规格： AWS::Serverless::StateMachine (p. 235)

为Amazon SAM应用程序配置代码签名

您可以使用Amazon SAM来启用无服务器应用程序的代码签名，以帮助确保仅部署可信代码。有关代码签名功能的更多信息，请参阅Amazon Lambda开发者指南中的为 Lambda 函数配置代码签名。

为无服务器应用程序配置代码签名之前，必须使用Amazon Signer 创建签名配置文件。您可以使用此签名配置文件执行以下任务：

1. 创建代码签名配置-声明`AWS::Lambda::CodeSigningConfig`资源以指定可信发布者的签名配置文件并设置验证检查的策略操作。您可以在与您的无服务器函数相同的Amazon SAM模板中、不同的Amazon SAM模板中或在模板中声明此对象。Amazon CloudFormation然后，您可以使用资源的亚马逊资源名称 (ARN) 指定函数的`CodeSigningConfigArn`属性，为无服务器函数启用代码签名。`AWS::Lambda::CodeSigningConfig`
2. 对@@ 代码进行签名-使用带`--signing-profiles`选项的`sam package`或`sam deploy`命令。

   Note

   要使用`sam package`或`sam deploy`命令成功签署您的代码，必须为使用这些命令的 Amazon S3 存储桶启用版本控制。如果您使用的是为您Amazon SAM创建的 Amazon S3 存储桶，则会自动启用版本控制。有关 Amazon S3 存储桶版本控制的更多信息以及在您提供的 Amazon S3 存储桶上启用版本控制的说明，请参阅《亚马逊简单存储服务用户指南》中的 Amazon S3 存储桶中使用版本控制。

当您部署无服务器应用程序时，Lambda 会对您启用代码签名的所有函数执行验证检查。Lambda 还会对这些函数所依赖的任何层执行验证检查。有关 Lambda 验证检查的更多信息，请参阅Amazon Lambda开发者指南中的签名验证。

示例

创建签名配置文件

要创建签名配置文件，运行以下命令：

```
aws signer put-signing-profile --platform-id "AWSLambda-SHA384-ECDSA" --profile-
name MySigningProfile
```

如果前面的命令成功，您将看到签名配置文件的 ARN 返回。例如：

```
{
    "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"
}
```

该profileVersionArn字段包含创建代码签名配置时要使用的 ARN。

创建代码签名配置并为函数启用代码签名

以下示例Amazon SAM模板声明AWS::Lambda::CodeSigningConfig资源并启用 Lambda 函数的代码签名。在此示例中，有一个可信配置文件，如果签名检查失败，则部署将被拒绝。

```
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"
```

为你的代码签名

您可以在打包或部署应用程序时对代码进行签名。使用sam package或sam deploy命令指定--signing-profiles选项，如以下示例命令所示。

打包应用程序时对函数代码进行签名：

```
sam package --signing-profiles HelloWorld=MySigningProfile --s3-bucket test-bucket --
output-template-file packaged.yaml
```

在打包应用程序时，对函数代码和函数所依赖的层进行签名：

```
sam package --signing-profiles HelloWorld=MySigningProfile MyLayer=MySigningProfile --s3-
bucket test-bucket --output-template-file packaged.yaml
```

对函数代码和层进行签名，然后执行部署：

```
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

要使用sam package或sam deploy命令成功签署您的代码，必须为使用这些命令的 Amazon S3 存储桶启用版本控制。如果您使用的是为您Amazon SAM创建的 Amazon S3 存储桶，则会自动启用版本控制。有关 Amazon S3 存储桶版本控制的更多信息以及在您提供的 Amazon S3 存储桶上启用版本控制的说明，请参阅《亚马逊简单存储服务用户指南》中的 Amazon S3 存储桶中使用版本控制。

使用以下方式提供签名配置文件sam deploy --guided

当您使用配置了代码签名的无服务器应用程序运行sam deploy --guided命令时，Amazon SAM会提示您提供用于代码签名的签名配置文件。有关sam deploy --guided提示的更多信息，请参阅Amazon SAM CLI 命令参考sam deploy (p. 412)中的。

构建无服务器应用程序

构建无服务器应用程序需要获取Amazon SAM模板文件、应用程序代码以及任何适用的特定语言文件和依赖关系，并将所有构建工件以正确的格式和位置放置在工作流程的后续步骤中。

例如，您可能希望本地测试应用程序，或者您可能希望使用Amazon SAM CLI 部署应用程序。这两项活动都使用应用程序的构建工件作为输入。

本节向您展示如何使用##·### （p. 408)命令通过构建无服务器应用程序Amazon SAM。您可以选择构建应用程序的所有功能和层，也可以选择构建应用程序的单个组件，例如特定的功能或层。

主题

构建应用程序

要构建您的无服务器应用程序，请使用##·### （p. 408)命令。此命令还会收集应用程序依赖项的构建工件，并将它们放在正确的格式和位置以供后续步骤（例如本地测试、打包和部署）使用。

您可以在`requirements.txt` (Python) 或`package.json` (Node.js) 等清单文件中指定应用程序的依赖关系，也可以使用函数资源的`Layers`属性来指定应用程序的依赖关系。该`Layers`属性包含 Lambda 函数所依赖的Amazon Lambda图层资源列表。

应用程序构建工件的格式取决于每个函数的`PackageType`属性。此属性的选项是：

- **`Zip`**— 包含应用程序代码及其依赖项的 .zip 文件存档。如果要将代码打包为 .zip 文件存档，则必须为函数指定 Lambda 运行时。
- **`Image`**— 容器映像，包括基本操作系统、运行时、扩展，以及应用程序代码及其依赖项。

有关 Lambda 包类型的更多信息，请参阅Amazon Lambda开发人员指南中的 Lambda 部署包。

主题

构建 .zip 文件存档

要将无服务器应用程序构建为 .zip 文件存档，请`PackageType: Zip`为无服务器函数声明。

Amazon SAM根据您指定的架构 (p. 130)构建应用程序。如果您未指定架构，则x86_64默认Amazon SAM使用。

如果您的 Lambda 函数依赖于具有原生编译程序的软件包，请使用标--use-container志。此标志在行为类似于 Lambda 环境的 Docker 容器中本地编译您的函数，因此在您部署它们到Amazon云时，它们的格式正确。

当您使用该--use-container选项时，默认情况下会从 Amazon ECR Pub licAmazon SAM 提取容器镜像。例如，如果您想从另一个存储库提取容器映像 DockerHub，则可以使用--build-image选项并提供备用容器映像的 URI。以下是使用 DockerHub 存储库中的容器镜像构建应用程序的两个示例命令：

```
# Build a Node.js 12 application using a container image pulled from DockerHub
sam build --use-container --build-image amazon/aws-sam-cli-build-image-nodejs12.x

# Build a function resource using the Python 3.8 container image pulled from DockerHub
sam build --use-container --build-image Function1=amazon/aws-sam-cli-build-image-python3.8
```

有关您可以使用的 URI 列表--build-image，请参阅镜像 (p. 458)其中包含多个支持的运行时的 DockerHub URI。

有关构建.zip 文件存档应用程序的其他示例，请参阅本主题后面的 “示例” 部分。

构建容器镜像

要将您的无服务器应用程序构建为容器镜像，请声明PackageType: Image您的无服务器函数。您还必须使用以下条目声明Metadata资源属性：

`Dockerfile`

与 Lambda 函数关联的 Docker 文件名称。

`DockerContext`

Dockerfile 的位置。

`DockerTag`

（可选）应用于构建映像的标签。

`DockerBuildArgs`

为编译生成参数。

以下是Metadata资源属性部分的示例：

```
Metadata:
  Dockerfile: Dockerfile
  DockerContext: ./hello_world
  DockerTag: v1
```

要下载使用Image包类型配置的示例应用程序，请参阅教程：部署 Hello World 应用程序 (p. 23)教程：部署 Hello World 应用程序。在询问您要安装哪种软件包类型的提示时，选择Image。

Note

如果您在 Dockerfile 中指定了多架构基础映像，则会为主机的架构Amazon SAM构建容器映像。要为不同的架构构构建，请指定使用特定目标架构的基础映像。

容器环境变量文件

要提供包含构建容器环境变量的 JSON 文件，sam build请在命令中使用--container-env-var-file参数。您可以提供适用于所有无服务器资源的单个环境变量，也可以为每个资源提供不同的环境变量。

格式

将环境变量传递到构建容器的格式取决于您为资源提供了多少环境变量。

要为所有资源提供单个环境变量，请指定如下所示的Parameters对象：

```
{
  "Parameters": {
    "GITHUB_TOKEN": "TOKEN_GLOBAL"
  }
}
```

要为每种资源提供不同的环境变量，请为每种资源指定对象，如下所示：

```
{
  "MyFunction1": {
    "GITHUB_TOKEN": "TOKEN1"
  },
  "MyFunction2": {
    "GITHUB_TOKEN": "TOKEN2"
  }
}
```

将您的环境变量另存为文件，例如名为env.json。以下命令使用此文件将您的环境变量传递到构建容器：

```
sam build --use-container --container-env-var-file env.json
```

优先顺序

- 您为特定资源提供的环境变量优先于所有资源的单一环境变量。
- 您在命令行上提供的环境变量优先于文件中的环境变量。

示例

示例 1：.zip 文件存档

以下sam build命令生成.zip 文件存档：

```
# 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 12 application using a container image pulled from DockerHub
sam build --use-container --build-image amazon/aws-sam-cli-build-image-nodejs12.x

# Build a function resource using the Python 3.8 container image pulled from DockerHub
sam build --use-container --build-image Function1=amazon/aws-sam-cli-build-image-python3.8
```

```
# Build and run your functions locally
sam build && sam local invoke

# For more options
sam build --help
```

示例 2：容器镜像

以下Amazon SAM模板作为容器镜像构建：

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      PackageType: Image
      ImageConfig:
        Command: ["app.lambda_handler"]
    Metadata:
      Dockerfile: Dockerfile
      DockerContext: ./hello_world
      DockerTag: v1
```

以下是 Dockerfile 的示例：

```
FROM public.ecr.aws/lambda/python:3.8

COPY app.py requirements.txt ./

RUN python3.8 -m pip install -r requirements.txt

# Overwrite the command by providing a different command directly in the template.
CMD ["app.lambda_handler"]
```

示例 3：npm ci

对于 Node.js 应用程序，您可以使用`npm ci`代替`npm install`来安装依赖项。要使用`npm ci`，请在您的 Lambda 函数的`Metadata`资源属性`UseNpmCi: True`下`BuildProperties`指定。要使用`npm ci`，您的应用程序中必须有 Lambda 函数`CodeUri`的`package-lock.json`或`npm-shrinkwrap.json`文件。

以下示例`npm ci`用于在运行时安装依赖项`sam build`：

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: hello-world/
      Handler: app.handler
      Runtime: nodejs14.x
      Architectures:
        - x86_64
      Events:
        HelloWorld:
          Type: Api
          Properties:
            Path: /hello
            Method: get
    Metadata:
      BuildProperties:
        UseNpmCi: True
```

在外部构建功能Amazon SAM

默认情况下，当您运行时sam build，Amazon SAM会生成所有函数资源。其他选项包括：

- 在之外构建所有函数资源Amazon SAM-如果您手动或通过其他工具构建所有函数资源，则sam build不是必需的。您可以跳过sam build并转到流程中的下一步，例如执行本地测试或部署应用程序。
- 在外部构建一些函数资源Amazon SAM-如果您Amazon SAM想在构建某些函数资源的同时在外部构建其他函数资源Amazon SAM，可以在Amazon SAM模板中指定。

在之外构建一些函数资源Amazon SAM

要在使用时Amazon SAM跳过某个函数sam build，请在Amazon SAM模板中配置以下内容：

1. 在函数中添加`SkipBuild: True`元数据属性。
2. 指定已构建函数资源的路径。

以下是`TestFunction`配置为跳过的示例。它的建成资源位于`built-resources/TestFunction.zip`。

```
TestFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: built-resources/TestFunction.zip
    Handler: TimeHandler::handleRequest
    Runtime: java11
  Metadata:
    SkipBuild: True
```

现在，当你运行时sam build，Amazon SAM将执行以下操作：

1. Amazon SAM将跳过配置为的函数`SkipBuild: True`。
2. Amazon SAM将构建所有其他函数资源并将它们缓存在`.aws-sam`构建目录中。
3. 对于跳过的函数，它们在`.aws-sam`构建目录中的模板将自动更新，以引用您的构建函数资源的指定路径。

   以下是`.aws-sam`构建目录`TestFunction`中缓存的模板的示例：

   ```
   TestFunction:
     Type: AWS::Serverless::Function
     Properties:
       CodeUri: ../../built-resources/TestFunction.zip
       Handler: TimeHandler::handleRequest
       Runtime: java11
     Metadata:
       SkipBuild: True
   ```

使用 esbuild 构建 Node.js Lambda 函数

要构建和打包 Node.jsAmazon Lambda 函数，你可以将Amazon SAM命令行界面与 es JavaScript build 打包器一起使用。esbuild 打包器支持你编写的 Lambda 函数 TypeScript。

要使用 esbuild 构建 Node.js Lambda 函数，请向您的`AWS:Serverless::Function`资源添加一个`Metadata`对象`esbuild`并为指定`BuildMethod`。当你运行sam build命令时，Amazon SAM使用 esbuild 捆绑你的 Lambda 函数代码。

元数据属性

该`Metadata`对象支持 esbuild 的以下属性。

BuildMethod

为您的应用程序指定打包器。`esbuild` 是唯一受支持的值。

BuildProperties

为您的 Lambda 函数代码指定编译属性。

该`BuildProperties`对象支持 esbuild 的以下属性。所有属性均为可选属性。默认情况下，Amazon SAM 使用您的 Lambda 函数处理程序作为入口点。

EntryPoints

为您的应用程序指定入口点。

外部

指定要从构建中省略的包列表。

格式

指定应用程序中生成 JavaScript 文件的输出格式。有关更多信息，请参阅 esbuild 网站中的格式。

加载程序

指定用于加载给定文件类型的数据的配置列表。

MainFields

指定在解析包时尝试导入哪些`package.json`字段。默认值为 `main,module`。

缩小

指定是否缩小捆绑的输出代码。默认值为 `true`。

OutExtension

自定义 esbuild 生成的文件的文件扩展名。有关更多信息，请参阅 esbuild 网站中的 Ou t 扩展。

源地图

指定打包器是否生成源映射文件。默认值为 `false`。

设置为时`true`，`NODE_OPTIONS: --enable-source-maps`将附加到 Lambda 函数的环境变量中，然后生成源映射并将其包含在函数中。

或者，`NODE_OPTIONS: --enable-source-maps`当包含在函数的环境变量中时，`Sourcemap`会自动设置为`true`。

发生冲突时，`Sourcemap: false`优先于`NODE_OPTIONS: --enable-source-maps`。

> **Note**
>
> 默认情况下，Lambda 使用Amazon Key Management Service (Amazon KMS) 加密所有静态环境变量。使用源映射时，要成功进行部署，您的函数的执行角色必须具有执行`kms:Encrypt`操作的权限。

SourcesContent

指定是否在源地图文件中包含源代码。设置为时`Sourcemap`配置此属性`true`。

- 指定SourcesContent: true包含所有源代码。
- 指定SourcesContent: false排除所有源代码。这会导致源地图文件大小变小，这可以通过缩短启动时间在生产中很有用。但是，调试器中将无法使用源代码。

默认值为 SourcesContent: true。

有关更多信息，请参阅 esbuild 网站中的源代码内容。

目标

指定目标 ECMAScript 版本。默认值为 es2020。

TypeScript Lambda 函数示例

以下示例Amazon SAM模板片段使用 esbuild 根据中的 TypeScript 代码创建 Node.js Lambda 函数hello-world/app.ts。

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: hello-world/
      Handler: app.handler
      Runtime: nodejs14.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
```

使用原生 AOT 编译构建 .NET 7 Lambda 函数

使用Amazon Serverless Application Model (Amazon SAM) 生成和打包您的 .NET 7Amazon Lambda 函数，利用原生提前 (AOT) 编译来缩短Amazon Lambda冷启动时间。

主题

- 了解更多信息 (p. 350)

.NET 7 原生 AOT 概述

过去，.NET Lambda 函数有冷启动时间，这会影响用户体验、系统延迟和无服务器应用程序的使用成本。Microsoft 在 .NET 7 中增加了对本机 AOT 编译的支持，它通过将 .NET 代码直接编译为机器和操作系统原生代码来提高性能，从而消除了运行时即时 (JIT) 编译。使用 .NET 7 原生 AOT 编译，您可以缩短 Lambda 函数的冷启动时间。要了解有关适用于 .NET 7 的原生 AOT 的更多信息，请参阅在 Dotnet GitHub 存储库中使用原生 AO T。

Amazon SAM与您的.NET 7 Lambda 函数一起使用

执行以下操作，使用Amazon Serverless Application Model (Amazon SAM) 配置您的 .NET 7 Lambda 函数：

- 在开发计算机上安装必备组件。
- 在您的Amazon SAM模板中定义 .NET 7 Lambda 函数。
- Amazon SAM使用 CLI 构建应用程序。

安装的先决条件

以下是必备的先决条件：

- CAmazon SAM LI
- .NET CLI
- Amazon.Lambda.Tools .NET 核心全球工具
- Docker

安装Amazon SAM CLI

1. 要检查是否已经安装了 CAmazon SAM LI，请运行以下命令：

```
sam --version
```

2. 要安装Amazon SAM CLI，请参阅安装 Amazon SAM CLI (p. 15)。
3. 要升级已安装的Amazon SAM CLI 版本，请参阅升级 CAmazon SAM LI (p. 443)。

安装 .NET CLE CLI

1. 要下载并安装 .NET Core CLI，请参阅从微软的网站下载 .NET CLI。
2. 有关 .NET Core CLI 的更多信息，请参阅《Amazon Lambda开发人员指南》中的 .NET Core CLI。

安装 Amazon.Lambda.Tools .NET Core Glo

1. 运行以下 命令：

```
dotnet tool install -g Amazon.Lambda.Tools
```

2. 如果您已安装该工具，请确保该工具是使用以下命令的最新版本：

```
dotnet tool update -g Amazon.Lambda.Tools
```

3. 有关 Amazon.Lambda.Tools .NET Core Global 的更多信息，请参阅上的适用于 .NET CLI 的Amazon扩展程序存储库 GitHub。

安装 Docker

- 使用原生 AOT 进行构建Docker，需要安装。有关安装说明，请参阅安装 Docker 以便与Amazon SAM CLI 一起使用 (p. 455)。

在您的Amazon SAM模板中定义 .NET 7 Lambda 函数

要在Amazon SAM模板中定义 .NET7 Lambda 函数，请执行以下操作：

- 通过设置`Runtime`为来定义自定义运行时`provided.al2`。我们定义自定义运行时是因为 .NET 7 运行时选项不可用。
- 向您的`AWS::Serverless::Function`资源添加一个`Metadata`对象并指定`dotnet7BuildMethod`。

```
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

对于该`Architectures`属性，`x86_64`是唯一支持的选项，因为.NET 7 无法在上运行`arm64`。有关更多信息，请参阅重新考虑.NET 7 编译环境 GitHub 问题。

Amazon SAM使用 CLI 构建应用程序

从项目的根目录运行`sam build`即可开始构建应用程序。如果已在 .NET 7 项目文件中定义了该`PublishAot`属性，则Amazon SAM CLI 将使用原生 AOT 编译进行构建。要了解有关该`PublishAot`属性的更多信息，请参阅 Microsoft .NET 文档中的原生 AOT 部署。

CLI 会调用 .NET CoreAmazon SAM CLI，该CLI 使用 Amazon.Lambda.Tools .NET Core Global Tool。

Note

构建时，如果`.sln`文件存在于项目的相同或父目录中，则包含该`.sln`文件的目录将挂载到容器中。如果找不到`.sln`文件，则只安装项目文件夹。因此，如果您要构建多项目应用程序，请确保该`.sln`文件位于属性中。

了解更多信息

有关构建 .NET 7 Lambda 函数的更多信息，请参阅使用 .NET 7 在 Lambda 上构建无服务器应用程序。

有关该sam build命令的参考，请参见山姆·布莱德 (p. 408)。

使用下列方法构建 Rust LambdaCargo Lambda

此功能位于 Amazon SAM 的预览版中，可能会发生变化。

在 RustAmazon Lambda 函数中使用Amazon SAM命令行界面 (Amazon SAMCLI)。

主题

先决条件

Cargo Lambda子命令

CAmazon SAM LI 需要安装的Cargo Lambda子命令Cargo。

- 要了解更多信息Cargo Lambda，请参阅什么是Cargo Lambda？
- 有关Cargo Lambda安装说明，请参阅Cargo Lambda文档中的安装。

选择加入Amazon SAM CLI 测试版功能

由于此功能处于预览阶段，您必须使用以下方法之一选择使用以下方法之一选择允许使用以下方法之一选择允许使用以下方法之一选择允许使用以下方法之一

1. 使用环境变量：`SAM_CLI_BETA_RUST_CARGO_LAMBDA=1`。
2. 在您的 `samconfig.toml` 文件中添加以下内容：

```
[default.build.parameters]
beta_features = true
[default.sync.parameters]
beta_features = true
```

3. 在使用支持的Amazon SAM CLI 命令时使用该`--beta-features`选项。例如：

```
$ sam build --beta-features
```

4. 在Amazon SAM CLI 提示您选择加入y时选择选项。

配置Amazon SAM为与 Rust Lambda 函数一起使用

步骤 1：配置允许配置允许配置允许配置允许配置Amazon SAM允许

使用以下内容配置您的Amazon SAM模板：

- 二进制-可选。指定您的模板何时包含多个 Rust Lambda 函数。
- BuildMethod – `rust-cargolambda`.

- CodeUri— 您的`Cargo.toml`文件路径。
- 处理器-`bootstrap`.
- 运行时间-`provided.al2`.

要了解有关自定义运行时的更多信息，请参阅Amazon Lambda开发者指南中的自定义Amazon Lambda运行时。

以下是已配置Amazon SAM模板的示例：

```
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
...
```

第 2 步：将Amazon SAM CLI 与你的 Rust Lambda 函数结合使用

在您的Amazon SAM模板中使用任何Amazon SAM CLI 命令。有关Amazon SAM CLI 命令的更多信息，请参阅Amazon SAMCLI 命令参考 (p. 408)。

示例

单一 Lambda 函数项目 Lam

以下是包含一个 Rust Lambda 函数的无服务器应用程序的示例。

项目目录结构：

```
.
### Cargo.lock
### Cargo.toml
### src
#   ### main.rs
### template.yaml
```

Amazon SAM模板：

```
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
```

```
...
```

多个 Lambda 函数项目

以下是包含多个 Rust Lambda 函数的无服务器应用程序的示例。

项目目录结构：

```
.
### Cargo.lock
### Cargo.toml
### src
#   ### function_a.rs
#   ### function_b.rs
### template.yaml
```

Amazon SAM模板：

```
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文件：

```
[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"
```

建筑层

您可以使用Amazon SAM来构建自定义图层。有关层的信息，请参阅《Amazon Lambda开发者指南》中的AmazonLambda 层。

要构建自定义层，请在Amazon Serverless Application Model (Amazon SAM) 模板文件中声明该图层，并添加一个带有BuildMethod条目的Metadata资源属性部分。的有效值BuildMethod是Amazon Lambda运行时的标识符，或makefile。包括一个BuildArchitecture条目来指定您的层支持的指令集架构。的有效值BuildArchitecture是 Lambda 指令集架构。

如果您指定makefile，请提供自定义 makefile，在其中声明build-*layer-logical-id*包含图层构建命令的表单的构建目标。如有必要，您的 makefile 负责编译图层，并将构建工件复制到工作流程后续步骤所需的正确位置。makefile 的位置由图层资源的ContentUri属性指定，必须命名Makefile。

> Note
>
> 创建自定义层时，Amazon Lambda依靠环境变量来查找层代码。Lambda 运行时包含将层代码复制到的/opt目录中的路径。您的项目的生成工件文件夹结构必须与运行时的预期文件夹结构相匹配，这样才能找到您的自定义层代码。
> 例如，对于 Python，您可以将代码放在python/子目录中。对于 NodeJS，您可以将代码放在nodejs/node_modules/子目录中。
> 有关更多信息，请参阅《Amazon Lambda开发者指南》中的在层中包含库依赖关系。

以下是Metadata资源属性部分的示例。

```
Metadata:
  BuildMethod: python3.8
  BuildArchitecture: arm64
```

> Note
>
> 如果您不包括Metadata资源属性部分，则Amazon SAM不会构建图层。相反，它会从图层资源CodeUri属性中指定的位置复制构建工件。有关更多信息，请参阅AWS::Serverless::LayerVersion资源类型的ContentUri (p. 229)属性。

在包含Metadata资源属性部分时，可以使用##·### (p. 408)命令来构建图层，既可以作为独立对象，也可以作为Amazon Lambda函数的依赖关系。

- 作为一个独立的对象。 您可能只想构建图层对象，例如，当您在本地测试图层的代码更改时，不需要构建整个应用程序。要独立构建图层，请使用sam build *layer-logical-id*命令指定图层资源。
- 作为 Lambda 函数的依赖关系。 当您在同一个Amazon SAM模板文件的 Lambda 函数的Layers属性中包含图层的逻辑 ID 时，该层就是该 Lambda 函数的依赖关系。当该层还包括带有BuildMethod条目的Metadata资源属性部分时，您可以通过使用命令构建整个应用程序或使用sam build命令指定函数资源来构建该sam build *function-logical-id*层。

示例

模板示例 1：针对 Python 3.6 运行时环境构建层

以下示例Amazon SAM模板针对 Python 3.6 运行时环境构建了一个层。

```
Resources:
  MyLayer:
    Type: AWS::Serverless::LayerVersion
    Properties:
      ContentUri: my_layer
      CompatibleRuntimes:
        - python3.9
    Metadata:
      BuildMethod: python3.9              # Required to have Amazon SAM build this layer
```

模板示例 2：使用自定义 makefile 构建图层

以下示例Amazon SAM模板使用自定义makefile来构建图层。

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

以下内容makefile包含构建目标和将要执行的命令。请注意，该ContentUri属性设置为my_layer，因此 makefile 必须位于my_layer子目录的根目录中，文件名必须为Makefile。另请注意，构建工件被复制到python/子目录中，以便Amazon Lambda能够找到层代码。

```
build-MyLayer:
    mkdir -p "$(ARTIFACTS_DIR)/python"
    cp *.py "$(ARTIFACTS_DIR)/python"
    python -m pip install -r requirements.txt -t "$(ARTIFACTS_DIR)/python"
```

sam 编译命令示例

以下sam build命令构建包含Metadata资源属性部分的图层。

```
# 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
```

构建自定义运行时

您可以使用该##·### (p. 408)命令构建 Lambda 函数所需的自定义运行时。通过指定Runtime: provided Lambda 函数，您可以声明该函数使用自定义运行时。

要构建自定义运行时，请使用BuildMethod: makefile条目声明Metadata资源属性。您提供自定义 makefile，在其中声明build-*function-logical-id*包含运行时生成命令的表单的生成目标。如有必要，您的 makefile 负责编译自定义运行时，并将构建工件复制到工作流程中后续步骤所需的正确位置。makefile 的位置由函数资源的CodeUri属性指定，必须命名Makefile。

示例

示例 1：用 Rust 编写的函数的自定义运行时

> Note
>
> 我们建议使用构建 Lambda 函数Cargo Lambda。要了解更多信息，请参阅 使用下列方法构建 Rust LambdaCargo Lambda (p. 351)。

以下Amazon SAM模板声明了一个为用 Rust 编写的 Lambda 函数使用自定义运行时的函数，并指示`sam build`为编`build-HelloRustFunction`译目标执行命令。

```
Resources:
  HelloRustFunction:
    Type: AWS::Serverless::Function
    Properties:
      FunctionName: HelloRust
      Handler: bootstrap.is.real.handler
      Runtime: provided
      MemorySize: 512
      CodeUri: .
    Metadata:
      BuildMethod: makefile
```

以下 makefile 包含编译目标和将要执行的命令。请注意，该`CodeUri`属性设置为`.`，因此 makefile 必须位于项目根目录中（即与应用程序Amazon SAM模板文件相同的目录）。文件名必须是`Makefile`。

```
build-HelloRustFunction:
  cargo build --release --target x86_64-unknown-linux-musl
  cp ./target/x86_64-unknown-linux-musl/release/bootstrap $(ARTIFACTS_DIR)
```

有关设置开发环境以执行前`makefile`面`cargo build`命令的更多信息，请参阅Amazon Lambda博客文章 Rust Runtime。

示例 2：适用于 Python3.7 的 Makefile 生成器（使用捆绑生成器的替代方案）

您可能需要使用捆绑生成器中未包含的库或模块。这个例子显示了一个带有 makefile 生成器的 Python3.7 运行时的Amazon SAM模板。

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: hello_world/
      Handler: app.lambda_handler
      Runtime: python3.7
    Metadata:
      BuildMethod: makefile
```

以下 makefile 包含编译目标和将要执行的命令。请注意，该`CodeUri`属性设置为`hello_world`，因此 makefile 必须位于`hello_world`子目录的根目录中，并且文件名必须是`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
```

测试和调试无服务器应用程序

使用Amazon SAM命令行界面 (CLI)，您可以在将应用程序上传到Amazon云之前，在本地测试和 “逐步调试” 无服务器应用程序。在完成打包和部署应用程序的步骤之前，您可以验证您的应用程序是否按预期运行，调试错误之处并修复任何问题。

当您在 CLAmazon SAM I 中以调试模式在本地调用 Lambda 函数时，您可以将其连接到调试程序。使用调试器，您可以逐行检查代码，查看各种变量的值，并修复问题，就像处理任何其他应用程序一样。

> Note
>
> 如果您的应用程序包含一个或多个层，则当您在本地运行和调试应用程序时，图层包将下载并缓存在您的本地主机上。有关更多信息，请参阅 图层在本地缓存方式 (p. 327)。

主题

在本地调用 Lambda 函数

您可以使用`sam #### (p. 423)`Amazon SAM CLI 命令并提供Amazon Lambda函数的逻辑 ID 和事件文件，在本地调用您的函数。或者，sam local invoke也可以接受`stdin`为事件。有关事件的更多信息，请参阅Amazon Lambda开发者指南中的事件。有关来自不同Amazon服务的事件消息格式的信息，请参阅Amazon Lambda开发者指南中的Amazon Lambda与其他服务一起使用。

> Note
>
> 该sam local invoke命令对应于Amazon Command Line Interface (Amazon CLI) 命令`aws lambda invoke`。您可以使用任一命令调用 Lambda 函数。

必须在要调用的函数的项目目录中运行sam local invoke命令。

示例：

```
# 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
```

环境变量文件

要在本地声明环境变量以覆盖模板中定义的值，请执行以下操作：

1. 创建一个 JSON 文件，其中包含要覆盖的环境变量。
2. 使用`--env-vars`参数覆盖模板中定义的值。

声明环境变量

要声明全局应用于所有资源的环境变量，请指定如下所示的`Parameters`对象：

```
{
    "Parameters": {
        "TABLE_NAME": "localtable",
        "BUCKET_NAME": "testBucket",
        "STAGE": "dev"
    }
}
```

要为每种资源声明不同的环境变量，请为每个资源指定对象，如下所示：

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

为每种资源指定对象时，您可以使用以下标识符，按优先级从高到低的顺序列出：

1. `logical_id`
2. `function_id`
3. `function_name`
4. 完整路径标识符

您可以使用上述两种方法在单个文件中一起声明环境变量。这样做时，您为特定资源提供的环境变量优先于全局环境变量。

将您的环境变量保存在 JSON 文件中，例如`env.json`。

重写环境变量值

要使用 JSON 文件中定义的环境变量替换环境变量，请将`--env-vars`参数与invoke或start-api命令一起使用。例如：

```
sam local invoke --env-vars env.json
```

层

如果您的应用程序包含层，有关如何调试本地主机上层问题的信息，请参阅使用图层 (p. 326)。

了解更多信息

有关在本地调用函数的实践示例，请参阅 “完整Amazon SAM研讨会” 中的模块 2-本地运行。

在本地运行 API Gateway

要启动可用于测试 HTTP 请求/响应功能Amazon API Gateway 本地实例，请使用sam #### api (p. 425)Amazon SAM CLI 命令。此功能具有热重载功能，因此您可以快速开发和迭代您的函数。

> Note
>
> 热重装是指仅刷新已更改的文件，并且应用程序的状态保持不变。相比之下，实时重新加载是指刷新整个应用程序，并且应用程序的状态丢失。

必须在包含要调用的函数的项目目录中运行该sam local start-api命令。

默认情况下，Amazon SAM使用Amazon Lambda代理集成并同时支持HttpApi和Api资源类型。有关HttpApi资源类型的代理集成的更多信息，请参阅《API Gate way 开发者指南》中的 HTTP API 使用 Amazon Lambda代理集成。有关与Api资源类型进行代理集成的更多信息，请参阅《API Gateway 开发者指南》中的 “了解 API Gatew ay Lambda 代理集成”。

示例：

```
sam local start-api
```

Amazon SAM自动查找Amazon SAM模板中定义了事件源HttpApi或Api事件源的所有函数。然后，它将函数安装到定义的 HTTP 路径上。

在以下Api示例中，Ratings函数在 f/ratings orGET 请求ratings.py:handler()时挂载：

```
Ratings:
  Type: AWS::Serverless::Function
  Properties:
    Handler: ratings.handler
    Runtime: python3.9
    Events:
      Api:
        Type: Api
        Properties:
          Path: /ratings
          Method: get
```

以下是Api响应示例：

```
// 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"
    });
}
```

环境变量文件

要在本地声明覆盖模板中定义的值的环境变量，请执行以下操作：

1. 创建一个 JSON 文件，其中包含要重写的环境变量。
2. 使用--env-vars参数覆盖模板中定义的值。

声明环境变量

要声明全局应用于所有资源的环境变量，请指定如下所示的Parameters对象：

```
{
    "Parameters": {
        "TABLE_NAME": "localtable",
        "BUCKET_NAME": "testBucket",
        "STAGE": "dev"
    }
}
```

要为每种资源声明不同的环境变量，请为每种资源指定对象，如下所示：

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

为每个资源指定对象时，您可以使用以下标识符，按优先级从高到低的顺序列出：

1. logical_id
2. function_id
3. function_name
4. 完整路径标识符

您可以使用上述两种方法在单个文件中一起声明环境变量。这样做时，您为特定资源提供的环境变量优先于全局环境变量。

将环境变量保存在 JSON 文件中，例如env.json。

重写环境变量值

要使用 JSON 文件中定义的环境变量替换环境变量，请使用带invoke或start-api命令的--env-vars参数。例如：

```
sam local start-api --env-vars env.json
```

层

如果您的应用程序包含层，有关如何调试本地主机上的层问题的信息，请参阅使用图层 (p. 326)。

与自动测试集成

您可以使用sam local invoke命令可通过在本地运行 Lambda 函数手动测试代码。使用Amazon SAMCLI，您可以通过首先针对本地 Lambda 函数运行测试来轻松创作自动集成测试，然后再部署到Amazon 云。

这些区域有：`sam local start-lambda`命令启动一个本地终端节点，该终端节点模拟Amazon Lambda调用终端节点。你可以从自动测试中调用它。因为此终端节点会模拟Amazon Lambda调用终端节点，您可以编写测试一次，然后针对本地 Lambda 函数或部署的 Lambda 函数运行测试（无需做任何修改）。您还可以针对部署的Amazon SAM堆栈在您的 CI/CD 管道中。

以下是过程的工作原理：

1. 启动本地 Lambda 终端节点。

   通过在包含您的目录中运行以下命令来启动本地 Lambda 终端节点。Amazon SAM模板：

   ```
   sam local start-lambda
   ```

   此命令在于启动本地终端节点`http://127.0.0.1:3001`模拟Amazon Lambda. 您可以针对此本地 Lambda 终端节点运行自动测试。当您使用Amazon CLI或 SDK，它会在本地执行请求中指定的 Lambda 函数，并返回响应。

2. 针对本地 Lambda 终端节点运行集成测试。

   在集成测试中，你可以使用Amazon通过开发工具包通过测试数据调用您的 Lambda 函数，等待响应并验证响应符合您的预期。要在本地运行集成测试，您应该配置Amazon用于发送 Lambda Invoke API 调用以调用您在上一步中启动的本地 Lambda 终端节点的软件开发工具包。

   以下是 Python 示例（Amazon适用于其他语言的 SDK 具有相似的配置）：

   ```
   import boto3
   import botocore

   # Set "running_locally" flag if you are running the integration test locally
   running_locally = True

   if running_locally:

       # Create Lambda SDK client to connect to appropriate Lambda endpoint
       lambda_client = boto3.client('lambda',
           region_name="us-west-2",
           endpoint_url="http://127.0.0.1:3001",
           use_ssl=False,
           verify=False,
           config=botocore.client.Config(
               signature_version=botocore.UNSIGNED,
               read_timeout=1,
               retries={'max_attempts': 0},
           )
       )
   else:
       lambda_client = boto3.client('lambda')


   # Invoke your Lambda function as you normally usually do. The function will run
   # locally if it is configured to do so
   response = lambda_client.invoke(FunctionName="HelloWorldFunction")

   # Verify the response
   assert response == "Hello World"
   ```

   您可以使用此代码通过设置来测试已部署的 Lambda 函数`running_locally`到`False`. 设置Amazon要连接的 SDKAmazon Lambda中的Amazon云。

生成示例事件

为了简化 Lambda 函数的本地开发和测试，您可以为多个AmazonAPI Gateway 之类的服务，Amazon CloudFormation、Amazon S3 等。

有关可为其生成示例事件有效负载的服务的完整列表，请使用以下命令：

```
sam local generate-event --help
```

有关可用于特定服务的选项列表，请使用以下命令：

```
sam local generate-event [SERVICE] --help
```

示例：

```
#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
```

在本地逐步调试 Lambda 函数

您可以使用Amazon SAM有各种Amazon用于在本地测试和调试无服务器应用程序的工具包和调试器。

例如，您可以通过设置断点、检查变量和一次一行执行函数代码来执行 Lambda 函数的本地逐步调试。本地逐步调试通过使您能够发现和解决云中可能遇到的问题，从而缩小反馈循环。

使用Amazon工具箱

AmazonToolkit 是集成开发环境 (IDE) 插件，它使您能够执行许多常见的调试任务，例如设置断点、检查变量以及一行一行执行函数代码。Amazon此工具箱使您能够更轻松地开发、调试和部署使用生成的无服务器应用程序。Amazon SAM. 它们提供了构建、测试、调试、部署和调用集成到 IDE 中的 Lambda 函数的体验。

有关 的更多信息Amazon你可以与之配合使用的工具包Amazon SAM，请参阅：

- Amazon Toolkit for Visual Studio Code
- Amazon Cloud9
- Amazon Toolkit for JetBrains

有很多Amazon适用于 IDE 和运行时的不同组合的工具包。下表列出了支持逐步调试的常见 IDE/Runtime 组合Amazon SAM应用程序：

IDE	运行时	Amazon Toolkit	逐步调试说明
Visual Studio 代码	• Node.js • Python • .NET • Java • Go（转到）	Amazon Toolkit for Visual Studio Code	使用Amazon Serverless Application中的Amazon Toolkit for Visual Studio Code用户指南

IDE	运行时	Amazon Toolkit	逐步调试说明
Amazon Cloud9	• Node.js • Python	Amazon Cloud9，与Amazon启用工具包[1]	使用Amazon无服务器应用程序使用Amazon工具包中的Amazon Cloud9用户指南.
WebStorm	Node.js	Amazon Toolkit for JetBrains[2]	运行（调用）或调试本地函数中的Amazon Toolkit for JetBrains
pyCharm	Python	Amazon Toolkit for JetBrains[2]	运行（调用）或调试本地函数中的Amazon Toolkit for JetBrains
骑手	.NET	Amazon Toolkit for JetBrains[2]	运行（调用）或调试本地函数中的Amazon Toolkit for JetBrains
IntelliJ	Java	Amazon Toolkit for JetBrains[2]	运行（调用）或调试本地函数中的Amazon Toolkit for JetBrains
戈兰	Go（转到）	Amazon Toolkit for JetBrains[2]	运行（调用）或调试本地函数中的Amazon Toolkit for JetBrains

备注:

1. 使用Amazon Cloud9逐步调试Amazon SAM应用程序，Amazon必须启用工具包。有关更多信息，请参阅 。启用Amazon工具包中的Amazon Cloud9用户指南.
2. 使用Amazon Toolkit for JetBrains逐步调试Amazon SAM要进行安装和配置，您必须首先按照中的说明进行安装和配置。安装Amazon Toolkit for JetBrains中的Amazon Toolkit for JetBrains.

正在运行Amazon SAM在调试模式下本地

除了与Amazon您还可在其他实例上运行Amazon SAM在 “调试模式” 中附加到第三方调试器ptvsd要么深入研究.

运行Amazon SAM在调试模式下，使用命令sam 本地调用 (p. 423)要么sam 本地启动 api (p. 425)使用`--debug-port`要么`-d`选项。

例如：

```
# 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

如果使用`sam local start-api`，本地 API Gateway 实例会公开您的所有 Lambda 函数。但是，因为您可以指定单个调试端口，所以每次只能调试一个函数。你需要在Amazon SAMCLI 绑定到端口，该端口允许调试器连接。

传递其他运行时调试参数

要在调试函数时传递额外的运行时参数，请使用环境变量DEBUGGER_ARGS. 这将一串参数直接传递到 run 命令中Amazon SAMCLI 用来启动你的功能。

例如，如果您希望在 Python 函数运行时加载像 ikPDB 这样的调试器，可以将以下命令传递如下：DEBUGGER_ARGS: -m ikpdb --ikpdb-port=5858 --ikpdb-working-directory=/var/task/ --ikpdb-client-working-directory=/myApp --ikpdb-address=0.0.0.0. 这将在运行时加载 ikPDB 与您指定的其他参数一起加载。

在这种情况下，你的满Amazon SAMCLI 命令如下：

```
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
```

你可以将调试器参数传递给所有运行时的函数。

使用Amazon CloudFormation Linter 验证您的Amazon SAM应用程序

Amazon CloudFormationLinter (cfn-lint) 是一个开源工具，可用于对Amazon CloudFormation模板进行详细验证。CFN-Lint 包含以Amazon CloudFormation资源规范为指导的规则。使用 cfn-lint 将您的资源与这些规则进行比较，以接收有关错误、警告或信息建议的详细消息。或者，创建自己的自定义规则进行验证。要了解有关 cfn-lint 的更多信息，请参阅Amazon CloudFormation GitHub 存储库中的 cfn-lint。

你可以使用 cfn-lint 通过Amazon SAM命令行界面Amazon Serverless Application Model (Amazon SAMAmazon SAMCLI) 通过--lint选项来验证你的 () 模板。sam validate

```
sam validate --lint
```

要自定义 cfn-lint 行为，例如创建自定义规则或指定验证选项，可以定义配置文件。要了解更多信息，请参阅cfn -lintAmazon CloudFormation GitHub 存储库中的Conf ig 文件。运行时sam validate --lint，将应用配置文件中定义的 cfn-lint 行为。

示例

对Amazon SAM模板执行 cfn-lint 验证

```
sam validate --lint --template myTemplate.yaml
```

了解更多信息

要了解有关 sam validate 命令的更多信息，请参阅 sam validate (p. 440)。

部署无服务器应用程序

Amazon SAM用Amazon CloudFormation作底层部署机制。有关更多信息，请参阅《Amazon CloudFormation 用户指南》中的什么是 Amazon CloudFormation？。部署无服务器应用程序的标准输入是使用山姆·布莱德 (p. 408)命令创建的构建工件。有关 sam build 的更多信息，请参阅 构建无服务器应用程序 (p. 342)。

您可以使用Amazon SAM命令行接口（CLI）命令手动部署应用程序。您还可以使用持续集成和持续部署（CI/CD）系统自动应用程序部署。您可以使用许多常见的 CI/CD 系统来部署Amazon SAM应用程序，包括 Jenkins Amazon CodePipeline、C GitLab I/CD 和 Acti GitHubon s。

使用 CI/CD 系统部署

Amazon SAM帮助组织为其首选的 CI/CD 系统创建管道，以便他们能够以最少的努力实现 CI/CD 的好处，例如加快部署频率、缩短变更交付时间和减少部署错误。

Amazon SAM借助构建容器映像，简化了无服务器应用程序的 CI/CD 任务。Amazon SAM提供的镜像包括用于许多支持的Amazon Lambda运行时的Amazon SAM CLI 和编译工具。这使得使用Amazon SAM CLI 构建和打包无服务器应用程序变得更加容易。这些映像还减少了团队为 CI/CD 系统创建和管理自己的映像的需求。有关Amazon SAM构建容器镜像的更多信息，请参阅镜像 (p. 458)。

多个 CI/CD 系统支持Amazon SAM构建容器映像。您应该使用哪个 CI/CD 系统取决于多个因素。其中包括您的应用程序是使用单个运行时还是多个运行时，或者您想在容器映像内构建应用程序，还是直接在主机（虚拟机 (VM) 或裸机主机）上构建应用程序。

Amazon SAM还为多个 CI/CD 系统提供了一组默认管道模板，这些模板封装Amazon了部署最佳实践。这些默认管道模板使用标准的 JSON/YAML 管道配置格式，内置的最佳实践有助于执行多账户和多区域部署，并验证管道无法对基础架构进行意外更改。

部署无服务器应用程序有两个主要选项：1) 修改现有流水线配置以使用Amazon SAM CLI 命令，或 2) 生成示例 CI/CD 管道配置，您可以将其用作自己的应用程序的起点。Amazon SAM

有关这些选项的更多信息，请参阅以下主题：

- 修改现有的 CI/CD 管道 (p. 367)
- 生成入门 CI/CD 管道 (p. 370)

使用Amazon SAM CLI 进行部署

在本地开发和测试无服务器应用程序后，您可以使用sam deploy (p. 412)命令部署应用程序。

要通过提示Amazon SAM指导您完成部署，请指定标--guided志。当您指定此标志时，该sam deploy命令会压缩您的应用程序工件，将它们上传到亚马逊Simple Storage Service (Amazon S3)（用于.zip 文件存档）或亚马逊Elastic Container Registry (Amazon ECR)（用于包含图像）。然后，该命令将您的应用程序部署到Amazon云端。

示例：

```
# Deploy an application using prompts:
```

```
sam deploy --guided
```

使用Amazon SAM CLI 对部署进行故障排除

Amazon SAMCLI 错误：“不满足安全限制”

跑步时sam deploy --guided，系统会提示你提出问题`HelloWorldFunction may not have authorization defined, Is this okay? [y/N]`。如果您以**N**（默认响应）回应此提示，会显示以下错误：

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

该提示通知您，您要部署的应用程序可能未经授权就配置了 Amazon API Gateway API。**N**回应此提示，就是说这不行。

要解决此问题，您可进行以下选择：

- 使用授权配置您的应用程序。有关配置授权的信息，请参阅控制对 API Gateway API 的访问 (p. 330)。
- 回复此问题**Y**以表明您可以部署未经授权配置了 API Gateway API 的应用程序。

逐步部署

如果您想逐步部署Amazon SAM应用程序而不是一次全部部署，则可以指定Amazon CodeDeploy提供的部署配置。有关更多信息，请参阅《Amazon CodeDeploy用户指南》中的 CodeDeploy “使用部署配置”。

有关配置Amazon SAM应用程序以逐步部署的信息，请参阅逐步部署无服务器应用程序 (p. 460)。

了解更多信息

有关部署无服务器应用程序的实际操作示例，请参阅 “完整Amazon SAM研讨会” 中的以下内容：

- 模块 3-手动部署- 学习如何使用Amazon SAM CLI 构建、打包和部署无服务器应用程序。
- 模块 4-CI/CD — 学习如何通过创建持续集成和交付 (CI/CD) 管道来实现构建、打包和部署阶段的自动化。

修改现有的 CI/CD 管道

根据您使用的 CI/CD 系统，使用Amazon SAM现有 CI/CD 管道部署无服务器应用程序的过程略有不同。

以下主题提供了配置 CI/CD 系统以在构建容器映像中Amazon SAM构建无服务器应用程序的示例：

主题

- 使用 GitHub 操作部署 (p. 370)

使用部署Amazon CodePipeline

要将Amazon CodePipeline管道配置为自动生成和部署Amazon SAM应用程序，您的Amazon CloudFormation模板和`buildspec.yml`文件必须包含执行以下操作的行：

1. 使用可用镜像中的必要运行时间引用构建容器镜像。以下示例使用`public.ecr.aws/sam/build-nodejs14.x`构建容器镜像。
2. 配置管道阶段以运行必要的Amazon SAM命令行接口 (CLI) 命令。以下示例运行两个Amazon SAM CLI 命令：sam build和sam deploy（带有必要的选项）。

此示例假设您已使用声明了Amazon SAM模板文件中的所有函数和层`runtime: nodejs14.x`。

Amazon CloudFormation模板片段：

```
  CodeBuildProject:
    Type: AWS::CodeBuild::Project
    Properties:
      Environment:
        ComputeType: BUILD_GENERAL1_SMALL
        Image: public.ecr.aws/sam/build-nodejs14.x
        Type: LINUX_CONTAINER
      ...
```

`buildspec.yml`片段：

```
version: 0.2
phases:
  build:
    commands:
      - sam build
      - sam deploy --no-confirm-changeset --no-fail-on-empty-changeset
```

有关适用于 Amazon Elastic Container Registry (Amazon ECR) 的可用容器映像列表，请参阅镜像 (p. 458)。

使用 Bitbucket 管道进行

要将 Bitbucket Pipel ine 配置为自动生成和部署Amazon SAM应用程序，您的`bitbucket-pipelines.yml`文件必须包含执行以下操作的行：

1. 使用可用镜像中的必要运行时间引用构建容器镜像。以下示例使用`public.ecr.aws/sam/build-nodejs14.x`构建容器镜像。
2. 配置管线阶段以运行必要的Amazon SAM命令行接口 (CLI) 命令。以下示例运行两个Amazon SAM CLI 命令：sam build和sam deploy（带有必要的选项）。

此示例假设您已使用声明了Amazon SAM模板文件中的所有函数和层`runtime: nodejs14.x`。

```
image: public.ecr.aws/sam/build-nodejs14.x

pipelines:
  branches:
    main: # branch name
```

```
    - step:
        name: Build and Package
        script:
          - sam build
          - sam deploy --no-confirm-changeset --no-fail-on-empty-changeset
```

有关适用于不同运行时的 Amazon Elastic Container Registry 的可用列表，请参阅镜像 (p. 458)。

使用詹金斯部署

要将 Jenkins 管道配置为自动生成和部署Amazon SAM应用程序，Jenkinsfile必须包含执行以下操作的行：

1. 使用可用镜像中的必要运行时间引用构建容器镜像。以下示例使用public.ecr.aws/sam/build-nodejs14.x构建容器镜像。
2. 配置管道阶段以运行必要的Amazon SAM命令行接口 (CLI) 命令。以下示例运行两个Amazon SAM CLI 命令：sam build和sam deploy（带有必要的选项）。

此示例假设您已使用声明了Amazon SAM模板文件中的所有函数和层runtime: nodejs14.x。

```
pipeline {
    agent { docker { image 'public.ecr.aws/sam/build-nodejs14.x' } }
    stages {
        stage('build') {
            steps {
                sh 'sam build'
                sh 'sam deploy --no-confirm-changeset --no-fail-on-empty-changeset'
            }
        }
    }
}
```

有关适用于 Amazon Elastic Container Registry (Amazon ECR) 以与不同运行时一起构建容器镜像的列表，请参阅镜像 (p. 458)。

使用 GitLab CI/CD 部署

要将GitLab管道配置为自动生成和部署Amazon SAM应用程序，您的gitlab-ci.yml文件必须包含执行以下操作的行：

1. 使用可用镜像中的必要运行时间引用构建容器镜像。以下示例使用public.ecr.aws/sam/build-nodejs14.x构建容器镜像。
2. 配置管道阶段以运行必要的Amazon SAM命令行接口 (CLI) 命令。以下示例运行两个Amazon SAM CLI 命令：sam build和sam deploy（带有必要的选项）。

此示例假设您已使用声明了Amazon SAM模板文件中的所有函数和层runtime: nodejs14.x。

```
image: public.ecr.aws/sam/build-nodejs14.x
deploy:
  script:
    - sam build
    - sam deploy --no-confirm-changeset --no-fail-on-empty-changeset
```

有关适用于不同运行时的 Amazon Elastic Container Registry (Amazon ECR) Registic Registic Registic Regist镜像 (p. 458)

使用 GitHub 操作部署

要配置GitHub管道以自动生成和部署Amazon SAM应用程序，必须先在主机上安装Amazon SAM命令行界面(CLI)。您可以在 GitHub 工作流程中使用GitHub 操作来帮助完成此设置。

以下示例 GitHub 工作流程使用一系列 GitHub操作设置 Ubuntu 主机，然后运行Amazon SAM CLI 命令来构建和部署Amazon SAM应用程序：

```
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
```

有关适用于 Amazon Elastic Container Registry (Amazon ECR) 的可用容器镜像列表，请参阅镜像 (p. 458)。

生成入门 CI/CD 管道

当您准备好以自动化方式部署无服务器应用程序时，可以为您选择的 CI/CD 系统生成部署管道。 Amazon SAM提供了一组入门流水线模板，您可以使用这些模板在几分钟内使用sam 管道 init (p. 435)命令生成管道。

入门工作流模板使用熟悉的 CI/CD 系统的 JSON/YAML 语法，并采用了最佳实践，例如管理多个账户和区域的工件，以及使用部署应用程序所需的最低权限。目前，Amazon SAMCLI 支持为 Jenkins、CI/CD Amazon CodePipeline、Actions 和 Bitbu GitLab cket 管道生成入门 CI/CD 管道配置。GitHub

以下是生成启动管道配置需要执行的高级任务：

1. 创建基础设施资源 — 您的管道需要某些Amazon资源，例如，具有必要权限的 IAM 用户和角色、Amazon S3 存储桶以及可选的 Amazon ECR 存储库。
2. 将您的 Git 存储库与 CI/CD 系统Connec t ——您的 CI/CD 系统需要知道哪个 Git 存储库将触发管道运行。请注意，此步骤可能不是必需的，具体取决于您使用的 Git 存储库和 CI/CD 系统的组合。
3. 生成您的管道配置-此步骤生成包含两个部署阶段的启动管道配置。
4. 将@@ 您的管道配置提交到 Git 存储库 — 此步骤是确保您的 CI/CD 系统知道您的管道配置并在提交更改后运行所必需的。

生成启动管道配置并将其提交到您的 Git 存储库后，每当有人对该存储库提交代码更改时，您的管道都会被触发自动运行。

这些步骤的顺序以及每个步骤的详细信息因您的 CI/CD 系统而异：

- 如果您正在使用 Amazon CodePipeline，请参阅 正在生成启动管道Amazon CodePipeline (p. 371)。

- 如果你使用的是 Jenkins、 GitLab CI/CD、 GitHub 操作或 Bitbucket 管道，请参阅为 Jenkins、 GitLab CI/CD、Actions 或 Bitbucket 管道生成启动管道 GitHub (p. 372)。

正在生成启动管道Amazon CodePipeline

要为生成启动流水线配置Amazon CodePipeline，请按以下顺序执行以下任务：

1. 创建基础设施资源
2. 生成管道配置
3. 将您的工作流配置提交到 Git
4. 将您的 Git 存储库与 CI/CD 系统Connect

> Note
>
> 以下过程使用了两个Amazon SAM CLI 命令，sam #### (p. 433)和sam ## init (p. 435)。之所以有两个命令，是为了处理这样的用例：管理员（即需要权限才能设置基础设施Amazon资源的用户，如 IAM 用户和角色）比开发人员（即只需要设置单个管道的权限，而不是所需的基础设施Amazon资源的用户）拥有更多的权限。

第 1 步：创建基础设施资源

使用的管道Amazon SAM需要某些Amazon资源，例如具有必要权限的 IAM 用户和角色、Amazon S3 存储桶以及可选的 Amazon ECR 存储库。您必须为管道的每个部署阶段提供一组基础架构资源。

您可以运行以下命令来帮助完成此设置：

```
sam pipeline bootstrap
```

> Note
>
> 为管道的每个部署阶段运行先前的命令。

第 2 步：生成管道配置

要生成管道配置，请运行以下命令：

```
sam pipeline init
```

第 3 步：将您的工作流配置提交到 Git 存储库

此步骤是确保您的 CI/CD 系统知道您的工作流配置所必需的，并且将在提交更改后运行。

第 4 步：将 Git 存储库与 CI/CD 系统Connect

因为Amazon CodePipeline您现在可以通过运行以下命令创建连接：

```
sam deploy -t codepipeline.yaml --stack-name <pipeline-stack-name> --
capabilities=CAPABILITY_IAM --region <region-X>
```

如果您使用的是 GitHub 或 Bitbucket，则在先前运行sam deploy命令后，按照开发者工具控制台用户指南中 “更新待处理的连接” 主题中找到的 “完成连接” 下的步骤完成连接。此外，还应存储sam deploy命令输出CodeStarConnectionArn中的副本，因为如果您想与之外的其他分支Amazon CodePipeline一起使用，则需要使用该副本main。

配置其他分支

默认情况下，Amazon CodePipeline使用分main支和Amazon SAM。如果要使用以外的分支main，则必须再次运行该sam deploy命令。请注意，根据使用的不是 Git 存储库，则还可能需要提供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 Amazon CodeCommit
sam deploy -t codepipeline.yaml --stack-name <feature-pipeline-stack-name> --
capabilities=CAPABILITY_IAM --parameter-overrides="FeatureGitBranch=<branch-name>"
```

了解更多信息

有关设置 CI/CD 管道的实践示例，请参阅《完整Amazon SAM研讨会》中的 CI/CD with。Amazon CodePipeline

为 Jenkins、 GitLab CI/CD、Actions 或 Bitbucket 管道生成启动管道 GitHub

要为 Jenkins、C GitLab I/CD、Ac GitHub tions 或 Bitbucket Pipelines 生成启动流水线配置，请按以下顺序执行以下任务：

1. 创建基础设施资源
2. 将您的 Git 存储库与 CI/CD 系统Connect
3. 创建凭证对象
4. 生成管道配置
5. 将您的工作流配置提交到 Git 存储库

> **Note**
>
> 以下过程使用了两个Amazon SAM CLI 命令，sam #### (p. 433)和sam ## init (p. 435)。之所以有两个命令，是为了处理这样的用例：管理员（即需要权限才能设置基础设施Amazon资源的用户，如 IAM 用户和角色）比开发人员（即只需要设置单个管道的权限，而不是所需的基础设施Amazon资源的用户）拥有更多的权限。

步骤 1：创建基础设施资源

使用的管道Amazon SAM需要某些Amazon资源，例如具有必要权限的 IAM 用户和角色、Amazon S3 存储桶以及可选的 Amazon ECR 存储库。您必须为管道的每个部署阶段提供一组基础架构资源。

您可以运行以下命令以帮助完成此设置：

```
sam pipeline bootstrap
```

> **Note**
>
> 为管道的每个部署阶段运行先前的命令。

您必须获取管道每个部署阶段的管道用户的Amazon证书（密钥 ID 和密钥），因为后续步骤需要这些证书。

第 2 步：将 Git 存储库与 CI/CD 系统Connect

必须将 Git 存储库连接到 CI/CD 系统，以便 CI/CD 系统能够访问您的应用程序源代码以进行构建和部署。

Note

如果您使用的是以下组合之一，则可以跳过此步骤，因为连接是自动完成的：

1. GitHub 使用 GitHub 存储库执行的操作
2. GitLab 带 GitLab 存储库的 CI/CD
3. 带有 Bitbucket 存储库的 Bitbu

要将 Git 存储库与 CI/CD 系统连接，请执行以下操作之一：

- 如果你使用的是 Jenkins，请参阅 Jenkins 文档 “添加分支源”。
- 如果您使用的是 GitLab CI/CD 以外的 Git 存储库 GitLab，请参阅 “连接外部存储库” GitLab文档。

步骤 3：创建凭证对象

每个 CI/CD 系统都有自己的方式来管理 CI/CD 系统访问您的 Git 存储库所需的证书。

要创建必要的凭证对象，请执行以下操作之一：

- 如果您使用的是 Jenkins，请创建一个存储密钥 ID 和密钥的单个 “凭据”。按照 “配置 Jenkins” 部分的 “使用Amazon SAM博客构建 Jenkins 流水线” 中的说明进行操作。在下一个步骤中，您需要用到 “凭证 ID”。
- 如果您使用的是 C GitLab I/CD，请创建两个 “受保护变量”，分别对应密钥 ID 和密钥。按照GitLab 文档中的说明进行操作——下一步将需要两个 “可变密钥”。
- 如果您使用的是 Ac GitHub tions，请创建两个 “加密密钥”，分别对应密钥和密钥。按照GitHub文档中的说明进行操作-下一步需要两个 “机密名称”。
- 如果您使用的是 Bitbucket 管道，请创建两个 “安全变量”，分别对应密钥 ID 和密钥。按照变量和机密中的说明进行操作——下一步需要两个 “机密名称”。

步骤 4：生成流水线配置

要生成管道配置，请运行以下命令。您需要输入在上一步中创建的凭证对象：

```
sam pipeline init
```

第 5 步：将您的工作流配置提交到 Git 存储库

此步骤是确保您的 CI/CD 系统知道您的工作流配置所必需的，并且将在提交更改后运行。

了解更多信息

有关使用设置 CI/CD 管道的实践示例GitHub Actions，请参阅《完整Amazon SAM研讨会》中的 CI/CD with。GitHub

自定义入门管线

作为 CI/CD 管理员，您可能需要自定义入门管道模板和相关的指导性提示，组织中的开发人员可以使用这些模板来创建管道配置。

Amazon SAMCLI 在创建入门模板时使用 Cookiecutter 模板。有关曲奇刀模板的详细信息，请访问 Cookiecutter。

您还可以使用`sam pipeline init`命令自定义Amazon SAM CLI 在创建管道配置时向用户显示的提示。要自定义用户提示，请执行以下操作：

1. 创建**`questions.json`**文件-该`questions.json`文件必须位于项目存储库的根目录中。这与`cookiecutter.json`文件是同一个目录。要查看该`questions.json`文件的架构，请参阅 questions.json.schema。要查看示例`questions.json`文件，请参阅 questions.json。
2. 将@@ 问题键映射为千篇一律的名称 —**`questions.json`** 文件中的每个对象都需要一个与 cookiecutter 模板中的名称相匹配的密钥。这种密钥匹配是Amazon SAM CLI 将用户提示响应映射到千篇一律模板的方式。要查看此密钥匹配示例，请参阅本主题后示例文件 (p. 374)面的部分。
3. 创建**`metadata.json`**文件-声明管道在文件中将`metadata.json`包含的阶段数。阶段数指示`sam pipeline init`命令要提示多少个阶段的信息，或者如果是`--bootstrap`选项，则指示要为多少阶段创建基础架构资源。要查看声明具有两个阶段的管道的示例`metadata.json`文件，请参阅 metadata.json。

示例项目

以下是示例项目，每个项目都包含 Cookiecutter 模板、`questions.json`文件和`metadata.json`文件：

- 詹金斯示例：两阶段 Jenkins 管道模板
- CodePipeline 示例：两阶段 CodePipeline 管道模板

示例文件

以下一组文件显示了`questions.json`文件中的问题如何与 Cookiecutter 模板文件中的条目相关联。请注意，这些示例是文件片段，而不是完整文件。要查看完整文件示例，请参阅本主题前示例项目 (p. 374)面的部分。

示例：**`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"
  }, {
    ...
```

示例：**`cookiecutter.json`**

```
{
  "outputDir": "aws-sam-pipeline",
  "pipeline_user_jenkins_credential_id": "",
  "sam_template": "",
```

```
    ...
```

示例：**Jenkinsfile**

```
pipeline {
  agent any
  environment {
    PIPELINE_USER_CREDENTIAL_ID = '{{cookiecutter.pipeline_user_jenkins_credential_id}}'
    SAM_TEMPLATE = '{{cookiecutter.sam_template}}'
    ...
```

在Amazon SAM管道中使用 OIDC 身份验证

Amazon Serverless Application Model(Amazon SAM) 支持 Bitbucket、Ac GitHub tions 以及 GitLab 持续集成和持续交付 (CI/CD) 平台的 OpenID Connect (OIDC) 用户身份验证。有了这种支持，您就可以使用这些平台上的授权 CI/CD 用户帐户来管理您的无服务器应用程序管道。否则，您将需要创建和管理多个Amazon Identity and Access Management (IAM) 用户来控制对Amazon SAM管道的访问权限。

使用Amazon SAM管道设置 OIDC

在`sam pipeline bootstrap`配置过程中，执行以下操作以使用您的Amazon SAM管道设置 OIDC。

1. 当提示选择身份提供者时，选择 OIDC。
2. 接下来，选择支持的 OIDC 提供商。
3. 输入 OIDC 提供商 URL，开头为**`https://`**。

   Note

   Amazon SAM生成`AWS::IAM::OIDCProvider`资源类型时引用此 URL。
4. 接下来，按照提示输入访问所选平台所需的 CI/CD 平台信息。这些详细信息因平台而异，可能包括：
   - OUC 客户端 ID。
   - 代码存储库名称或通用唯一标识符 (UUID)。
   - 与存储库关联的通用或组织名称。
   - GitHub 代码存储库所属的组织。
   - GitHub 存储库名称。
   - 进行部署的分支。
5. Amazon SAM显示输入的 OIDC 配置的摘要。输入设置的编号进行编辑，或按下Enter继续。
6. 当提示确认创建支持输入的 OIDC 连接所需的资源时，Y按下继续。

Amazon SAM使用提供的配置生成`AWS::IAM::OIDCProvider`Amazon CloudFormation资源，该资源担任管道执行角色。要了解有关此Amazon CloudFormation资源类型的更多信息，请参阅Amazon CloudFormation用户指南中的 AWS:: IAM:: OIDCProvid er。

Note

如果您的中已经存在身份提供商 (IdP) 资源Amazon Web Services 账户，请Amazon SAM引用该资源，而不是创建新资源。

示例

以下是使用Amazon SAM管道设置 OIDC 的示例。

```
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
    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]:
```

了解更多信息

有关在Amazon SAM管道中使用 OIDC 的更多信息，请参阅sam 管道引导 (p. 433)。

监控无服务器应用程序

配置和监视您的Amazon Serverless Application Model (Amazon SAM) 应用程序。

主题

使用应用程序洞察监控您的无服务器 CloudWatch 应用程序

Amazon A CloudWatch pplication Insights 可帮助您监控应用程序中的Amazon资源，以帮助识别潜在问题。它可以分析Amazon资源数据以寻找问题迹象，并构建自动仪表板以将其可视化。您可以将 Applicati CloudWatch on Insights 配置为与您的Amazon Serverless Application Model (Amazon SAM) 应用程序一起使用。要了解有关 CloudWatch 应用程序见解的更多信息，请参阅亚马逊 CloudWatch 用户指南中的亚马逊 CloudWatch 应用程序见解。

主题

使用以下 CloudWatch 方式配置应用程序见解Amazon SAM

通过Amazon SAM命令行界面 (Amazon SAMCLI) 或Amazon SAM模板为您的 CloudWatch Amazon SAM应用程序配置 Application Insights。

通过Amazon SAM CLI 进行配置

使用初始化应用程序时sam init，通过交互式流程或使用--application-insights选项激活 CloudWatch Application Insights。

要通过Amazon SAM CLI 交互式流程激活 CloudWatch 应用程序见解，请在出现提示**y**时输入。

```
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]:
```

要使用该--application-insights选项激活 App CloudWatch lication Insights，请执行以下操作。

```
sam init --application-insights
```

要了解有关sam init命令的更多信息，请参阅sam init (p. 414)。

通过Amazon SAM模板配置

通过在Amazon SAM模板中定义`AWS::ResourceGroups::Group`和`AWS::ApplicationInsights::Application`资源来激活 App CloudWatch lication Insights。

```
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— 通过使用 ResourAmazon ce Groups，同时管理和自动完成针对大量资源的任务。在这里，您可以创建一个资源组以与 Application Insights 一起 CloudWatch 使用。有关此资源类型的更多信息，请参阅AWS::ResourceGroups::GroupAmazon CloudFormation用户指南中的。
- AWS::ApplicationInsights::Application— 为资源组配置 CloudWatch 应用程序见解。有关此资源类型的更多信息，请参阅AWS::ApplicationInsights::ApplicationAmazon CloudFormation用户指南中的。

这两种资源都会Amazon CloudFormation在应用程序部署时自动传递给。您可以使用Amazon SAM模板中的Amazon CloudFormation语法进一步配置 Application CloudWatch Insights。有关更多信息，请参阅亚马逊 CloudWatch 用户指南中的使用Amazon CloudFormation模板。

使用sam init --application-insights命令时，这两个资源都会在您的Amazon SAM模板中自动生成。以下是生成的模板示例。

```
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
```

后续步骤

配置 Appl CloudWatch ication Insights 后sam deploy，用于sam build构建应用程序和部署应用程序。Appl CloudWatch ication Insights 支持的所有资源都将配置为进行监控

- 有关支持的资源列表，请参阅亚马逊 CloudWatch 用户指南中的支持的日志和指标。
- 要了解如何访问 CloudWatch 应用程序见解，请参阅亚马逊 CloudWatch 用户指南中的访问 CloudWatch 应用程序见解。

使用日志

为了简化故障排除，Amazon SAMCLI 有一个名为的命令sam log (p. 430). 此命令让您可以获取从命令行中由您的 Lambda 函数生成的日志。

> Note
>
> 这些区域有：`sam logs`命令适用于所有人Amazon Lambda函数，而不仅仅是您使用的部署函数Amazon SAM.

通过获取日志Amazon CloudFormation堆

当你的函数是一部分时Amazon CloudFormation堆栈中，您可以使用该函数的逻辑 ID 获取日志：

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

按 Lambda 函数名称获取日志

或者，您可以使用该函数的名称获取日志：

```
sam logs -n mystack-HelloWorldFunction-1FJ8PD
```

结尾日志

添加`--tail`选项在新日志到达时查看它们。这在部署期间或在解决生产问题时非常有用。

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

查看特定时间范围的日志

您可以使用`-s`和`-e`选项

```
sam logs -n HelloWorldFunction --stack-name mystack -s '10min ago' -e '2min ago'
```

筛选日志

使用`--filter`选项在日志事件中快速查找匹配字词、短语或值的日志：

```
sam logs -n HelloWorldFunction --stack-name mystack --filter "error"
```

在输出中，Amazon SAMCLI 为单词 “error” 的所有匹配项添加下划线，以便您可以在日志输出中轻松找到筛选关键字。

突出显示时

当您的 Lambda 函数崩溃或超时时，Amazon SAMCLI 以红色突出显示超时消息。这有助于您轻松找到在日志输出巨流中超时的特定执行。

JSON 漂亮的打印

如果您的日志消息打印 JSON 字符串，则Amazon SAMCLI 自动整齐打印 JSON 以帮助您直观地解析和理解 JSON。

使用Amazon SAM CLI 发布无服务器应用程序

要使您的Amazon SAM应用程序可供其他人查找和部署，您可以使用Amazon SAM CLI 将其发布到Amazon Serverless Application Repository。要使用Amazon SAM CLI 发布应用程序，必须使用Amazon SAM模板对其进行定义。您还必须在本地或Amazon云端对其进行了测试。

按照本主题中的说明创建新应用程序、创建现有应用程序的新版本或更新现有应用程序的元数据。（您要做什么取决于应用程序中是否已存在Amazon Serverless Application Repository，以及是否有任何应用程序元数据正在更改。）有关应用程序元数据的更多信息，请参阅Amazon SAM元数据部分属性 (p. 384)。

先决条件

在Amazon Serverless Application Repository使用Amazon SAM CLI 将应用程序发布到之前，必须具备以下条件：

- CAmazon SAM LI 已安装。有关更多信息，请参阅 安装 Amazon SAM CLI (p. 15)。要确定Amazon SAM CLI 是否已安装，请运行以下命令：

```
sam --version
```

- 有效的Amazon SAM模板。
- 您的应用程序代码和Amazon SAM模板引用的依赖关系。
- 语义版本，仅用于公开共享您的应用程序。这个值可以像 1.0 一样简单。
- 指向应用程序源代码的 URL。
- 一个 `README.md` 文件。此文件应描述客户如何使用您的应用程序以及在将应用程序部署到自己的Amazon账户之前如何对其进行配置。
- 一个`LICENSE.txt`文件，只需要公开共享您的应用程序。
- 如果您的应用程序包含任何嵌套应用程序，则必须已将它们发布到Amazon Serverless Application Repository。
- 有效的亚马逊Simple Storage Service (Amazon S3) 存储桶策略，该策略授予服务对您在打包应用程序时上传到 Amazon S3 的对象的读取权限。要设置此策略，请执行以下操作：

  1. 通过以下网址打开 Simple Storage Service（Amazon S3）控制台：https://console.aws.amazon.com/s3/。
  2. 选择用于打包应用程序的 Amazon S3 存储桶的名称。
  3. 请选择权限。
  4. 在 Permissions（权限）标签页中，在 Bucket policy（存储桶策略）下，请选择 Edit（编辑）。
  5. 在“编辑存储桶策略”页面上，将以下策略声明粘贴到策略编辑器中。在政策声明中，确保在元素中使用您的存储桶名称，在`ResourceCondition`元素中使用您的Amazon账户 ID。`Condition`元素中的表达式确保仅Amazon Serverless Application Repository有权访问来自指定Amazon帐户的应用程序。有关策略声明的更多信息，请参阅 IAM 用户指南中的 IAM JSON 策略元素参考。

```
{
    "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. 选择 Save changes（保存更改）。

发布新应用程序

步骤 1：向Amazon SAM模板添加分Metadata区

首先，在Amazon SAM模板中添加一个Metadata分区。提供要发布到的应用程序信息Amazon Serverless Application Repository。

以下是示例Metadata部分：

```
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
        ...
```

有关Amazon SAM模板Metadata部分的更多信息，请参阅Amazon SAM元数据部分属性 (p. 384)。

步骤 2：Package 应用程序

运行以下Amazon SAM CLI 命令，将应用程序的工件上传到 Amazon S3 并输出一个名为的新模板文件packaged.yaml：

```
sam package --output-template-file packaged.yaml --s3-bucket <your-bucket-name>
```

在下一步中，您可以使用packaged.yaml模板文件将应用程序发布到Amazon Serverless Application Repository。此文件与原始模板文件 (template.yaml) 类似，但它有一个关键区别，即CodeUriLicenseUrl、和ReadmeUrl属性指向 Amazon S3 存储桶和包含相应构件的对象。

来自示例 packaged.yaml 模板文件的以下代码段显示了 CodeUri 属性：

```
MySampleFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: s3://bucketname/fbd77a3647a4f47a352fc0ObjectGUID

...
```

步骤 3：发布应用程序

要将Amazon SAM应用程序的私有版本发布到Amazon Serverless Application Repository，请运行以下 Amazon SAM CLI 命令：

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

该sam publish命令的输出包括指向上您的应用程序的链接Amazon Serverless Application Repository。您也可以直接进入Amazon Serverless Application Repository登录页面搜索您的应用程序。

步骤 4：共享应用程序（可选）

默认情况下，您的应用程序设置为私有，因此其他Amazon帐户不可见。要与他人共享您的应用程序，您必须将其设为公开或向特定Amazon帐户列表授予权限。

有关使用共享应用程序的信息Amazon CLI，请参阅《Amazon Serverless Application Repository开发人员指南》中的Amazon Serverless Application Repository基于资源的策略示例。有关使用共享应用程序的信息Amazon Web Services Management Console，请参阅《Amazon Serverless Application Repository开发人员指南》中的共享应用程序。

发布现有应用程序的新版本

将应用程序发布到后Amazon Serverless Application Repository，您可能需要发布该应用程序的新版本。例如，您可能更改了 Lambda 函数代码或向应用程序架构添加了新组件。

要更新之前发布的应用程序，请使用前面介绍的相同流程再次发布该应用程序。在Amazon SAM模板文件的Metadata部分中，提供与最初发布模板文件相同的应用程序名称，但要包含一个新SemanticVersion值。

例如，假设一个以的名称SampleApp和为发布SemanticVersion的应用程序1.0.0。要更新该应用程序，该Amazon SAM模板必须具有应用程序名称SampleApp和的1.0.1（或除SemanticVersion之外的任何其他名称1.0.0）。

其他主题

- Amazon SAM元数据部分属性 (p. 384)

Amazon SAM元数据部分属性

AWS::ServerlessRepo::Application是元数据密钥，可用于指定要发布到Amazon Serverless Application Repository.

Note

Amazon CloudFormation 内部函数不支持AWS::ServerlessRepo::Application元数据键

属性

此表提供有关的属性的信息Metadata的 部分Amazon SAM模板。本节需要将应用程序发布到Amazon Serverless Application Repository使用Amazon SAMCLI。

属性	类型	必需	描述
Name	字符串	TRUE	应用程序的名称。 最小长度 = 1。最大长度 = 140。 模式："[a-zA-Z0-9\\-]+";
Description	字符串	TRUE	关于应用程序的描述。 最小长度 = 1。最大长度 = 256。
Author	字符串	TRUE	发布应用程序的作者的姓名。 最小长度 = 1。最大长度 = 127。 模式："^[a-z0-9](([a-z0-9]|-(?!-))*[a-z0-9])?$";
SpdxLicenseId	字符串	FALSE	有效的许可证标识符。要查看有效许可证标识符的列表，请参阅SPDX 许可证列表在软件包 Data Exchange (SPDX)网站.
LicenseUrl	字符串	FALSE	对本地许可证文件的引用，或指向许可证文件的 Amazon S3 链接，该文件与应用程序的 spdxLicSseID 值匹配。 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的Amazon SAM尚未使用sam package命令可以引用此属性的本地文件。但是，要使用sam publish命令中，此属性必须引用 Amazon S3 存储桶。 最大大小：5 MB。 您必须为此属性提供一个值，才能使您的应用程序变为公有的。请注意，应用程序发布后，您无法更新此属性。因此，要向应用程序添加许可证，您必须先删除该许可证，或者发布具有其他名称的新应用程序。
ReadmeUrl	字符串	FALSE	对本地自述文件的引用或指向自述文件的 Amazon S3 链接，其中包含对应用程序及其工作原理的更详细描述。 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的Amazon SAM尚未使用sam package命令可以引用此属性的本地文件。但是，要使用sam publish命令中，此属性必须引用 Amazon S3 存储桶。 最大大小：5 MB。
Labels	字符串	FALSE	改善在搜索中发现应用程序的结果的标签。 最小长度 = 1。最大长度 = 127。最大标签数量：10.

属性	类型	必需	描述
			模式："^[a-zA-Z0-9+\\-_:\\/@]+$";
HomePageUrl	字符串	FALSE	一个 URL，其中包含有关应用程序的更多信息例如，应用程序的 GitHub 存储库的位置。
SemanticVersion	字符串	FALSE	应用程序的语义版本。有关语义版本控制规范，请参阅语义版本控制网站. 您必须为此属性提供一个值，才能使您的应用程序变为公有的。
SourceCodeUrl	字符串	FALSE	指向应用程序源代码的公共存储库的链接。

使用案例

本节列出了发布应用程序的使用案例以及Metadata为该用例处理的属性。哪些属性不为给定用例列出的将被忽略。

- 创建新应用程序— 如果中没有应用程序，则创建一个新的应用程序Amazon Serverless Application Repository使用账户的名称匹配。
  - Name
  - SpdxLicenseId
  - LicenseUrl
  - Description
  - Author
  - ReadmeUrl
  - Labels
  - HomePageUrl
  - SourceCodeUrl
  - SemanticVersion
  - 的内容Amazon SAM模板（例如，任何事件源、资源和 Lambda 函数代码）

- 创建应用程序版本— 如果中已有应用程序，则创建应用程序版本Amazon Serverless Application Repository使用账户的名称匹配和语义版是正在改变。
  - Description
  - Author
  - ReadmeUrl
  - Labels
  - HomePageUrl
  - SourceCodeUrl
  - SemanticVersion
  - 的内容Amazon SAM模板（例如，任何事件源、资源和 Lambda 函数代码）

- 更新应用程序— 如果中已有应用程序，则应用程序将更新Amazon Serverless Application Repository使用账户的名称匹配和语义版不是正在改变。
  - Description
  - Author

- ReadmeUrl
- Labels
- HomePageUrl

示例

以下是示例：Metadata部分:

```
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
```

使用第三方服务

本节提供有关在第三方服务中使用Amazon Serverless Application Model (Amazon SAM) 的文档。

主题

Amazon SAMCLI Terraform 支持

> Terraform 支持是Amazon SAM CLI 的预览版，可能会发生变化。要提供反馈和提交功能请求，请创建GitHub问题。

在 Terraform 项目中使用Amazon Serverless Application Model命令行界面 (Amazon SAMCLI) 对Amazon Lambda函数和层进行本地调试和测试。

要了解有关 Terraform 的更多信息，请参阅：

- HashiCorp Terraform 用于概述 Terraform。
- 入门-Amazon 在 Terraform 的开发者文档中，开始使用 Terraform 和Amazon。

主题

Amazon SAMCLI 对 Terraform 的支持是什么？

> Terraform 支持是Amazon SAM CLI 的预览版，可能会发生变化。要提供反馈和提交功能请求，请创建GitHub问题。

在 Terraform 项目中使用Amazon Serverless Application Model命令行界面 (Amazon SAMCLI) 对Amazon Lambda函数和层进行本地调试和测试。

主题

使用Amazon SAM CLI 进行本地调试和测试

CAmazon SAM LI 支持您的 Terraform 项目使用以下命令：

- sam build— 将 Lambda 资源Package 到您的 Terraform 项目中，以便与Amazon SAM CLI 一起用于本地调试和测试。有关 sam build 的更多信息，请参阅山姆·布莱德 (p. 408)。
- sam local invoke— 调用一次Amazon Lambda函数。有关 sam local invoke 的更多信息，请参阅sam 本地调用 (p. 423)。
- sam local start-lambda— 启动 Lambda 函数的本地终端节点，以便使用Amazon Command Line Interface (Amazon CLI) 或 SDK 在本地调用您的函数。有关 sam local start-lambda 的更多信息，请参阅sam local start-la (p. 427)。

Amazon SAMCLI 如何与你的 Terraform 项目互动

Amazon SAMCLI 使用 Terraform 命令检查您的项目状态，以便识别 Lambda 资源及其源和包构件。（可选）需要在 Terraform 配置文件中定义元数据资源以供Amazon SAM CLI 参考。在Amazon SAM CLI 对 Terraform 的支持下，你可以：

- 用于sam build为本地测试做准备。
- 使用sam local invoke和sam local start-lambda来调试和测试您的 Lambda 函数。

Amazon SAMCLI Terraform 支持的好处

借助Amazon SAM CLI 对 Terraform 的支持，您可以在应用和部署更改之前立即在本地调试和测试 Lambda 函数，从而加快开发和测试流程及工作流程。

后续步骤

要为在 Terraform 上使用Amazon SAM CLI 做好准备，请参阅开始使用 (p. 389)。

开始使用 Terraform 对Amazon SAM CLI 的支持

> Terraform 支持是Amazon SAM CLI 的预览版，可能会发生变化。要提供反馈和提交功能请求，请创建GitHub问题。

Amazon Serverless Application Model命令行界面 (Amazon SAMCLI) 支持对您的 Terraform 项目中的 Amazon Lambda功能和层进行本地调试和测试。

主题

Amazon SAMCLI 地形先决条件

完成所有先决条件，即可开始在您的 Terraform 项目中使用Amazon SAM CLI。

1. 安装 Python 3.6 或更新版

   需要使用 Python 3.6 或更高版本才能与Amazon SAM CLI 一起使用。有关安装说明，请参阅 Python 的初学者指南中的下载 Python。

   运行以下命令验证 Python 3.6 或更新版本是否已添加到您的计算机路径中：

```
python --version
```

输出应显示 3.6 或更高版本的 Python。

2. 安装或升级Amazon SAM CLI

   要检查是否安装了Amazon SAM CLI，请运行以下命令：

   ```
   sam --version
   ```

   如果已经安装了Amazon SAM CLI，则输出将显示一个版本。要升级到最新版本，请参阅升级 CAmazon SAM LI (p. 443)。

   有关安装Amazon SAM CLI 及其所有先决条件的说明，请参阅安装 Amazon SAM CLI (p. 15)。

3. 安装 Terraform

   要检查是否安装了 Terraform，请运行以下命令：

   ```
   terraform -version
   ```

   要安装 Terraform，请参阅 Terraform 开发者文档中的安装 Terraform。

4. 安装 Docker 进行本地测试

   Amazon SAM命令行界面需要 Docker 进行本地测试。要安装 Docker，请参阅安装 Docker 以便与 Amazon SAM CLI 一起使用 (p. 455)。

5. 安装制作工具（仅限 Windows）

   Make是 Windows 的软件包管理器和安装程序。要使用 Chocolatey 进行安装，请参阅 Windows 如何安装和使用 “Make” 中的 “使用 Chocolat ey”。

后续步骤

现在，你已经准备好开始在你的 Terraform 项目中使用Amazon SAM CLI 了。要了解更多信息，请参阅：

- 最好搭配使用：Amazon SAMCLI 和HashiCorp Terraform — 关于在 Terraform 上使用Amazon SAM CLI 的教程。
- 使用带有 Terraform 的Amazon SAM CLI 进行本地调试和测试 (p. 390)— 关于在 Terraform 上使用 Amazon SAM CLI 的文档。

使用带有 Terraform 的Amazon SAM CLI 进行本地调试和测试

> Terraform 支持是Amazon SAM CLI 的预览版，可能会发生变化。要提供反馈和提交功能请求，请创建GitHub问题。

Amazon Serverless Application Model命令行界面 (Amazon SAMCLI) 支持对您的Amazon Lambda函数和层进行本地调试和测试。有关此功能的介绍，请参见Amazon SAMCLI 对 Terraform 的支持是什么？ (p. 388)

主题

- 本地调试和测试概述 (p. 391)

本地调试和测试概述

准备 Terraform 项目以进行本地调试和测试的一般过程包括：

1. 根据您的 Terraform 项目结构确定是否需要定义。sam metadata resource
2. 根据您的 Terraform 项目结构确定是否需要运行sam build。
3. 使用支持的Amazon SAM CLI 命令在本地调试和测试您的 Lambda 函数。

在本地调试和测试时，使用terraform plan和terraform apply来准备和部署您的更改。

相同的元数据资源

sam metadata resource是一种`null_resource` Terraform 资源类型，根据你的 Terraform 项目结构是可选的。要了解有关此资源类型的更多信息，请参阅 Terraform 文档中的 null_ resource。

为Amazon SAM CLIsam metadata resource 提供了在您的 Terraform 项目中查找 Lambda 函数和层及其源代码、构建依赖关系和构建逻辑所需的信息。如果您在 Terraform 项目之外构建 Lambda 资源工件，并将这些工件路径传递给您的 Terraform 项目，sam metadata resource则不是必需的。

或者，如果您使用 Terraform 变量定义您的 Lambda 工件，也不需要定义sam metadata resource。Amazon SAMCLI 将使用这些 Terraform 变量。有关 Terraform 变量的更多信息，请参阅 `TF_VAR_name`Terraform 的开发者文档

否则，如果您在 Terraform 项目中构建 Lambda 函数和层，sam metadata resource则每个函数和层都需要使用。

定义相同的元数据资源

定义您的时，请使用以下准则sam metadata resource：

- 将您的sam metadata resource开头命名`sam_metadata_`为，以Amazon SAM便 CLI 将其标识为sam metadata resource。
- 在`triggers`区块中定义您的属性。
- 使用`depends_on`参数定义您的 Lambda 函数或层构建逻辑。

这是一个基本的sam metadata resource模板结构：

```
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
  ]
}
```

您的详细信息sam metadata resource将因 Lambda 资源类型（函数或层）和打包类型（ZIP 或映像）而异。有关定义此资源的更多信息以及示例，请参阅相同的元数据资源 (p. 394)。

在 Terraform 上使用Amazon SAM命令行界面

使用支持的Amazon SAM CLI 命令时需要定义的主要属性是：

- `--hook-name`
- `--beta-features`

这些属性可以在运行Amazon SAM CLI 命令时从命令行定义，也可以在您的`samconfig.toml`文件中定义。添加以下内容以在`samconfig.toml`文件中定义这些属性：

```
version = 0.1
[default]
[default.build.parameters]
hook_name = "terraform"
beta_features = true
```

或者，在使用Amazon SAM CLI 和支持的命令时配置这些属性。有关从命令行定义这些属性的示例，请参见以下示例：

```
# Using sam build with Terraform
sam build --hook-name terraform --beta-features

# Using sam local invoke with Terraform
sam local invoke --hook-name terraform --beta-features

# Using sam local start-lambda with Terraform
sam local start-lambda --hook-name terraform --beta-features
```

如果您不允许使用测试版功能，Amazon SAMCLI 将在您定义`--hook-name`选项时提示您选择加入：

```

Supporting Terraform applications is a beta feature.
Please confirm if you would like to proceed using AWS SAM CLI with terraform application.
You can also enable this beta feature with 'sam local invoke --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/.
```

使用 sam build 构建你的 Terraform 项目

只有在定义了时才需要运行sam build以初始化您的 Terraform 项目以进行本地调试和测试sam metadata resource。否则sam build，不需要运行。

要进行构建sam build，请从 Terraform 项目根目录运行以下命令：

```
sam build --hook-name terraform --beta-features lambda-resource-id
```

或者，您可以提供要构建id的 Lambda 函数或层。接受的 ID 是 Lambda 函数名称或完整的 Terraform 资源地址，例如aws_lambda_function.list_books或module.list_book_function.aws_lambda_function.this[0]。

> **Note**
>
> 运行时sam build，您的sam metadata resourcedepends_on参数必须引用您的 Lambda 资源构建逻辑。

有关 sam build 的更多信息，请参阅山姆·布莱德 (p. 408)。

使用 sam 本地调用进行本地测试

> **Note**
>
> 要使用Amazon SAM CLI 在本地进行测试，必须安装和配置 Docker。有关说明，请参阅 安装 Docker 以便与Amazon SAM CLI 一起使用 (p. 455)。

以下是通过传入事件在本地测试 Lambda 函数的示例：

```
sam local invoke --hook-name terraform --beta-features hello_world_function -e events/
event.json -
```

有关 sam local invoke 的更多信息，请参阅sam 本地调用 (p. 423)。

使用 sam 本地启动 lambda 进行本地测试

以下是使用在本地测试 Lambda 函数的示例Amazon CLI：

```
# Start Lambda
sam local start-lambda --hook-name terraform --beta-features hello_world_function

# Test 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
```

有关 sam local start-lambda 的更多信息，请参阅sam local start-la (p. 427)。

不支持的 Terraform 功能

目前不支持以下 Terraform 功能：

- 链接到多个层的 Lambda 函数。
- 对局部值进行地形化处理。

使用带有 Serverless.tf 的Amazon SAM CLI 进行本地调试和测试

> Terraform 支持是Amazon SAM CLI 的预览版，可能会发生变化。要提供反馈和提交功能请求，请创建GitHub问题。

Amazon Serverless Application Model命令行界面 (Amazon SAMCLI) 可以与 Serverless.tf 模块一起使用，对您的Amazon Lambda函数和层进行本地调试和测试。支持以下Amazon SAM CLI 命令：

- sam local invoke
- sam local start-lambda

要开始在 Serverless.tf 模块中使用Amazon SAM CLI，请更新到最新版本的 Serverless.tf 和Amazon SAM CLI。

Note

Serverless.tf 版本 4.6.0 及更高版本，支持Amazon SAM CLI 集成。

要了解有关 Serverless.tf 的更多信息，请参阅 terraform-aws-lambda-module。

Amazon SAM带有 Terraform 参考的 CLI

Terraform 支持是Amazon SAM CLI 的预览版，可能会发生变化。要提供反馈和提交功能请求，请创建GitHub 问题。

本节是在 Terraform 中使用Amazon Serverless Application Model命令行界面 (Amazon SAMCLI) 进行本地调试和测试的参考。

Amazon SAM支持的功能参考

可以在此处找到支持与 Terraform 一起使用的Amazon SAM CLI 功能的参考文档：

- 山姆·布莱德 (p. 408)
- sam 本地调用 (p. 423)
- sam local start-la (p. 427)

Terraform 特定参考资料

可以在此处找到专门在 Terraform 中使用Amazon SAM CLI 的参考文档：

- 相同的元数据资源 (p. 394)

相同的元数据资源

Terraform 支持是Amazon SAM CLI 的预览版，可能会发生变化。要提供反馈和提交功能请求，请创建GitHub问题。

本页包含 Terraform 项目所用sam metadata resource资源类型的参考信息。

- 有关在 Terraform 上使用Amazon Serverless Application Model命令行界面 (Amazon SAMCLI) 的简介，请参阅Amazon SAMCLI 对 Terraform 的支持是什么？ (p. 388)。
- 要在 Terraform 上使用Amazon SAM CLI，请参阅使用带有 Terraform 的Amazon SAM CLI 进行本地调试和测试 (p. 390)。

主题

- Arguments (p. 395)
- 示例 (p. 395)

Arguments

参数	描述
built_output_path	通往Amazon Lambda函数内置工件的路径。
docker_build_args	Docker 构建参数 JSON 对象的解码字符串。此参数是可选的。
docker_context	包含 Docker 镜像构建上下文的目录路径。
docker_file	Docker 文件的路径。此路径与路docker_context径是相对的。 此参数是可选的。默认值为 Dockerfile。
docker_tag	创建的 Docker 镜像标签的值。该值为可选项。
depends_on	您的 Lambda 函数或层的建筑资源路径。要了解更多信息，请参阅 Terraform 文档中的depends_on论点。
original_source_code	定义 Lambda 函数的路径。此值可以是字符串、字符串数组或解码为字符串的 JSON 对象。 • 对于字符串数组，由于不支持多个代码路径，因此仅使用第一个值。 • 对于 JSON 对象，还source_code_property必须定义。
resource_name	Lambda 函数名称。
resource_type	您的 Lambda 函数包类型的格式。可接受的值是： • IMAGE_LAMBDA_FUNCTION • LAMBDA_LAYER • ZIP_LAMBDA_FUNCTION
source_code_property	JSON 对象中 Lambda 资源代码的路径。当original_source_code是 JSON 对象时定义此属性。

示例

使用 ZIP 软件包类型引用 Lambda 函数的元数据元数据资源

```
# 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
  ]
}
```

使用图像软件包类型引用 Lambda 函数的 sam 元数据资源

```
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"
  }
}
```

引用 Lambda 图层的垃圾元数据资源

```
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]
}
```

无服务器应用程序示例

以下示例向您展示了如何下载、测试和部署许多其他无服务器应用程序，包括如何配置事件源和Amazon资源。

主题

处理 DynamoDB 事件

使用此示例应用程序，您可以在概述和快速入门指南中学到的内容的基础上再安装另一个示例应用程序。此应用程序由一个 Lambda 函数组成，该函数由 DynamoDB 表事件源调用。Lambda 函数非常简单——它记录通过事件源消息传入的数据。

本练习向您展示如何模仿调用 Lambda 函数时传递给 Lambda 函数的事件源消息。

开始前的准备工作

确保您已完成中的所需设置安装 Amazon SAM CLI (p. 15)。

步骤 1：初始化应用程序

在本节中，您将下载应用程序包，它由Amazon SAM模板和应用程序代码组成。

初始化应用程序

1. 在 Amazon SAM CLI 命令提示符处运行以下命令。

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

   请注意，gh:在上面的命令中扩展为 GitHub 网址https://github.com/。
2. 查看命令创建的目录的内容 (dynamodb_event_reader/)：
   - template.yaml— 定义了 Read DynamoDB 应用程序需要的两个Amazon资源：一个 Lambda 函数和一个 DynamoDB 表。模板还定义了两个资源之间的映射。
   - read_dynamodb_event/目录 — 包含 DynamoDB 应用程序代码。

步骤 2：在本地测试应用程序

要进行本地测试，请使用Amazon SAM CLI 生成示例 DynamoDB 事件并调用 Lambda 函数：

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

该generate-event命令会创建测试事件源消息，例如将所有组件部署到Amazon云端时创建的消息。此事件源消息通过管道传送到 Lambda 函数 ReadDynamo dbeVent。

根据中的源代码，验证预期的消息是否已打印到控制台app.py。

步骤 3：Package 应用程序

在本地测试应用程序后，您可以使用Amazon SAM CLI 创建部署包，用于将应用程序部署到Amazon云端。

创建 Lambda 部署程序包

1. 在要保存打包代码的位置创建 S3 存储桶。如果要使用现有 S3 存储桶，请跳过此步骤。

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

2. 通过在命令提示符处运行以下package CLI 命令来创建部署程序包。

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

在下一步中部署应用程序时packaged.yaml，您可以指定新的模板文件。

步骤 4：部署应用程序

现在您已经创建了部署包，您可以使用它将应用程序部署到Amazon云端。然后，您测试应用程序。

将无服务器应用程序部署到Amazon云端

- 在Amazon SAM CLI 中，使用deploy CLI 命令部署您在模板中定义的所有资源。

```
sam deploy \
    --template-file packaged.yaml \
    --stack-name sam-app \
    --capabilities CAPABILITY_IAM \
    --region us-east-1
```

在命令中，--capabilities参数Amazon CloudFormation允许创建 IAM 角色。

Amazon CloudFormation创建模板中定义Amazon的资源。您可以在Amazon CloudFormation控制台中访问这些资源的名称。

在Amazon云端测试无服务器应用程序

1. 打开 DynamoDB 控制台。
2. 将记录插入刚创建的表中。
3. 转到表格的 “指标” 选项卡，然后选择 “查看所有 CloudWatch 指标”。在 CloudWatch 控制台中，选择 Logs 以查看日志输出。

后续步骤

该Amazon SAM GitHub 存储库包含其他示例应用程序供您下载和试用。要访问此存储库，请参阅Amazon SAM示例应用程序。

处理 Amazon S3 事件

使用此示例应用程序，您可以在前面示例中学到的内容的基础上再接再厉，安装更复杂的应用程序。此应用程序由一个 Lambda 函数组成，该函数由 Amazon S3 对象上传事件源调用。本练习向您展示如何通过 Lambda 函数访问Amazon资源和调用Amazon服务。

此示例无服务器应用程序在 Amazon S3 中处理对象创建事件。对于上传到存储桶的每张图片，Amazon S3 都会检测对象创建事件并调用 Lambda 函数。Lambda 函数调用 Amazon Rekognition 来检测图像中的文本。然后，它将Amazon Rekognition 返回的结果存储在 DynamoDB 表中。

> Note
>
> 在此示例应用程序中，您执行步骤的顺序与前面的示例略有不同。其原因是，此示例要求您先创建Amazon资源并配置 IAM 权限，然后才能在本地测试 Lambda 函数。我们将利用它Amazon CloudFormation来创建资源并为您配置权限。否则，您需要手动执行此操作，然后才能在本地测试 Lambda 函数。
> 由于此示例更为复杂，因此在执行此示例之前，请确保您熟悉安装前面的示例应用程序。

开始前的准备工作

确保您已在中完成所需的设置安装 Amazon SAM CLI (p. 15)。

步骤 1：初始化应用程序

在本节中，您将下载示例应用程序，其中包括 Amazon SAM 模板和应用程序代码。

初始化应用程序

1. 在 Amazon SAM CLI 命令提示符处运行以下命令。

```
sam init \
--location https://github.com/aws-samples/cookiecutter-aws-sam-s3-rekognition-dynamodb-python \
--no-input
```

2. 查看命令创建的目录的内容 (`aws_sam_ocr/`)：

   - `template.yaml`— 定义 Amazon S3 应用程序需要的三种Amazon资源：一个 Lambda 函数、一个 Amazon S3 存储桶和 DynamoDB 表。该模板还定义了这些资源之间的映射和权限。
   - `src/`目录 — 包含 Amazon S3 应用程序代码。
   - `SampleEvent.json`— 示例事件源，用于本地测试。

步骤 2：Package 应用程序

在本地测试此应用程序之前，必须使用Amazon SAM CLI 创建部署包，使用该部署包将应用程序部署到Amazon云端。此部署创建了在本地测试应用程序所需的必要Amazon资源和权限。

创建 Lambda 部署程序包

1. 在要保存打包代码的位置创建 S3 存储桶。如果要使用现有 S3 存储桶，请跳过此步骤。

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

2. 在命令提示符处运行以下`package` CLI 命令来创建部署程序包。

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

在下一步中部署应用程序时packaged.yaml，您可以指定新的模板文件。

步骤 3：部署应用程序

现在您已经创建了部署包，您可以使用它来将应用程序部署到Amazon云端。然后，您可以通过在Amazon云端调用该应用程序来对其进行测试。

将无服务器应用程序部署到Amazon云端

- 在Amazon SAM CLI 中，使用deploy命令部署您在模板中定义的所有资源。

```
sam deploy \
    --template-file packaged.yaml \
    --stack-name aws-sam-ocr \
    --capabilities CAPABILITY_IAM \
    --region us-east-1
```

在命令中，--capabilities参数Amazon CloudFormation允许创建 IAM 角色。

Amazon CloudFormation创建在模板中定义Amazon的资源。您可以在Amazon CloudFormation控制台中访问这些资源的名称。

在Amazon云端测试无服务器应用程序

1. 将图像上传到您为该示例应用程序创建的 Amazon S3 存储桶。
2. 打开 DynamoDB 控制台并找到已创建的表。有关Amazon Rekognition 返回的结果，请参阅下表。
3. 验证 DynamoDB 表中是否包含包含 Amazon Rekognition 在上传的图像中找到的文本的新记录。

步骤 4：在本地测试应用程序

必须先检索由创建的Amazon资源的名称，然后才能在本地测试应用程序Amazon CloudFormation。

- 从中检索 Amazon S3 密钥名称和存储桶名称Amazon CloudFormation。通过替换对象密钥、存储桶名称和存储桶 ARN 的值来修改SampleEvent.json文件。
- 检索 DynamoDB 表名称。此名称用于以下sam local invoke命令。

使用Amazon SAM CLI 生成示例 Amazon S3 事件并调用 Lambda 函数：

```
TABLE_NAME=Table name obtained from Amazon CloudFormation console sam local invoke --event
 SampleEvent.json
```

该TABLE_NAME=部分用于设置 DynamoDB 表的名称。该--event参数指定了包含要传递给 Lambda 函数的测试事件消息的文件。

现在，您可以根据 Amazon Rekognition 返回的结果验证预期的 DynamoDB 记录是否已创建。

后续步骤

该Amazon SAM GitHub 存储库包含其他示例应用程序供您下载和试用。要访问此存储库，请参阅Amazon SAM示例应用程序。

Amazon Cloud Development Kit (Amazon CDK)

您可以使用Amazon SAM CLI 在本地测试和构建使用定义的无服务器应用程序Amazon Cloud Development Kit (Amazon CDK)。由于Amazon SAM CLI 在Amazon CDK项目结构中运行，因此您仍然可以使用 Amazon CDKToolkit 创建、修改和部署Amazon CDK应用程序。

有关安装和配置的信息Amazon CDK，请参阅《Amazon Cloud Development Kit (Amazon CDK)开发人员指南》Amazon CDK中的入门。

> Note
>
> CAmazon SAM LI 支持从 1.135.0 版本开始的Amazon CDKAmazon CDK v1 和从版本 2.0.0 开始的 v2。

主题

入门Amazon SAM和Amazon CDK

本主题介绍在Amazon CDK应用程序中使用Amazon SAM CLI 所需的内容，并提供有关构建和本地测试简单Amazon CDK应用程序的说明。

先决条件

要将Amazon SAM CLI 与一起使用Amazon CDK，必须安装Amazon CDK、和Amazon SAM CLI。

- 有关安装的信息Amazon CDK，请参阅《Amazon Cloud Development Kit (Amazon CDK)开发者指南》Amazon CDK中的入门指南。
- 有关安装Amazon SAM CLI 的信息，请参阅安装 Amazon SAM CLI (p. 15)。

创建和本地测试Amazon CDK应用程序

要使用Amazon SAM CLI 在本地测试Amazon CDK应用程序，您必须拥有一个包含 Lambda 函数的Amazon CDK应用程序。使用以下步骤创建带有 Lambda 函数的基本Amazon CDK应用程序。有关更多信息，请参阅Amazon Cloud Development Kit (Amazon CDK)开发人员指南Amazon CDK中的使用创建无服务器应用程序。

> Note
>
> CAmazon SAM LI 支持从 1.135.0 版本开始的Amazon CDKAmazon CDK v1 和从版本 2.0.0 开始的 v2。

第 1 步：创建 Amazon CDK 应用程序

在本教程中，初始化一个使用的Amazon CDK应用程序 TypeScript。

要运行的命令：

Amazon CDK v2

```
mkdir cdk-sam-example
cd cdk-sam-example
cdk init app --language typescript
```

Amazon CDK v1

```
mkdir cdk-sam-example
cd cdk-sam-example
cdk init app --language typescript
npm install @aws-cdk/aws-lambda
```

步骤 2：添加 Lambda 函数到应用程序

将中的`lib/cdk-sam-example-stack.ts`代码替换为以下代码：

Amazon 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_7,
      handler: 'app.lambda_handler',
      code: lambda.Code.fromAsset('./my_function'),
    });
  }
}
```

Amazon 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_7,
      handler: 'app.lambda_handler',
      code: lambda.Code.fromAsset('./my_function'),
    });
  }
}
```

步骤 3：添加 Lambda 函数代码

创建名为 `my_function` 的目录。在此目录中，创建名为 `app.py` 的文件。

要运行的命令：

```
mkdir my_function
cd my_function
touch app.py
```

将以下代码添加到 app.py：

```
def lambda_handler(event, context):
    return "Hello from SAM and the CDK!"
```

步骤 4：测试 Lambda 函数

您可以使用Amazon SAM CLI 在本地调用您在Amazon CDK应用程序中定义的 Lambda 函数。为此，您需要函数构造标识符和合成Amazon CloudFormation模板的路径。

要运行的命令：

```
cdk synth --no-staging
```

```
sam local invoke MyFunction --no-event -t ./cdk.out/CdkSamExampleStack.template.json
```

输出示例：

```
Invoking app.lambda_handler (python3.7)

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
```

有关使用Amazon SAM CLI 测试Amazon CDK应用程序的可用选项的更多信息，请参阅本地测试Amazon CDK应用程序 (p. 404)。

本地测试Amazon CDK应用程序

您可以使用Amazon SAMCLI 在本地测试Amazon CDK应用程序通过从您的项目根目录运行以下命令Amazon CDK应用程序：

- sam 本地调用 (p. 423)
- sam 本地启动 api (p. 425)
- sam local start-la (p. 427)

在运行任何sam local命令带Amazon CDK应用程序，你必须运行`cdk synth`.

运行时sam local invoke您需要您希望调用的函数构造标识符，以及综合的路径Amazon CloudFormationTemplate。如果您的应用程序使用嵌套堆栈，为了解决命名冲突，还需要定义函数的堆栈名称。

使用方法：

```
# 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 Amazon CDK application
sam local start-api -t ./cdk.out/CdkSamExampleStack.template.json [OPTIONS]

# Start a local endpoint that emulates Amazon Lambda
sam local start-lambda -t ./cdk.out/CdkSamExampleStack.template.json [OPTIONS]
```

示例

考虑使用以下示例声明的堆栈和函数：

```
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) {
    ...
    new lambda.Function(this, 'MyFunction', {
        ...
    });
    new lambda.Function(this, 'MyNestedFunction', {
        ...
    });
}
```

以下命令在本地调用上述示例中定义的 Lambda 函数：

```
# 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
```

构建Amazon CDK应用程序

这些区域有：Amazon SAMCLI 提供了对构建在您的中定义的 Lambda 函数和层的支持Amazon CDK应用程序山姆·布莱德 (p. 408).

对于使用 zip 工件的 Lambda 函数，请运行`cdk synth`在你跑之前`sam local`命令。`sam build`不是必需项。

如果您的Amazon CDK应用程序使用具有映像类型的函数，运行`cdk synth`然后运行`sam build`在你跑之前`sam local`命令。当你跑`sam build`、Amazon SAM例如，不构建使用运行时特定结构的 Lambda 函数或图层，nodeJS函数.`sam build`不支持捆绑资产.

示例

从运行以下命令Amazon CDK项目根目录构建应用程序。

```
sam build -t ./cdk.out/CdkSamExampleStack.template.json
```

部署Amazon CDK应用程序

这些区域有：Amazon SAMCLI 不支持部署Amazon CDK应用程序。使用`cdk deploy`部署您的应用程序。有关更多信息，请参阅 。Amazon CDK工具包（cdk 命令）中的Amazon Cloud Development Kit (Amazon CDK)开发人员指南

Amazon SAM 引用

Amazon SAM规格

该Amazon SAM规范是 Apache 2.0 许可证下的开源规范。该Amazon SAM规范的当前版本可在中找到Amazon Serverless Application Model(Amazon SAM) 规格 (p. 77)。

Amazon SAM模板是Amazon CloudFormation模板的扩展。有关Amazon CloudFormation模板的完整参考，请参阅《Amazon CloudFormation用户指南》中的模板参考。

Amazon SAMCLI 命令参考

Amazon Serverless Application Model命令行接口 (Amazon SAMCLI) 是一种命令行工具，您可以将其与Amazon SAM模板和支持的第三方集成一起使用，以构建和运行您的无服务器应用程序。

您可以使用Amazon SAM CLI 命令开发、测试您的无服务器应用程序并将其部署到Amazon Web Services 云。以下是Amazon SAM CLI 命令的一些示例：

- `sam init`— 如果您是首次Amazon SAM使用 CLI 的用户，则可以在不使用任何参数的情况下运行`sam init`命令来创建 Hello World 应用程序。该命令使用您选择的语言生成预配置的Amazon SAM模板和示例应用程序代码。
- `sam local invoke`和`sam local start-api` — 使用这些命令在本地测试应用程序代码，然后再将其部署到Amazon Web Services 云。
- `sam logs`— 使用此命令提取您的 Lambda 函数生成的日志。在将应用程序部署到之后，这可以帮助你测试和调试应用程序Amazon Web Services 云。
- `sam package`— 使用此命令将应用程序代码和依赖项捆绑到部署包中。您需要部署包才能将应用程序上传到Amazon Web Services 云。
- `sam deploy`— 使用此命令将您的无服务器应用程序部署到Amazon Web Services 云。它创建Amazon资源并设置Amazon SAM模板中定义的权限和其他配置。

有关安装Amazon SAM CLI 的说明，请参阅安装 Amazon SAM CLI (p. 15)。

Amazon SAM策略模板

借Amazon SAM助，您可以从策略模板列表中进行选择，将Amazon Lambda函数的权限范围限定为应用程序使用的资源。

主题

- Amazon SAM策略模板 (p. 278)
- 镜像 (p. 458)
- 中的遥测Amazon SAMCLI (p. 462)
- 管理访问权限 (p. 272)

Amazon SAM CLI 参考

本部分是Amazon Serverless Application Model命令行界面 (Amazon SAMCLI) 的参考资料。

主题

- Amazon SAMCLI 命令参考 (p. 408)
- Amazon SAM CLI 配置文件 (p. 441)
- 管理Amazon SAM CLI 版本 (p. 443)
- 设置Amazon证书 (p. 448)
- Amazon SAMCLI 纠正 (p. 449)

Amazon SAMCLI 命令参考

本节是Amazon SAM CLI 命令的参考资料。有关安装Amazon SAM CLI 的说明，请参阅安装 Amazon SAM CLI (p. 15)。

主题

- 山姆·布莱德 (p. 408)
- sam delete (p. 411)
- sam deploy (p. 412)
- sam init (p. 414)
- 同样的名单 (p. 416)
- sam 本地生成事件 (p. 421)
- sam 本地调用 (p. 423)
- sam 本地启动 api (p. 425)
- sam local start-la (p. 427)
- sam log (p. 430)
- sam package (p. 431)
- sam 管道引导 (p. 433)
- sam 管道 init (p. 435)
- sam publish (p. 436)
- 山姆同步 (p. 436)
- sam trace (p. 439)
- sam validate (p. 440)

山姆·布莱德

Amazon Serverless Application Model命令行界面 (Amazon SAMCLI)`sam build` 命令的选项。

- 有关Amazon SAM CLI 的简介，请参阅什么是Amazon SAM CLI？ (p. 6)。
- 有关使用Amazon SAM CLI`sam build` 命令的文档，请参阅使用 sam build (p. 41)。

用量

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

Arguments

参数	描述
RESOURCE_LOGICAL_ID	可选。指示Amazon SAM构建Amazon SAM模板中声明的单个资源。指定资源的构建工件将是工作流程中唯一可用于后续命令的构件，即sam package和sam deploy。

选项

选项	描述
--hook-name TEXT	用于扩展Amazon SAM CLI 功能的挂钩的名称。 可接受的值:terraform.
--skip-prepare-infra	如果未对基础架构进行任何更改，则跳过准备阶段。与--hook-name选项一起使用。
-b, --build-dir DIRECTORY	存储已建构件的目录路径。使用此选项可删除此目录及其所有内容。
-s, --base-dir DIRECTORY	解析相对于该目录的函数或图层源代码的相对路径。如果要更改解析源代码文件夹相对路径的方式，请使用此选项。默认情况下，相对路径是相对于Amazon SAM模板的位置进行解析的。 除了您正在构建的根应用程序或堆栈中的资源外，此选项还应用嵌套应用程序或堆栈。 此选项适用于以下资源类型和属性： • 资源类型：AWS::Serverless::Function属性：CodeUri • 资源类型：AWS::Serverless::Function资源属性：Metadata条目：DockerContext • 资源类型：AWS::Serverless::LayerVersion属性：ContentUri • 资源类型：AWS::Lambda::Function属性：Code • 资源类型：AWS::Lambda::LayerVersion属性：Content
-u, --use-container	如果您的函数依赖于具有原生编译依赖项的包，请使用此选项在类似 Lambda 的 Docker 容器中构建您的函数。
-e, --container-env-var TEXT	要传递到构建容器的环境变量。您可以多次指定该选项。此选项的每个实例都采用一个键值对，其中键是资源和环境变量，值是环境变量的值。例如：--container-env-var Function1.GITHUB_TOKEN=TOKEN1 --container-env-var Function2.GITHUB_TOKEN=TOKEN2。 此选项仅在指定该--use-container选项时适用，否则会导致错误。
-ef, --container-env-var-file PATH	包含容器环境变量值的 JSON 文件的路径和文件名。有关容器环境变量文件的更多信息，请参阅容器环境变量文件 (p. 344)。

选项	描述
	此选项仅在指定该--use-container选项时适用，否则会导致错误。
--build-image TEXT	您要下拉最新镜像获取 Builder。默认情况下，从 Amazon ECR Public 中 Amazon SAM拉取容器镜像。请使用此选项从其他位置拉取镜像。 您可以多次指定该选项。此选项的每个实例都可以采用字符串或键值对。如果您指定字符串，则它是用于应用程序中所有资源的容器映像的 URI。例如，sam build --use-container --build-image amazon/aws-sam-cli-build-image-python3.8。如果您指定键值对，则密钥是资源名称，值是用于该资源的容器映像的 URI。例如 sam build --use-container --build-image Function1=amazon/aws-sam-cli-build-image-python3.8。使用键值对，您可以为不同的资源指定不同的容器镜像。 此选项仅在指定该--use-container选项时适用，否则会导致错误。
-m, --manifest PATH	要使用的自定义依赖项清单文件（例如 package.json）的路径，而不是默认路径。
-t, --template-file, --template PATH	Amazon SAM模板文件的路径和文件名[default: template.[yaml|yml]]。此选项与不兼容--hook-name。
--parameter-overrides	（可选）包含Amazon CloudFormation编码为键值对的字符串。使用与Amazon Command Line Interface (Amazon CLI) 相同的格式。例如：'ParameterKey=KeyPairName、=、ParameterValueMyKeyParameterKey =InstanceType、ParameterValue =t1.micro'。此选项与不兼容--hook-name。
--skip-pull-image	指定命令是否应跳过下拉最新 Docker 镜像获取 Lambda 运行时的操作。
--docker-network TEXT	指定 Lambda Docker 容器应连接到的现有 Docker 网络的名称或 ID，以及默认的桥接网络。如果未指定此项，Lambda 容器将仅连接到默认的桥接 Docker 网络。
--beta-features | --no-beta-features	允许或拒绝测试版功能。
--parallel	启用parallel 构建。使用此选项可parallel 构建Amazon SAM模板的功能和层。默认情况下，函数和层是按顺序构建的。
--cached | --no-cached	启用或禁用缓存构建。使用此选项可重用与先前版本相比没有更改的编译工件。 Amazon SAM评估您是否更改了项目目录中的任何文件。默认情况下，编译版本不会被缓存。如果调用该--no-cached选项，它将覆盖 samcofig.toml 中的cached = true设置。注意：Amazon SAM不评估您是否更改了项目所依赖的第三方模块，而您没有提供特定版本。例如，如果您的 Python 函数包含带有条目的requirements.txtrequests=1.x文件，并且最新的请求模块版本从1.1更改为，则在运行非缓存版本之前 Amazon SAM不会提取最新版本。1.2
--cache-dir	指定时存储缓存构件--cached的目录。默认的缓存目录是.aws-sam/cache。
-x, --exclude	要从 SAM CLI 版本中排除的资源的名称。例如，如果您的模板包含Function1Function2、和，Function3并且您运行sam build --exclude Function2，则只Function3会生成Function1和。
--profile TEXT	您的凭证文件中用于获取Amazon证书的特定配置文件。

选项	描述
`--region TEXT`	要部署到的Amazon区域。例如，us-east-1。
`--config-file PATH`	包含要使用的默认参数值的配置文件的路径和文件名。项目目录根目录中的默认值为`samconfig.toml` “”。有关配置文件的详细信息，请参阅 Amazon SAM CLI 配置文件 (p. 441)。
`--config-env TEXT`	环境名称，用于指定要使用的配置文件中的默认参数值。默认值为 “默认”。有关配置文件的详细信息，请参阅 Amazon SAM CLI 配置文件 (p. 441)。
`--debug`	打开调试日志记录以打印Amazon SAM CLI 生成的调试消息并显示时间戳。
`--help`	显示此消息并退出。

sam delete

删除Amazon SAM通过删除应用程序Amazon CloudFormation堆栈、打包并部署到 Amazon S3 和 Amazon ECR 的工件，以及Amazon SAM模板文件。

还要检查是否部署了 Amazon ECR 伴侣堆栈，如果是，则提示用户删除该堆栈和 Amazon ECR 存储库。如果`--no-prompts`已指定，然后默认情况下将删除配套堆栈和 Amazon ECR 存储库。

使用方法：

```
sam delete [OPTIONS]
```

选项：

选项	描述
`--stack-name TEXT`	的名称Amazon CloudFormation堆栈要删除。
`--no-prompts`	将此选项指定为Amazon SAM在非交互模式下运行。必须提供堆栈名称，无论是`--stack-name`选项，或者在配置中`toml`文件。
`--region TEXT`	这些区域有：Amazon要部署到的区域。例如，us-east-1。
`--profile TEXT`	获取凭证文件中的特定配置文件Amazon凭证。
`--config-file PATH`	包含要使用的默认参数值的配置文件的路径和文件名。默认值是 。`samconfig.toml`在项目目录的根目录中。有关配置文件的详细信息，请参阅 Amazon SAM CLI 配置文件 (p. 441)。
`--config-env TEXT`	指定配置文件中要使用的默认参数值的环境名称。原设定值为 `default`。有关配置文件的详细信息，请参阅 Amazon SAM CLI 配置文件 (p. 441)。
`--s3-bucket`	要删除的 Amazon S3 存储桶的路径。
`--s3-prefix`	要删除的 Amazon S3 存储桶的前缀。
`--debug`	打开调试日志记录以打印调试消息Amazon SAMCLI 生成和显示时间戳。
`--help`	显示此消息并退出。

sam deploy

Amazon Serverless Application Model命令行界面 (Amazon SAMCLI)`sam deploy` 命令的选项。

- 有关Amazon SAM CLI 的简介，请参阅什么是Amazon SAM CLI？ (p. 6)。
- 有关使用Amazon SAM CLI`sam deploy` 命令的文档，请参阅使用 sam deploy (p. 50)。

用量

```
$ sam deploy <options>
```

环境变量

环境变量	描述
SAM_CLI_POLL_DELAY	指定DescribeStack API 调用之间的延迟（以秒为单位）。 以下是示例： `$ SAM_CLI_POLL_DELAY=5 sam deploy`

选项

选项	描述
-g, --guided	指定此选项可让Amazon SAM CLI 使用提示来指导您完成部署。
-t, --template-file, --template PATH	您的Amazon SAM模板所在的路径和文件名。 注意：如果您指定此选项，则仅Amazon SAM部署模板及其指向的本地资源。
--stack-name TEXT	（必选）您要部署到的Amazon CloudFormation堆栈的名称。如果指定现有堆栈，则该命令将更新堆栈。如果指定新堆栈，则该命令将创建它。
--s3-bucket TEXT	此命令用于上传您的 Amazon S3 桶的Amazon CloudFormation名称。如果您的模板大于 51,200 字节，则该--s3-bucket选项或--resolve-s3选项为必填项。如果您同时指定--s3-bucket和--resolve-s3选项，则会出现错误。
--s3-prefix TEXT	添加到 Amazon S3 桶的对象名称的前缀。前缀名称是 Amazon S3 桶的路径名（文件夹名称）。
--image-repository TEXT	此命令用于上传您的函数映像的 Amazon ECR 存储库的名称。使用Image包类型声明的函数需要此选项。
--signing-profiles LIST	用于签署部署包的签名配置文件列表。此选项采用键值对列表，其中密钥是要签名的函数或层的名称，值是签名配置文件，用可选的配置文件所有者分隔:。例如，FunctionNameToSign=SigningProfileName1 LayerNameToSign=SigningProfileName2:SigningProfileOwner。
--capabilities LIST	必须指定才能创建特定堆栈Amazon CloudFormation的功能列表。一些堆栈模板可能包含可影响您的中权限的资源Amazon Web Services 账户，例如，通过创建新的Amazon Identity and Access Management（IAM）

选项	描述
	用户。对于这些堆栈，必须通过指定此选项来明确确认其功能。有效值仅为 CAPABILITY_IAM 和 CAPABILITY_NAMED_IAM。如果包含 IAM 资源，则可以指定任意一个功能。如果包含具有自定义名称的 IAM 资源，则必须指定CAPABILITY_NAMED_IAM。如果不指定此选项，则该操作将返回InsufficientCapabilities错误。
--region TEXT	Amazon Web Services 区域要部署到的。例如，us-east-1。
--profile TEXT	您的凭证文件中获取Amazon证书的特定配置文件。
--kms-key-id TEXT	用于加密 Amazon S3 存储桶中静态对象的Amazon Key Management Service (Amazon KMS) 密钥的 ID。如果不指定此选项，则Amazon SAM使用 Amazon S3 托管式加密密钥。
--force-upload	指定此选项可上传构件，即使它们与 Amazon S3 存储段中的现有构件相匹配。匹配的伪像会被覆盖。
--no-execute-changeset	表示是否应用变更集。如果您想在应用变更集之前查看堆栈更改，请指定此选项。此命令创建一个Amazon CloudFormation变更集，然后在不应用变更集的情况下退出。要应用变更集，请运行不带此选项的相同命令。
--role-arn TEXT	应用变更集时的 IAmazon ResourcAmazon CloudFormation e Name (ARN)。
--fail-on-empty-changeset | --no-fail-on-empty-changeset	指定在堆栈无需更改时是否返回非零退出代码。默认行为是返回非零的退出代码。
--confirm-changeset | --no-confirm-changeset	提示确认Amazon SAM CLI 是否部署了计算出的变更集。
--use-json	为Amazon CloudFormation模板输出 JSON。默认输出是 YAML。
--resolve-s3	自动创建 Amazon S3 存储桶，用于打包和部署非引导式部署。如果您指定了该--guided选项，则Amazon SAM CLI 会忽略--resolve-s3。如果您同时指定--s3-bucket和--resolve-s3选项，则会出现错误。
--resolve-image-repos	自动创建 Amazon ECR 存储库，用于打包和部署非引导式部署。此选项仅适用于具有PackageType: Image指定的函数和图层。如果您指定了该--guided选项，则Amazon SAM CLI 会忽略--resolve-image-repos。注意：如果使用此选项Amazon SAM自动为函数或层创建任何 Amazon ECR 存储库，并且您随后从Amazon SAM模板中删除了这些函数或层，则相应的 Amazon ECR 存储库将自动删除。
--metadata	用于附加到模板中引用的所有对象的元数据地图。
--notification-arns LIST	与堆栈Amazon CloudFormation关联的 Amazon Snple Notifple (Amazon SNS) 主题 ARN。
--tags LIST	要与创建或更新的堆栈关联的标签列表。 Amazon CloudFormation还可以将这些标签传播到堆栈中支持它的资源。
--parameter-overrides	包含编码为键/值对的Amazon CloudFormation参数覆盖的字符串。使用与Amazon Command Line Interface (Amazon CLI) 相同的格式。例如，ParameterKey=ParameterValue InstanceType=t1.micro。

选项	描述
`--disable-rollback | --no-disable-rollback`	指定在部署期间出现错误时是否回滚Amazon CloudFormation堆栈。默认情况下，如果部署期间出现错误，您的Amazon CloudFormation堆栈会回滚到上一个稳定状态。如果您指定`--disable-rollback`并在部署期间出现错误，则不会回退在错误发生之前创建或更新的资源。
`--on-failure [ROLLBACK | DELETE | DO_NOTHING]`	指定堆栈创建失败时要采取的操作。 以下选项可用： • ROLLBACK— 将堆栈回滚到先前的已知良好状态。 • DELETE— 将堆栈回滚到先前的已知良好状态（如果存在）。否则，删除堆栈。 • DO_NOTHING— 既不回滚也不删除堆栈。效果与的相同`--disable-rollback`。 默认行为是 ROLLBACK。 注意：您可以指定`--disable-rollback`选项或`--on-failure`选项，但不能同时指定二者。
`--config-file PATH`	包含要使用的默认参数值的配置文件的路径和文件名。默认值`samconfig.toml`在项目目录的根目录中。有关配置文件的详细信息，请参阅 Amazon SAM CLI 配置文件 (p. 441)。
`--config-env TEXT`	环境名称，用于指定要使用的配置文件中的默认参数值。默认值为`default`。有关配置文件的详细信息，请参阅 Amazon SAM CLI 配置文件 (p. 441)。
`--no-progressbar`	将构件上传到 Amazon S3 时不要显示进度条。
`--debug`	开启调试日志记录以打印Amazon SAM CLI 生成的调试消息并显示时间戳。
`--help`	显示此消息并退出。

sam init

Amazon Serverless Application Model命令行界面 (Amazon SAMCLI)`sam init` 命令的选项。

- 有关Amazon SAM CLI 的简介，请参阅什么是Amazon SAM CLI？ (p. 6)。
- 有关使用Amazon SAM CLI`sam init` 命令的文档，请参阅使用 sam init (p. 63)。

用量

```
$ sam init <options>
```

选项

选项	描述
`-a, --architecture [x86_64 | arm64]`	您的应用程序 Lambda 函数的指令集架构。指定`x86_64`或之一`arm64`。

选项	描述
`--app-template TEXT`	您要使用的托管应用程序模板的标识符。如果您不确定，请在`sam init`没有选项的情况下致电交互式工作流程。 如果已指定但未提供此参数`--no-interactive`，则此参数`--location`为必填参数。 此参数仅在Amazon SAM CLI 版本 0.30.0 及更高版本中可用。使用较早版本指定此参数会导致错误。
`--application-insights | --no-application-insights`	为您的 CloudWatch 应用程序激活亚马逊应用程序见解监控。要了解更多信息，请参阅 使用应用程序洞察监控您的无服务器 CloudWatch 应用程序 (p. 377)。 默认选项是 `--no-application-insights`。
`--base-image [amazon/nodejs18.x-base | amazon/nodejs16.x-base | amazon/nodejs14.x-base | amazon/nodejs12.x-base | amazon/python3.10-base | amazon/python3.9-base | amazon/python3.8-base | amazon/python3.7-base | amazon/ruby2.7-base | amazon/go1.x-base | amazon/java11-base | amazon/java8.al2-base | amazon/java8-base | amazon/dotnet6-base | amazon/dotnet5.0-base | amazon/dotnetcore3.1-base]`	您的应用程序的基本映像。此选项仅在包裹类型为时适用`Image`。 如果已指定、`--package-type`指定`--no-interactive`为且未指定此参数`Image`，则此参数`--location`为必填参数。
`--config-file PATH`	包含要使用的默认参数值的配置文件的路径和文件名。项目目录根目录中的默认值为 “samconfig.toml”。有关配置文件的详细信息，请参阅 Amazon SAM CLI 配置文件 (p. 441)。
`--config-env TEXT`	环境名称，用于指定要使用的配置文件中的默认参数值。默认值为 “default" (默认) 。有关配置文件的详细信息，请参阅 Amazon SAM CLI 配置文件 (p. 441)。
`-d, --dependency-manager [gradle | mod | maven | bundler | npm | cli-package | pip]`	您的 Lambda 运行时的依赖关系管理器。
`--debug`	打开调试日志记录以打印Amazon SAM CLI 生成的调试消息并显示时间戳。

选项	描述
`--extra-content`	覆盖模板`cookiecutter.json`配置中的任何自定义参数，例如，`{"customParam1": "customValue1", "customParam2":"customValue2"}`
`-h, --help`	显示此消息并退出。
`-l, --location TEXT`	模板或应用程序位置（Git、Mercurial、HTTP/HTTPS、.zip 文件、路径）。 如果`--no-interactive`指定了和`--runtime`、和，`--name`则此参数为`--app-template`必填参数。 对于 Git 存储库，必须使用仓库根目录的位置。 对于本地路径，模板必须采用.zip 文件或 Cookiecutter 格式。
`-n, --name TEXT`	要作为目录生成的项目的名称。 如果已指定但未提供此参数`--no-interactive`，则此参数`--location`为必填参数。
`--no-input`	禁用 Cookiecutter 提示并接受模板配置中定义的 vcfdefault 值。
`--no-interactive`	禁用初始化参数的交互式提示，如果缺少任何必需的值，则失败。
`-o, --output-dir PATH`	输出初始化应用程序的位置。
`--package-type [Zip | Image]`	示例应用程序的包类型。 `Zip`创建.zip 文件存档并`Image`创建容器映像。
`-r, --runtime [ruby2.7 | java8 | java8.al2 | java11 | nodejs12.x | nodejs14.x | nodejs16.x | nodejs18.x | dotnet6 | dotnet5.0 | dotnetcore3.1 | python3.10 | python3.9 | python3.8 | python3.7 | go1.x]`	您的应用程序的 Lambda 运行时间。此选项仅在包裹类型为时适用`Zip`。 如果已指定、`--package-type`指定`--no-interactive`为且未指定此参数`Zip`，则此参数`--location`为必填参数。
`--tracing | --no-tracing`	激活对您的 Lambda 函数的Amazon X-Ray跟踪。

同样的名单

输出有关无服务器应用程序中的资源和无服务器应用程序状态的重要信息。在部署sam list前后使用，为本地和云开发提供帮助。

用量

```
sam list <option> <command> ...
```

选项

选项	描述
`-h, --help`	显示此消息并退出。

命令

命令	描述
`endpoints`	显示堆Amazon CloudFormation栈中的云端和本地端点列表。有关更多信息，请参阅终端节点列表 (p. 417)：
`resources`	显示您的Amazon Serverless Application Model (Amazon SAM) 模板中在部署Amazon CloudFormation时创建的资源。有关更多信息，请参阅垃圾列表资源 (p. 418)：
`stack-outputs`	显示来自Amazon SAM或Amazon CloudFormation模板的Amazon CloudFormation堆栈输出。有关更多信息，请参阅 相同列表堆栈输出 (p. 420)。

终端节点列表

显示堆Amazon CloudFormation栈中的云端和本地端点列表。您可以通过sam local和sam sync命令与这些资源进行交互。

Amazon Lambda此命令支持Amazon API Gateway 资源类型。

> Note
>
> 为您的Amazon API Gateway 资源配置自定义域时支持自定义域。此命令将输出自定义域而不是默认端点。

用量

```
sam list endpoints <option> <parameter> ...
```

选项

选项	描述
`--config-env TEXT`	环境名称，用于指定要使用的配置文件中的默认参数值。 默认值：`default` 有关配置文件的详细信息，请参阅 Amazon SAM CLI 配置文件 (p. 441)。
`--config-file TEXT`	包含要使用的默认参数值的配置文件的路径和文件名。 默认值：`samconfig.toml`在当前工作目录中。

选项	描述
	有关配置文件的详细信息，请参阅 Amazon SAM CLI 配置文件 (p. 441)。
`--debug`	开启调试日志，打印Amazon SAM CLI 生成的带有时间戳的调试消息。
`-h, --help`	显示此消息并退出。
`--output [json|table]`	指定输出结果的格式。 默认值：`table`
`--profile TEXT`	使用凭证文件中的特定配置文件，使用凭证文件中的特定配置文件。Amazon
`--region TEXT`	设置服务的Amazon区域。例如，`us-east-1`。
`--stack-name TEXT`	已部署Amazon CloudFormation堆栈的名称。堆栈名称可以在应用程序`samconfig.toml`的文件或指定的配置文件中找到。 如果未指定此选项，则将显示模板中定义的本地资源。
`-t, --template-file, --template PATH`	Amazon SAM模板文件。 默认值：`template.[yaml|yml|json]`

示例

以 json 格式显示名为的Amazon CloudFormation堆栈中部署的资源终端节点的输出`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']"
    ]
  }
]
```

垃圾列表资源

显示您的Amazon Serverless Application Model (Amazon SAM) 模板中Amazon CloudFormation由部署时Amazon SAM转换在中创建的资源。

在部署前sam list resources与Amazon SAM模板一起使用以查看将要创建的资源。提供Amazon CloudFormation堆栈名称以查看包含已部署资源的合并列表。

Note

要从您的Amazon SAM模板生成资源列表，需要对模板进行本地转换。将在特定区域等条件下部署的资源包含在此列表中。

用量

```
sam list resources <option> <parameter> ...
```

选项

选项	描述
`--config-env TEXT`	环境名称，用于指定要使用的配置文件中的默认参数值。 默认值：`default` 有关配置文件的详细信息，请参阅 Amazon SAM CLI 配置文件 (p. 441)。
`--config-file TEXT`	包含要使用的默认参数值的配置文件的路径和文件名。 默认值：`samconfig.toml`在当前工作目录中。 有关配置文件的详细信息，请参阅 Amazon SAM CLI 配置文件 (p. 441)。
`--debug`	开启调试日志，打印Amazon SAM CLI 生成的带有时间戳的调试消息。
`-h, --help`	显示此消息并退出。
`--output [json|table]`	指定输出结果的格式。 默认值：`table`
`--profile TEXT`	从凭证文件中选择特定配置文件以获取Amazon凭证文件。
`--region TEXT`	设置服务的Amazon区域。例如，`us-east-1`。
`--stack-name TEXT`	已部署Amazon CloudFormation堆栈的名称。堆栈名称可以在应用程序`samconfig.toml`的文件或指定的配置文件中找到。 提供后，模板中的资源逻辑 ID 将映射到中对应的物理 IDAmazon CloudFormation。要了解有关物理 ID 的更多信息，请参阅Amazon CloudFormation用户指南中的资源字段。 如果未指定此选项，则将显示模板中定义的本地资源。
`-t, --template-file, --template PATH`	Amazon SAM模板文件。

选项	描述
	默认值：`template.[yaml|yml|json]`

示例

以表格格式显示Amazon SAM模板中的本地资源和名为的Amazon CloudFormation堆栈中部署的资源的输出`test-stack`。使用与本地模板相同的目录中运行。

```
$ 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                           -
-------------------------------------------------------------------------------------------------------------
```

相同列表堆栈输出

显示来自Amazon Serverless Application Model (Amazon SAM) 或Amazon CloudFormation模板的Amazon CloudFormation堆栈输出。有关的更多信息`Outputs`，请参阅《Amazon CloudFormation用户指南》中的输出。

用量

```
sam list stack-outputs <option> <parameter> ...
```

选项

选项	描述
`--config-env TEXT`	环境名称，用于指定要使用的配置文件中的默认参数值。 默认值：`default` 有关配置文件的详细信息，请参阅 Amazon SAM CLI 配置文件 (p. 441)。
`--config-file TEXT`	包含要使用的默认参数值的配置文件的路径和文件名。 默认值：`samconfig.toml`在当前工作目录中。 有关配置文件的详细信息，请参阅 Amazon SAM CLI 配置文件 (p. 441)。

选项	描述
`--debug`	开启调试日志，打印Amazon SAM CLI 生成的带有时间戳的调试消息。
`-h, --help`	显示此消息并退出。
`--output [json|table]`	指定输出结果的格式。 默认值：`table`
`--profile TEXT`	使用凭证文件中的特定配置文件以获取Amazon凭证文件。
`--region TEXT`	设置服务的Amazon区域。例如，`us-east-1`。
`--stack-name TEXT`	已部署Amazon CloudFormation堆栈的名称。堆栈名称可以在应用程序`samconfig.toml`的文件或指定的配置文件中找到。 此选项是必需的。

示例

以表格格式显示Amazon CloudFormation堆栈中名为的资源的输出`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 本地生成事件

从不同的事件来源（例如 Amazon S3、Amazon Amazon API Gateway 和 Amazon SNS）生成示例有效负载。这些有效负载包含事件源发送给您的 Lambda 函数的信息。

使用方法：

```
sam local generate-event [OPTIONS] COMMAND [ARGS]...
```

示例：

```
Generate the event that S3 sends to your Lambda function when a new object is uploaded
sam local generate-event s3 [put/delete]

# You can even customize the event by adding parameter flags. To find which flags apply to
 your command,
run:

sam local generate-event s3 [put/delete] --help

# Then you can add in those flags that you wish to customize using

sam local generate-event s3 [put/delete] --bucket <bucket> --key <key>

# After you generate a sample event, you can use it to test your Lambda function locally
sam local generate-event s3 [put/delete] --bucket <bucket> --key <key> | sam local invoke -
e - <function logical id>
```

选项：

选项	描述
`--config-file PATH`	配置文件的路径和文件名，包含要使用的默认参数值。默认值为项目目录根目录中的 “samconfig.toml”。有关配置文件的详细信息，请参阅 Amazon SAM CLI 配置文件 (p. 441)。
`--config-env TEXT`	指定配置文件中要使用的默认参数值的环境名称。默认值为 “默认值”。有关配置文件的详细信息，请参阅 Amazon SAM CLI 配置文件 (p. 441)。
`--help`	显示此消息并退出。

命令：

- alb
- alexa-skills-kit
- alexa-smart-home
- apigateway
- batch
- cloudformation
- cloudfront
- cloudwatch
- codecommit
- codepipeline
- cognito
- config
- dynamodb
- kinesis
- lex
- lex-v2
- rekognition
- s3
- ses
- sns

- sqs
- stepfunctions
- workmail

sam 本地调用

调用一次局部Amazon Lambda函数，并在调用完成后退出。

默认情况下，当您使用此命令时，Amazon SAMCLI 假定您当前的工作目录是项目的根目录。CAmazon SAM LI 首先尝试找到使用山姆·布莱德 (p. 408)命令生成的模板文件，该文件位于.aws-sam子文件夹中，命名为template.yaml或template.yml。接下来，Amazon SAMCLI 会尝试在当前工作目录template.yml中找到名为template.yaml或的模板文件。如果您指定该--template选项，Amazon SAMCLI 的默认行为将被覆盖，并将仅加载该Amazon SAM模板及其指向的本地资源。

要调用嵌套应用程序或堆栈的函数，您可以使用以下格式提供应用程序或堆栈逻辑 ID 以及函数逻辑 IDStackLogicalId/FunctionLogicalId。

该sam local invoke命令对于开发处理异步事件的无服务器函数非常有用，例如来自 Amazon Kinesiis 事件或 Simple Storage Service (Amazon S3) Amazon Kinesis。如果你想编写测试用例的脚本，它也很有用。您可以使用--event参数传递事件正文。有关事件的更多信息，请参阅Amazon Lambda开发者指南中的事件。有关来自不同Amazon服务的事件消息格式的详细信息，请参阅Amazon Lambda开发者指南中的使用其他服务。

运行时输出（例如日志）输出到stderr，Lambda 函数结果输出到stdout。

> **Note**
>
> 如果您的Amazon SAM模板中定义了多个函数，则必须提供要调用的函数的。FUNCTION_LOGICAL_ID

用法：

```
sam local invoke [OPTIONS] [FUNCTION_LOGICAL_ID]
```

选项：

选项	描述
--hook-name TEXT	用于扩展Amazon SAM CLI 功能的挂接名称。 可接受的值:terraform.
--skip-prepare-infra	如果未对基础架构进行任何更改，则跳过准备阶段。与--hook-name选项一起使用。
-e, --event PATH	包含调用 Lambda 函数时传递给 Lambda 函数的事件数据的 JSON 文件。如果不指定此选项，则不假定任何事件。要从中输入 JSONstdin，必须传入值 '-'。有关来自不同Amazon服务的事件消息格式的详细信息，请参阅Amazon Lambda开发者指南中的使用其他服务。
--no-event	使用空事件调用该函数。
-t, --template PATH	Amazon SAM模板文件。 注意：如果指定此选项，则仅Amazon SAM加载模板及其指向的本地资源。此选项与不兼容--hook-name。

选项	描述
`-n, --env-vars PATH`	包含 Lambda 函数环境变量值的 JSON 文件。有关环境变量文件的详细信息，请参阅环境变量文件 (p. 358)。
`--parameter-overrides`	（可选）包含编码为键值对的Amazon CloudFormation参数覆盖的字符串。使用与Amazon Command Line Interface (Amazon CLI) 相同的格式。例如：'ParameterKey=KeyPairName、=、ParameterValueMyKeyParameterKey =InstanceType、ParameterValue =t1.micro'。此选项与不兼容--hook-name。
`-d, --debug-port TEXT`	指定后，在调试模式下启动 Lambda 函数容器并在本地主机上公开此端口。
`--debugger-path TEXT`	安装到 Lambda 容器中的调试器的主机路径。
`--debug-args TEXT`	要传递给调试器的其他参数。
`-v, --docker-volume-basedir TEXT`	Amazon SAM文件所在的基本目录的位置。如果 Docker 在远程计算机上运行，则必须在 Docker 计算机上安装Amazon SAM文件存在的路径，然后修改此值以匹配远程计算机。
`--docker-network TEXT`	Lambda Docker 容器应连接的现有 Docker 网络的名称或 ID，以及默认的桥接网络。如果未指定，Lambda 容器将仅连接到默认的桥接 Docker 网络。
`--container-env-vars`	（可选）在本地调试时，将环境变量传递给 Lambda 函数图像容器。
`-l, --log-file TEXT`	要向其发送运行时日志的日志文件。
`--layer-cache-basedir DIRECTORY`	指定将模板使用的图层下载到的基目录的位置。
`--skip-pull-image`	默认情况下，Amazon SAMCLI 检查 Lambda 的最新远程运行时环境并自动更新您的本地映像以保持同步。 指定此选项可跳过下拉您的 Lambda 运行时环境的最新Docker映像。
`--beta-features | --no-beta-features`	允许或拒绝测试版功能。
`--force-image-build`	指定Amazon SAM CLI 是否应重建用于通过层调用 Lambda 函数的镜像。
`--invoke-image TEXT`	要用于本地函数调用的容器镜像的 URI。默认情况下，从 Amazon ECRAmazon SAM Public ic 中提取。使用此选项从其他位置拉取镜像。 例如，`sam local invoke MyFunction --invoke-image amazon/aws-sam-cli-emulation-image-python3.8`。
`--profile TEXT`	您的凭证文件中获取Amazon证书的特定配置文件。
`--region TEXT`	要部署到的Amazon区域。例如，us-east-1。
`--config-file PATH`	包含要使用的默认参数值的配置文件的路径和文件名。项目目录根目录中的默认值为`samconfig.toml` “”。有关配置文件的详细信息，请参阅Amazon SAM CLI 配置文件 (p. 441)。
`--config-env TEXT`	环境名称，用于指定要使用的配置文件中的默认参数值。默认值为 “。有关配置文件的详细信息，请参阅Amazon SAM CLI 配置文件 (p. 441)。
`--shutdown`	在调用完成后模拟关机事件，以测试扩展对关机行为的处理。

选项	描述
`--container-host TEXT`	本地仿真 Lambda 容器的主机。默认值为 `localhost`。如果你想在 macOS 上的 Docker 容器中运行Amazon SAM CLI，你可以指定`host.docker.internal`。如果要在与Amazon SAM CLI 不同的主机上运行容器，则可以指定远程主机的 IP 地址。
`--container-host-interface TEXT`	容器端口应绑定到的主机网络接口的 IP 地址。默认值为 `127.0.0.1`。用于绑定`0.0.0.0`到所有接口。
`--debug`	打开调试日志记录以打印Amazon SAM CLI 生成的调试消息并显示时间戳。
`--help`	显示此消息并退出。

sam 本地启动 api

允许您在本地运行无服务器应用程序以进行快速开发和测试。当您在包含无服务器函数和Amazon SAM模板的目录中运行此命令时，它会创建一个托管所有函数的本地 HTTP 服务器。

默认情况下，当您使用此命令时，Amazon SAMCLI 假定您当前的工作目录是项目的根目录。CAmazon SAM LI 首先尝试找到使用山姆·布莱德 (p. 408)命令生成的模板文件，该文件位于`.aws-sam`子文件夹中，命名为`template.yaml`或`template.yml`。接下来，Amazon SAMCLI 会尝试在当前工作目录`template.yml`中找到名为`template.yaml`或的模板文件。如果您指定该`--template`选项，Amazon SAMCLI 的默认行为将被覆盖，并将仅加载该Amazon SAM模板及其指向的本地资源。

当访问（通过浏览器、CLI 等）时，它会在本地启动 Docker 容器来调用该函数。它读取`AWS::Serverless::Function`资源的 CodeUri 属性以查找文件系统中包含 Lambda 函数代码的路径。这可能是项目的 Node.js 和 Python 等解释型语言的根目录，也可以是存储已编译构件或 Java 存档 (JAR) 文件的构建目录。

如果您使用的是解释型语言，则每次调用时都会在 Docker 容器中立即使用本地更改。对于需要复杂打包支持的更多编译语言或项目，我们建议您运行自己的构建解决方案，并指Amazon SAM向包含编译构件的目录或文件。

要查看使用此命令的 end-to-end 示例，请参阅教程：部署 Hello World 应用程序 (p. 23)。该`sam local start-api`命令是其中的一部分步骤 6：（可选）在本地测试应用程序 (p. 34)。

用法：

```
sam local start-api [OPTIONS]
```

选项：

选项	描述
`--host TEXT`	要绑定的本地主机名或 IP 地址（默认值：'127.0.0.1'）。
`-p, --port INTEGER`	要监听的本地端口号（默认值：'3000'）。
`-s, --static-dir TEXT`	位于此目录中的任何静态资产（例如 CSS/JavaScript /HTML）文件均显示在/。
`-t, --template PATH`	Amazon SAM模板文件。 注意：如果指定此选项，则仅Amazon SAM加载模板及其指向的本地资源。
`-n, --env-vars PATH`	包含 Lambda 函数环境变量值的 JSON 文件。

选项	描述
`--parameter-overrides`	可选。一个包含以键值对编码的Amazon CloudFormation 参数替代选项可：使用与Amazon CLI —相同的格式，例如，'ParameterKey=KeyPairName， ParameterValueMyKey ParameterKey=InstanceType，ParameterValue=t1.micro'。
`-d, --debug-port TEXT`	指定后，在调试模式下启动 Lambda 函数容器并在本地主机上公开此端口。
`--debugger-path TEXT`	将安装到 Lambda 容器中的调试器的主机路径。
`--debug-args TEXT`	要传递给调试器的其他参数。
`--warm-containers [EAGER | LAZY]`	可选。指定Amazon SAM CLI 如何管理每个函数的容器。 以下： EAGER: 所有函数的容器在启动时加载，并在调用之间保持不变。 LAZY: 容器仅在首次调用每个函数时加载。这些容器会保留下来以供额外调用。
`--debug-function`	可选。指定在指定时应用调试选项`--warm-containers`的 Lambda 函数。此参数适用于`--debug-port``--debugger-path`、和`--debug-args`。
`-v, --docker-volume-basedir TEXT`	Amazon SAM文件所在的基本目录的位置。如果 Docker 在远程计算机上运行，则必须在 Docker 计算机上安装Amazon SAM文件存在的路径，然后修改此值以匹配远程计算机。
`--docker-network TEXT`	Lambda Docker 容器应连接的现有 Docker 网络的名称或 ID，以及默认的桥接网络。如果未指定此项，Lambda 容器将仅连接到默认的桥接 Docker 网络。
`--container-env-vars`	可选。在本地调试时，将环境变量传递给图像容器。
`-l, --log-file TEXT`	要向其发送运行时日志的日志文件。
`--layer-cache-basedir DIRECTORY`	指定将模板使用的图层下载到的基于位置。
`--skip-pull-image`	指定CLI是否应跳过下拉最新 Docker 镜像获取 Lambda 运行时的操作。
`--force-image-build`	指定 CLI 是否应重建用于调用带层的函数的镜像。
`--invoke-image TEXT`	您要在 Lambda 函数中使用容器映像的 URI。默认情况下，从 Amazon ECR Public 中Amazon SAM拉取容器镜像。使用此选项可： 您可以多次指定该选项。此选项的每个实例都可以采用字符串或键值对。如果您指定字符串，则它是用于应用程序中所有函数的容器映像的 URI。例如，`sam local start-api --invoke-image public.ecr.aws/sam/emu-python3.8`。如果您指定键值对，则密钥是资源名称，值是用于该资源的容器映像的 URI。例如 `sam local start-api --invoke-image public.ecr.aws/sam/emu-python3.8 --invoke-image Function1=amazon/aws-sam-cli-emulation-image-python3.8`。使用键值对，您可以为不同的资源指定不同的容器镜像。
`--profile TEXT`	您的凭证文件中用于获取Amazon证书的特定配置文件。

选项	描述
`--region TEXT`	要部署到的Amazon区域。例如，us-east-1。
`--config-file PATH`	包含要使用的默认参数值的配置文件的路径和文件名。项目目录根目录中的默认值为 “samconfig.toml”。有关配置文件的详细信息，请参阅 Amazon SAM CLI 配置文件 (p. 441)。
`--config-env TEXT`	环境名称，用于指定要使用的配置文件中的默认参数值。默认值为 “DISABLED" (默可：有关配置文件的详细信息，请参阅 Amazon SAM CLI 配置文件 (p. 441)。
`--shutdown`	在调用完成后模拟关机事件，以测试扩展对关机行为的处理。
`--container-host TEXT`	本地仿真 Lambda 容器的主机。默认值为 `localhost`。如果你想在 macOS 上的 Docker 容器中运行Amazon SAM CLI，你可以指定`host.docker.internal`。如果要在与Amazon SAM CLI 不同的主机上运行容器，则可以指定远程主机的 IP 地址。
`--container-host-interface TEXT`	容器端口应绑定到的主机网络接口的 IP 地址。默认值为 `127.0.0.1`。用于绑定`0.0.0.0`到所有接口。
`--debug`	打开调试日志记录以打印Amazon SAM CLI 生成的调试消息并显示时间戳。
`--help`	显示此消息并退出。

sam local start-la

允许您使用Amazon CLI或 SDK 以编程方式在本地调用 Lambda 函数。此命令启动模拟的本地端点Amazon Lambda。

默认情况下，当您使用此命令时，Amazon SAMCLI 会假定您当前的工作目录是项目的根目录。CAmazon SAM LI 首先尝试查找使用山姆·布莱德 (p. 408)命令生成的模板文件，该文件位于`.aws-sam`子文件夹中，命名为`template.yaml`或`template.yml`。接下来，Amazon SAMCLI 尝试在当前工作目录`template.yml`中找到名为`template.yaml`或的模板文件。如果您指定该`--template`选项，Amazon SAMCLI 的默认行为将被覆盖，并将仅加载该Amazon SAM模板及其指向的本地资源。

您可以针对此本地 Lambda 终端节点运行自动测试。当您使用Amazon CLI或 SDK 向此终端节点发送调用时，它将在本地执行请求中指定的 Lambda 函数。

用法：

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

示例：

```
# SETUP
# ------
# Start the local Lambda endpoint by running this command in the directory that contains
 your Amazon SAM template.

sam local start-lambda

# USING AWS CLI
# -------------
# Then, you can invoke your Lambda function locally using the AWS CLI
```

```
aws lambda invoke --function-name "HelloWorldFunction" --endpoint-url
 "http://127.0.0.1:3001" --no-verify-ssl out.txt

# USING AWS SDK
# -------------
# You can also use the AWS SDK in your automated tests to invoke your functions
 programatically.
# Here is a Python example:
#
#     self.lambda_client = boto3.client('lambda',
#                                       endpoint_url="http://127.0.0.1:3001",
#                                       use_ssl=False,
#                                       verify=False,
#                                       config=Config(signature_version=UNSIGNED,
#                                                     read_timeout=0,
#                                                     retries={'max_attempts': 0}))
#    self.lambda_client.invoke(FunctionName="HelloWorldFunction")
```

选项：

选项	描述
`--hook-name TEXT`	用于扩展Amazon SAM CLI 功能的挂钩的名称。 可接受的值:`terraform`.
`--skip-prepare-infra`	如果没有对基础架构进行任何更改，则跳过准备阶段。与`--hook-name`选项一起使用。
`--host TEXT`	要绑定的本地主机名或 IP 地址（默认值：'127.0.0.1'）。
`-p, --port INTEGER`	要监听的本地端口号（默认值：'3001'）。
`-t, --template PATH`	Amazon SAM模板文件。 注意：如果指定此选项，则仅Amazon SAM加载模板及其指向的本地资源。此选项与不兼容`--hook-name`。
`-n, --env-vars PATH`	包含 Lambda 函数环境变量值的 JSON 文件。
`--parameter-overrides`	可选。包含编码为键/值对的Amazon CloudFormation参数替代项的字符串。使用与Amazon CLI —for 相同的格式，例如 'ParameterKey=、= ParameterValueMyKey ParameterKey =KeyPairNameInstanceType、ParameterValue =t1.micro'。此选项与不兼容`--hook-name`。
`-d, --debug-port TEXT`	指定后，以调试模式启动 Lambda 函数容器，并在本地主机上公开此端口。
`--debugger-path TEXT`	要安装到 Lambda 容器中的调试器的主机路径。
`--debug-args TEXT`	要传递给调试器的其他参数。
`--warm-containers [EAGER | LAZY]`	可选。指定Amazon SAM CLI 如何管理每个函数的容器。 以下两个选项可用： `EAGER`：所有函数的容器在启动时加载，并在两次调用之间保持不变。 `LAZY`：仅在首次调用每个函数时加载容器。这些容器会保留以供额外调用。

选项	描述
`--debug-function`	可选。指定 Lambda 函数以在指定时`--warm-containers`应用调试选项。此参数适用于`--debug-port``--debugger-path`、和`--debug-args`。
`-v, --docker-volume-basedir TEXT`	Amazon SAM文件存在的基本目录的位置。如果 Docker 在远程计算机上运行，则必须挂载 Docker 计算机上Amazon SAM文件所在的路径，并修改此值以匹配远程计算机。
`--docker-network TEXT`	Lambda Docker 容器应连接到的现有 Docker 网络的名称或 ID，以及默认的桥接网络。如果指定此项，Lambda 容器将仅连接到默认的桥接 Docker 网络。
`--container-env-vars`	可选。在本地调试时将环境变量传递给图像容器。
`-l, --log-file TEXT`	要向其发送运行时日志的日志文件。
`--layer-cache-basedir DIRECTORY`	指定将模板使用的图层下载到的基准位置。
`--invoke-image TEXT`	要用于本地函数调用的容器镜像的 URI。默认情况下，从 Amazon ECR Public 中Amazon SAM拉取镜像。使用此选项从其他位置拉取镜像。 例如，`sam local start-lambda MyFunction --invoke-image amazon/aws-sam-cli-emulation-image-python3.8`。
`--skip-pull-image`	指定命令行界面是否应跳过下拉最新 Docker 镜像获取 Lambda 运行时的操作。
`--force-image-build`	指定 CLI 是否应重建用于调用带图层的函数的映像。
`--beta-features | --no-beta-features`	允许或拒绝测试版功能。
`--profile TEXT`	您的凭证文件中获取Amazon证书的特定个人资料。
`--region TEXT`	要部署到的Amazon区域。例如，us-east-1。
`--config-file PATH`	包含要使用的默认参数值的配置文件的路径和文件名。默认值为项目目录根目录下的 “samconfig.toml”。有关配置文件的详细信息，请参阅 Amazon SAM CLI 配置文件 (p. 441)。
`--config-env TEXT`	在配置文件中指定要使用的默认参数值的环境名称。默认值为 “默认”。有关配置文件的详细信息，请参阅 Amazon SAM CLI 配置文件 (p. 441)。
`--shutdown`	在调用完成后模拟关机事件，以测试关机行为的扩展处理。
`--container-host TEXT`	本地模拟的 Lambda 容器的主机。默认值为 `localhost`。如果你想在 macOS 上的 Docker 容器中运行Amazon SAM CLI，你可以指定`host.docker.internal`。如果要在不同于Amazon SAM CLI 的主机上运行容器，则可以指定远程主机的 IP 地址。
`--container-host-interface TEXT`	容器端口应绑定的主机网络接口的 IP 地址。默认值为 `127.0.0.1`。用于绑定`0.0.0.0`到所有接口。
`--debug`	打开调试日志记录以打印Amazon SAM CLI 生成的调试消息并显示时间戳。
`--help`	显示此消息并退出。

sam log

获取 Lambda 函数生成的日志。

当你的函数是一部分时Amazon CloudFormation堆栈，您可以在指定堆栈名称时使用函数的逻辑 ID 来获取日志。

使用方法：

```
sam logs [OPTIONS]
```

示例：

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

# You can view logs for a specific time range using the -s (--start-time) and -e (--end-
time) options
sam logs -n HelloWorldFunction --stack-name mystack -s '10min ago' -e '2min ago'

# You can also add the --tail option to wait for new logs and see them as they arrive.
sam logs -n HelloWorldFunction --stack-name mystack --tail

# Use the --filter option to quickly find logs that match terms, phrases or values in your
 log events.
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
```

选项：

选项	描述
-n, --name TEXT	要获取日志的资源的名称。如果此资源是一部分Amazon CloudFormation堆栈，这可以是中函数资源的逻辑 IDAmazon CloudFormation/Amazon SAMTemplate。通过再次重复参数可以提供多个名称。如果资源位于嵌套堆栈中，则可以在该名称前面加上嵌套堆栈名称的名称，以便从该资源中提取日志（NestedStackLogicalId/ResourceLogicalId）。如果没有给出资源名称，将扫描给定的堆栈，并提取所有受支持资源的日志信息。如果不指定此选项，Amazon SAM获取指定堆栈中所有资源的日志。支持以下资源类型： • AWS::Serverless::Function • AWS::Lambda::Function • AWS::Serverless::Api • AWS::ApiGateway::RestApi • AWS::Serverless::HttpApi • AWS::ApiGatewayV2::Api

选项	描述
	• AWS::Serverless::StateMachine • AWS::StepFunctions::StateMachine
--stack-name TEXT	The name of the Amazon CloudFormation资源所属的堆栈。
--filter TEXT	您需指定表达式以在日志事件中快速查找匹配字词、短语或值的日志。这可以是简单的关键字（例如“错误”）或亚马逊支持的模式 CloudWatch 日志。有关语法，请参阅亚马逊 CloudWatch 日志文档.
-s, --start-time TEXT	从此时开始获取日志。时间可以是“5 分钟前”、“昨天”等相对值，也可以是像 '2018-01-01 10:10:10' 这样的格式时间戳。默认为“10 分钟前”。
-e, --end-time TEXT	到目前为止，获取记录。时间可以是“5 分钟前”、“明天”等相对值，也可以是格式化的时间戳，例如 '2018-01-01 10:10:10'。
--profile TEXT	获取凭证文件中的特定配置文件Amazon凭证。
--region TEXT	这些区域有：Amazon要部署到的区域。例如，us-east-1。
-t, --tail	尾部日志输出。这忽略了结束时间参数，并在日志可用时继续获取日志。
--include-traces	在日志输出中包括 X-Ray 跟踪。
--output TEXT	指定日志的输出格式。要打印格式化的日志，请指定text. 要将日志打印为 JSON，请指定json.
--cw-log-group LIST	包括来自 CloudWatch 记录您指定的日志组。如果您同时指定此选项name、Amazon SAM除了来自指定资源的日志外，还包括来自指定日志组的日志。
--config-file PATH	配置文件包含要使用的默认参数值的配置文件的路径和文件名。默认值为项目目录根目录中的“samconfig.toml”。有关配置文件的详细信息，请参阅 Amazon SAM CLI 配置文件 (p. 441)。
--config-env TEXT	指定配置文件中要使用的默认参数值的环境名称。默认值为“默认值”。有关配置文件的详细信息，请参阅 Amazon SAM CLI 配置文件 (p. 441)。
--debug	打开调试日志记录以打印由Amazon SAMCLI 并显示时间戳。
--help	显示此消息并退出。

sam package

打包 Amazon SAM 应用程序。此命令将创建.zip文件中的代码和依赖项，然后将文件上传到 Amazon Simple Storage Service (Amazon S3) 中。Amazon SAM启用对 Amazon S3 中存储的所有文件的加密。然后，它返回 Amazon SAM 模板的副本，并将对本地构件的引用替换为此命令已将构件上传到的 Amazon S3 位置。

默认情况下，使用此命令时，Amazon SAMCLI 假设你当前的工作目录是项目的根目录。这些区域有：Amazon SAMCLI 首先尝试查找使用山姆·布莱德 (p. 408)命令，位于.aws-sam子文件夹，并命名为template.yaml. 然后，Amazon SAMCLI 尝试找到名为的模板文件template.yaml要么template.yml在当前工作目录中。如果你指定--template选项，Amazon SAMCLI 的默认行为被覆盖，并且只会打包那个Amazon SAM模板以及它指向的本地资源。

> Note
>
> sam deploy (p. 412)现在隐式执行的功能sam package. 您可以使用sam deploy (p. 412)命令直接打包和部署应用程序。

使用方法：

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

选项：

选项	描述
`-t, --template-file, --template PATH`	您的路径和文件名Amazon SAM已找到模板。 注意：如果指定此选项，Amazon SAM仅打包模板和它指向的本地资源。
`--s3-bucket TEXT`	Amazon S3 存储桶的名称，此命令上传您的 Amazon S3 存储桶的名称。Amazon CloudFormationTemplate。如果你的模板大于 51,200 字节，那么`--s3-bucket`或者`--resolve-s3`选项是必需的。如果指定了两个`--s3-bucket`和`--resolve-s3`选项，然后将会导致出现错误。
`--s3-prefix TEXT`	添加到上传到 Amazon S3 存储桶的工件名称中的前缀。前缀名称是 Amazon S3 存储桶的路径名称（文件夹名称）。这仅适用于声明的函数`Zip`软件包类型。
`--image-repository TEXT`	此命令上传函数映像的 Amazon Elastic Container Registry (Amazon ECR) 存储库的 URI。对于使用声明的函数所必需的`Image`软件包类型。
`--kms-key-id TEXT`	ID 的 IDAmazon Key Management Service(Amazon KMS) 用于加密 Amazon S3 存储桶中静态的项目的密钥。如果未指定此选项，则Amazon SAM使用 Amazon S3 托管加密密钥。
`--signing-profiles LIST`	（可选）用来签署部署包的签名配置文件列表。此参数获取键值对的列表，其中键是要签名的函数或图层的名称，值是签名配置文件，可选配置文件所有者分隔为：可选配置文件所有者。:. 例如：`FunctionNameToSign=SigningProfileName1 LayerNameToSign=SigningProfileName2:SigningProfileOwner`。
`--output-template-file PATH`	命令将打包模板写入到其中的文件路径。如果您未指定路径，此命令将模板写入标准输出中。
`--use-json`	输出 JSON 的输出 JSONAmazon CloudFormationTemplate。默认情况下使用 YAML。
`--resolve-s3`	自动创建用于打包的 Amazon S3 存储桶。如果指定了两个`--s3-bucket`和`--resolve-s3`选项，然后将会导致出现错误。
`--force-upload`	覆盖 Amazon S3 存储桶中的现有文件。指定此标志可上传工件，即使工件与 Amazon S3 存储桶中的现有工件匹配。
`--metadata`	（可选）要附加到模板中引用的所有工件的元数据映射。
`--profile TEXT`	获取凭证文件中的特定配置文件Amazon凭证。
`--region TEXT`	这些区域有：Amazon要部署到的区域。例如，us-east-1。
`--config-file PATH`	包含要使用的默认参数值的配置文件的路径和文件名。默认值为项目目录根目录中的 “samconfig.toml”。有关配置文件的详细信息，请参阅 Amazon SAM CLI 配置文件 (p. 441)。
`--config-env TEXT`	指定配置文件中要使用的默认参数值的环境名称。默认值为 “默认”。有关配置文件的详细信息，请参阅 Amazon SAM CLI 配置文件 (p. 441)。

选项	描述
`--no-progressbar`	将工件上传到 Amazon S3 时，请勿显示进度条。
`--debug`	打开调试日志记录以打印由Amazon SAMCLI 并显示时间戳。
`--help`	显示此消息并退出。

Note

如果Amazon SAM模板包含`Metadata`关于 serverlessRepo 的部分，以及`LicenseUrl`要么`ReadmeUrl`属性包含对本地文件的引用，必须更新Amazon CLI到 1.16.77 或更高版本。有关的更多信息`Metadata`的部分Amazon SAM模板和发布应用程序Amazon SAMCLI，请参阅使用Amazon SAM CLI 发布无服务器应用程序 (p. 382).

sam 管道引导

此命令生成连接到 CI/CD 系统所需的Amazon基础架构资源。在运行sam pipeline init命令之前，必须为管道中的每个部署阶段运行此步骤。

此命令设置以下Amazon基础架构资源：

- 通过以下方式配置管道权限的选项：
  - 具有要与 CI/CD 系统共享的访问密钥 ID 和私有密钥访问证书的管道 IAM 用户。

    Note

    我们建议定期轮换访问密钥。有关更多信息，请参阅《IAM 用户指南》中的对于需要长期凭证的使用场景定期轮换访问密钥。
  - 通过 OIDC 支持 CI/CD 平台。有关在Amazon SAM管道中使用 OIDC 的简介，请访问在Amazon SAM管道中使用 OIDC 身份验证 (p. 375)。
- Amazon CloudFormation用于部署Amazon SAM应用程序的Amazon CloudFormation执行 IAM 角色。
- 用于存放Amazon SAM工件的 Amazon S3 存储桶。
- （可选）一个 Amazon ECR 镜像存储库，用于存放容器镜像 Lambda 部署包（如果您有包类型的资源`Image`）。

用法：

```
sam pipeline bootstrap [OPTIONS]
```

选项：

选项	描述
`--config-env TEXT`	环境名称，用于指定要使用的配置文件中的默认参数值。默认值为**`default`**。有关配置文件的详细信息，请参阅 Amazon SAM CLI 配置文件 (p. 441)。
`--config-file PATH`	包含要使用的默认参数值的配置文件的路径和文件名。默认值`samconfig.toml`在项目目录的根目录中。有关配置文件的详细信息，请参阅 Amazon SAM CLI 配置文件 (p. 441)。
`--interactive | --no-interactive`	禁用引导参数的交互式提示，如果缺少任何必需的参数，则失败。默认值为`--interactive`。对于此命令，`--stage`是唯一需要的参数。

选项	描述
	注意：如果同时指定`--use-oidc-provider`，`--no-interactive`则必须包含 OIDC 提供商的所有必需参数。
`--stage TEXT`	相应的部署阶段的名称。它用作创建Amazon的基础架构资源的后缀。
`--pipeline-user TEXT`	与 CI/CD 系统共享访问密钥的 IAM 用户的 Amazon 资源名称（ARN）。它用于授予此 IAM 用户访问相应Amazon账户的权限。如果未提供，该命令将创建 IAM 用户以及访问密访问密钥凭证。
`--pipeline-execution-role TEXT`	管道用户承担的 IAM 角色的 ARN，将在此阶段运行。仅当您希望使用自己的角色时提供。如果未提供，则此命令将创建一个新角色。
`--cloudformation-execution-role TEXT`	部署应用程序堆栈Amazon CloudFormation时要担任的 IAM 角色的 ARN。仅当您希望使用自己的角色时提供。否则，该命令将创建一个新角色。
`--bucket TEXT`	包含Amazon SAM工件的 Amazon S3 桶的 ARN。
`--create-image-repository | --no-create-image-repository`	如果未提供 Amazon ECR 映像存储库，请指定是否创建 Amazon ECR 映像存储库。Amazon ECR 存储库保存 Lambda 函数或包类型为的层的容器映像`Image`。默认为 `--no-create-image-repository`。
`--image-repository TEXT`	存放 Lambda 函数或包类型为的层的容器映像的 Amazon ECR 镜像存储库的 ARN`Image`。如果提供，则忽略这些`--create-image-repository`选项。如果未提供且`--create-image-repository`已指定，则该命令将创建一个。
`--confirm-changeset | --no-confirm-changeset`	提示确认您的资源部署。
`--permissions-provider [oidc | iam]`	选择权限提供者来担任管道执行角色。默认值为 **`iam`**。
`--oidc-provider-url TEXT`	OIDC 提供者的URL。值必须以开头**`https://`**。
`--oidc-client-id TEXT`	配置为与您的 OIDC 提供商一起使用的客户端 ID。
`--github-org TEXT`	存储库所属的 GitHub 组织。如果不存在组织，请输入存储库所有者的用户名。此选项专用于使用 Ac GitHub tions OIDC 获取权限。
`--github-repo TEXT`	进行部署的 GitHub 存储库的名称。此选项专用于使用 Ac GitHub tions OIDC 获取权限。
`--deployment-branch TEXT`	进行部署的分支的名称。此选项专用于使用 Ac GitHub tions OIDC 获取权限。
`--oidc-provider [github-actions | gitlab | bitbucket-pipelines]`	将用于 OIDC 权限的 CI/CD 提供商的名称。 GitLab支持 GitHub、和 Bitbucket。
`--gitlab-group TEXT`	存储库所属的 GitLab 组。此选项特定于使用 GitLab OIDC 获取权限。
`--gitlab-project TEXT`	GitLab 项目名称。此选项特定于使用 GitLab OIDC 获取权限。

选项	描述
`--bitbucket-repo-uuid TEXT`	Bitbucket 存储库的 UUID。此选项特定于使用 Bitbucket OIDC 获取权限。 Note 这个值可以在 https://bitbucket.org/ w *orkspace*/### /admin/addon/admin/pipelines/openid-connect 上找到
`cicd-provider TEXT`	Amazon SAM管道的 CI/CD 平台。
`--debug`	打开调试日志记录并打印Amazon SAM CLI 生成的调试消息，并显示时间戳。
`--profile TEXT`	您的凭证文件中用于获取Amazon证书的特定配置文件。
`--region TEXT`	要部署到的Amazon区域。例如，`us-east-1`。
`-h, --help`	显示此消息并退出。

问题排查

错误：缺少必需的参数

如果`--no-interactive`同时指定了`--use-oidc-provider`且未提供任何必需的参数，则将显示此错误消息以及缺失参数的描述。

sam 管道 init

此命令生成一个管道配置文件，CI/CD 系统可以使用该文件来部署无服务器应用程序Amazon SAM.

在使用前sam pipeline init，您必须为管道中的每个阶段引导必要的资源。你可以运行这项操作sam pipeline init --bootstrap以指导完成设置和配置文件生成过程，或者参考之前使用sam pipeline bootstrap命令。

使用方法：

```
sam pipeline init [OPTIONS]
```

选项：

选项	描述
`--config-env TEXT`	指定配置文件中要使用的默认参数值的环境名称。默认值为 `default`。有关配置文件的详细信息，请参阅 Amazon SAM CLI 配置文件 (p. 441)。
`--config-file TEXT`	包含要使用的默认参数值的配置文件的路径和文件名。默认值是 。`samconfig.toml`在项目根目录中。有关配置文件的详细信息，请参阅 Amazon SAM CLI 配置文件 (p. 441)。
`--bootstrap`	启用交互模式，引导用户完成必要的创建Amazon基础设施资源。
`--debug`	打开调试日志记录以打印调试消息Amazon SAMCLI 生成，并显示时间戳。
`-h, --help`	显示此消息并退出。

sam publish

发布Amazon SAM申请到Amazon Serverless Application Repository. 需要一个打包的Amazon SAM模板并将应用程序发布到指定的Amazon区域。

这些区域有：`sam publish`命令期望Amazon SAM要包含一个模板`Metadata`该部分包含发布所需的应用程序元数据。在`Metadata`部分中，`LicenseUrl`和`ReadmeUrl`属性必须参考 Amazon Simple Storage Service (Amazon S3) 存储桶，而不是本地文件。有关的更多信息`Metadata`的 部分Amazon SAM模板，请参阅使用Amazon SAM CLI 发布无服务器应用程序 (p. 382).

默认情况下，`sam publish`将应用程序创建为私有。在其他之前Amazon允许帐户查看和部署您的应用程序，您必须共享它。有关共享应用程序，请参阅。Amazon Serverless Application Repository基于资源的策略示例中的Amazon Serverless Application Repository开发人员指南.

> **Note**
>
> 目前`sam publish`不支持发布本地指定的嵌套应用程序。如果您的应用程序包含嵌套应用程序，则必须将其单独发布到Amazon Serverless Application Repository在发布父应用程序之前。

使用方法：

```
sam publish [OPTIONS]
```

示例：

```
# To publish an application
sam publish --template packaged.yaml --region us-east-1
```

选项：

选项	描述
`-t, --template PATH`	的路径Amazon SAM模板文件`[default: template.[yaml|yml]]`.
`--semantic-version TEXT`	（可选）使用此选项提供应用程序的语义版本，该版本覆盖`SemanticVersion`中的财产`Metadata`部分为模板文件。有关语义版本控制的更多信息，请参阅语义版本控制规范.
`--profile TEXT`	从您的凭证文件中获取的特定配置文件Amazon凭证。
`--region TEXT`	这些区域有：Amazon要部署到的区域。例如，us-east-1。
`--config-file PATH`	配置文件的路径和文件名，其中包含要使用的默认参数值。默认值为"`samconfig.toml`"在项目目录的根目录中。有关配置文件的详细信息，请参阅 Amazon SAM CLI 配置文件 (p. 441)。
`--config-env TEXT`	指定配置文件中要使用的默认参数值的环境名称。默认值为 “默认值”。有关配置文件的详细信息，请参阅 Amazon SAM CLI 配置文件 (p. 441)。
`--debug`	打开调试日志记录以打印调试消息Amazon SAMCLI 生成，并显示时间戳。
`--help`	显示此消息并退出。

山姆同步

使用Amazon Serverless Application Model命令行界面 (Amazon SAMCLI)`sam sync` 命令将本地应用程序更改同步到Amazon Web Services 云。

- 有关Amazon SAM CLI 的简介，请参阅什么是Amazon SAM CLI？ (p. 6)。
- 有关使用Amazon SAM CLI 的文档，请参阅使用 Amazon SAM CLI (p. 37)。

用量

```
$ sam sync <command> <option> ...
```

选项

选项	描述
-t, --template-file, --template PATH	您的Amazon SAM模板所在的路径和文件名。 注意：如果您指定此选项，则仅Amazon SAM部署模板及其指向的本地资源。
--code	默认情况下，Amazon SAM同步应用程序中的所有资源。指定此选项以仅同步代码资源，其中包括以下内容： • 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 要同步代码资源，请直接Amazon SAM使用Amazon服务 API，而不是通过进行部署Amazon CloudFormation。要更新Amazon CloudFormation堆栈，请运行sam sync --watch或sam deploy。
--watch	启动一个进程，监视您的本地应用程序是否有更改，并自动将其同步到 Amazon Web Services 云。默认情况下，当您指定此选项时，会在更新应用程序中的所有资源时Amazon SAM同步它们。使用此选项，Amazon SAM执行初始Amazon CloudFormation部署。然后，Amazon SAM使用Amazon服务 API 更新代码资源。 Amazon SAMAmazon CloudFormation 用于在更新Amazon SAM模板时更新基础架构资源。
--resource-id TEXT	指定要同步的资源 ID。要同步多个资源，可以多次指定此选项。该选项支持此--code选项。例如，--resource-id Function1 --resource-id Function2。
--resource TEXT	指定要同步的资源类型。要同步多个资源，可以多次指定此选项。该选项支持此--code选项。必须是以下内容内容内容内容内容内容内容内容内容内容内容内容内容内容内容内容内容内容--code内容之一 例如，--resource AWS::Serverless::Function --resouce AWS::Serverless::LayerVersion。

选项	描述
`--skip-deploy-sync | --no-skip-deploy-sync`	如果不需要初始基础架构同步，请指定`--skip-deploy-sync`跳过初始基础架构同步。Amazon SAMCLI 会将您的本地Amazon SAM模板与已部署的Amazon CloudFormation模板进行比较，并仅在检测到更改时才执行部署。 指定`--no-skip-deploy-sync`在每次运行`sam sync`时执行Amazon CloudFormation部署。 要了解更多信息，请参阅 跳过初始Amazon CloudFormation部署 (p. 70)。 默认选项是 `--skip-deploy-sync`。
`--dependency-layer | --no-dependency-layer`	指定是否将单个函数的依赖关系分成另一层以加快同步过程。 默认设置为 `--dependency-layer`。
`-u, --use-container`	如果您的函数依赖于具有原生编译依赖项的包，请使用此选项在Amazon Lambda类似 Docker 的容器中构建您的函数。 注意：目前，此选项与不兼容`--dependency-layer`。如果您使用和`--use-container``--dependency-layer`，Amazon SAMCLI 会通知您并继续使用`--no-dependency-layer`。
`--stack-name TEXT`	（必需）您的应用程序的Amazon CloudFormation堆栈名称。
`--s3-bucket TEXT`	此此此命令用于上载您的Amazon CloudFormation模板的 Amazon Simple Storage Service Service Service Service 如果您的模板大于 51,200 字节，则需要使用`--s3-bucket`或`--resolve-s3`选项。如果您同时指定`--s3-bucket`和`--resolve-s3`选项，则会出现错误。
`--s3-prefix TEXT`	您上载到的 Amazon S3 存储桶的工件名称中添加了该前缀。前缀是 Amazon S3 存储桶的路径名称（文件夹名称）。这仅适用于使用`Zip`包类型声明的函数。
`--capabilities LIST`	您指定允许创建某些堆栈Amazon CloudFormation的功能列表。一些堆栈模板可能影响您Amazon Web Services 账户的中的Amazon Identity and Access Management权限。默认功能为`CAPABILITY_NAMED_IAM`和`CAPABILITY_AUTO_EXPAND`。指定此选项可覆盖默认值。有效值包括： • CABILITY_IAM • CAPABILITY_NAMED_IAM • 能力_资源_政策 • CAPABILITY_AUTO_EXPAND

选项	描述
-s, --base-dir DIRECTORY	解析相对于该目录的函数或图层源代码的相对路径。使用此选项更改解析源代码文件夹相对路径的方式。默认情况下，相对路径是相对于Amazon SAM模板的位置进行解析的。 除了您正在构建的根应用程序或堆栈中的资源外，此选项还适用于嵌套应用程序或堆栈。此外，此选项适用于以下资源类型和属性： • 资源类型：AWS::Serverless::Function属性：CodeUri • 资源类型：AWS::Serverless::Function资源属性：Metadata条目：DockerContext • 资源类型：AWS::Serverless::LayerVersion属性：ContentUri • 资源类型：AWS::Lambda::Function属性：Code • 资源类型：AWS::Lambda::LayerVersion属性：Content
--parameter-overrides	一个包含以键Amazon CloudFormation和/或对的形式进行编码。使用与Amazon Command Line Interface (Amazon CLI) 相同的格式。例如，ParameterKey=ParameterValue InstanceType=t1.micro。
--image-repository TEXT	此此此此用于上传您的函数镜像的 Amazon Elastic Container Registry (Amazon ECR) 存储库的名称。使用Image包类型声明的函数是必需的。
--kms-key-id TEXT	用于加密 Amazon S3 存储桶中静态对象的Amazon Key Management Service (Amazon KMS) 密钥的 ID。如果您未指定此此可Amazon SAM用 Amazon S3 托管式加密密钥。
--role-arn TEXT	应用变更集时Amazon CloudFormation担任的 IAM 角色的 Amazon 资源名称 (ARN)。
--notification-arns LIST	与堆栈关联的 Amazon Simple NoticAmazon CloudFormation ation Service Service (Amazon SNS) 主题 ARN。
--tags LIST	与已创建或更新的堆栈关联的标签列表。 Amazon CloudFormation还可以将这些标签传播到堆栈中支持它的资源。
--metadata	用于附加到您在模板中引用的所有对象的元数据地图。

sam trace

取回Amazon X-Ray你的痕迹Amazon Web Services 账户中的Amazon Web Services 区域.

使用方法：

```
sam traces [OPTIONS]
```

选项：

选项	描述
--trace-id TEXT	X-Ray 跟踪的唯一标识符。
--start-time TEXT	从此时开始获取跟踪信息。时间可以是 “5 分钟前”、“昨天” 等相对值，也可以是像 '2018-01-01 10:10:10' 这样的格式时间戳。默认为 “10 分钟前”。

选项	描述
`--end-time TEXT`	到目前为止，获取跟踪信息。时间可以是 “5 分钟前”、“明天” 等相对值，也可以是格式化的时间戳，例如 '2018-01-01 10:10:10'。
`--tail`	尾部跟踪输出。这忽略了结束时间参数，并在痕迹变得可用时继续显示。
`--output TEXT`	指定日志的输出格式。要打印格式化的日志，请指定`text`. 要将日志打印为 JSON，请指定`json`.

示例

运行以下命令按 ID 获取 X-Ray 跟踪。

```
sam traces --trace-id tracing-id-1 --trace-id tracing-id-2
```

运行以下命令以在 X-Ray 跟踪可用时将其尾部。

```
sam traces --tail
```

sam validate

验证Amazon SAM模板文件是否有效。

用法：

```
sam validate [OPTIONS]
```

选项：

选项	描述
`-t, --template, --template-file PATH`	Amazon SAM模板文件。默认值为 `template.[yaml|yml]`. 如果您的模板位于当前工作目录中并命名为`template.[yaml|yml|json]`，则不需要此选项。 如果您刚刚运行sam build，则不需要此选项。
`--lint`	通过对模板运行 linting 验证cfn-lint。创建`cfnlintrc`配置文件以指定其他参数。有关更多信息，请参阅Amazon CloudFormation GitHub 存储库中的cfn-lint。
`--profile TEXT`	您的凭证文件中获取Amazon证书的特定配置文件。
`--region TEXT`	要部署到的Amazon区域。例如，us-east-1。
`--config-file PATH`	包含要使用的默认参数值的配置文件的路径和文件名。项目目录根目录中的默认值为 “samconfig.toml”。有关配置文件的详细信息，请参阅 Amazon SAM CLI 配置文件 (p. 441)。
`--config-env TEXT`	环境名称，用于指定要使用的配置文件中的默认参数值。默认值为 “DISe”。有关配置文件的详细信息，请参阅 Amazon SAM CLI 配置文件 (p. 441)。

选项	描述
`--debug`	打开调试日志记录以打印Amazon SAM CLI 生成的调试消息并显示时间戳。

Amazon SAM CLI 配置文件

这些区域有：Amazon SAMCLI 支持存储其命令的默认参数的项目级配置文件。此配置文件位于TOML 文件格式，默认文件名为`samconfig.toml`. 文件的默认位置是项目的根目录，其中包含项目的Amazon SAM模板文件。

您可以手动编辑此文件以为任何文件设置默认参数Amazon SAMCLI 命令。此外，`sam deploy --guided`命令将参数子集写入配置文件。有关此命令的更多信息，请参阅使用编写配置`sam deploy --guided` (p. 442)本主题后面的。

示例

下面是一个示例配置文件，其中包含三组用于`default`环境。一套用于所有命令，一套用于`deploy`命令，一个是用于`build`命令。

```
version=0.1
[default.global.parameters]
stack_name = "common-stack"

[default.deploy.parameters]
stack_name = "my-app-stack"
s3_bucket = "my-source-bucket"
s3_prefix = "my-s3-prefix"
image_repositories = ["my-function-1=image-repo-1", "my-function-2=image-repo-2"]
region = "us-west-2"
confirm_changeset = true
capabilities = "CAPABILITY_IAM"
tags = "project=\"my-application\" stage=\"production\""

[default.build.parameters]
container_env_var = ["Function1.GITHUB_TOKEN=TOKEN1", "Function2.GITHUB_TOKEN=TOKEN2"]
container_env_var_file = "env.json"
no_beta_features = true
```

配置文件规则

这些区域有：Amazon SAMCLI 将以下规则应用于配置文件：

文件名和位置

- 默认配置文件名为`samconfig.toml`并位于项目的根目录中。
- 您可以使用`--config-file`参数。

表

- 这些区域有：Amazon SAMCLI 使用 TOML 表按环境和命令对配置条目进行分组。单个配置文件可以包含多个环境的表、命令和子命令。
- 默认环境名称命名为`default`. 您可以使用`--config-env`参数。
- 对于命令，表头的格式为`[`*`environment.command`*`.parameters]`. 例如，对于`sam deploy`的命令`default`环境中，配置表标题为`[default.deploy.parameters]`.

- 对于子命令，表头的格式为[*environment*.*command_subcommand*.parameters]. 也就是说，用分隔命令和子命令_（下划线）。例如，对于sam local invoke的命令default环境中，配置表标题为[default.local_invoke.parameters].
- 如果任何命令或子命令包含-（连字符）字符，将其替换为_（下划线）。例如，对于sam local start-api命令，配置表标题是[default.local_start_api.parameters].
- 要为所有命令指定参数，请使用global关键字作为表标题中的命令([*environment*.global.parameters])。例如，用于的全局表标题default环境[default.global.parameters].

配置条目

- 每个配置条目都是一个 TOML 键值对。
- 配置密钥是带有-（连字符）字符替换为_（下划线）。有关每个命令的可用参数列表，请参阅Amazon SAMCLI 命令参考 (p. 408)或者运行sam *command* --help.
- 配置值可采用以下形式：
  - 对于切换参数，值可以是true要么false（没有引号）。例如：confirm_changeset = true。
  - 对于采用单个参数的参数，值是被包围的参数" "（引号）。例如：region = "us-west-2"。
  - 对于采用参数列表的参数，参数在其中以空格分隔" "（引号）。例如：capabilities = "CAPABILITY_IAM CAPABILITY_NAMED_IAM"。
    - 要指定键值对的列表，这些对用空格分隔，每对的值都被编码包围" "（引号）。例如：tags = "project=\"my-application\" stage=\"production\""。
  - 对于可以多次指定的参数，该值是参数数组。例如：image_repositories = ["my-function-1=image-repo-1", "my-function-2=image-repo-2"]。

优先顺序

- 您在命令行中提供的参数值优先于配置文件中的相应条目。例如，如果您的配置文件包含条目stack_name = "DefaultStack"然后你运行命令sam deploy --stack-name MyCustomStack，那么部署的堆栈名称为MyCustomStack.
- 对于parameter_overrides条目，您在命令行上提供的参数值和配置文件中的条目都优先于在Parameters模板文件的一节。
- 在特定命令表中提供的条目优先于global命令部分。例如，假定配置文件包含以下表和条目：

```
[default.global.parameters]
stack_name = "common-stack"

[default.deploy.parameters]
stack_name = "my-app-stack"
```

在这种情况下，sam deploy命令使用堆栈名称my-app-stack，以及任何其他命令（例如，sam logs）使用堆栈名称common-stack.

使用编写配置sam deploy --guided

当你运行sam deploy --guided命令，Amazon SAMCLI 通过一系列提示指导您完成部署。

这些提示包括问题"Save arguments to samconfig.toml [Y/n]:". 如果你回应**Y**在这个提示下，Amazon SAMCLI 使用的值更新配置文件deploy命令。例如，对于default环境Amazon SAM更新[default.deploy.parameters]表。

中的条目列表deploy命令表那Amazon SAM可以更新包含以下内容：

- stack_name
- s3_bucket
- s3_prefix
- image_repository
- region
- confirm_changeset
- capabilities
- signing_profiles
- disable_rollback
- parameter_overrides

  Note

  配置文件有一种特殊情况，该文件在两者中都包含相同参数的条目deploy和global命令表。在这种情况下，如果你运行sam deploy --guided并为该参数提供与global命令表条目，然后deploy命令表条目被删除。
  通过在sam deploy --guided提示已在中指定的相同值global命令表，Amazon SAM假设你想默认使用中的值global命令表。

引导式提示默认值的规则

要控制提示的默认值，请执行以下操作：Amazon SAM运行时会显示 CLIsam deploy --guided，您可以在命令行中指定参数，也可以在现有配置文件中指定条目。

这些提示的规则如下：

- 如果在命令行中指定值，则Amazon SAMCLI 使用这些命令行值作为相应提示的默认值。
- 如果有现有配置文件，则Amazon SAMCLI 使用该文件中匹配表中的条目作为相应提示的默认值。

命令行和配置文件之间的优先级规则与优先顺序本主题前面的一节。

管理Amazon SAM CLI 版本

通过升级、降级和卸载来管理您的Amazon Serverless Application Model命令行界面 (Amazon SAMCLI) 版本。或者，您可以选择下载并Amazon SAM安装夜间版本。

主题

升级 CAmazon SAM LI

Linux

要在 Linux 上升级 CLAmazon SAM I安装 Amazon SAM CLI (p. 15)，请按照中的安装说明进行操作，但在安装命令中添加--update选项，如下所示：

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

macOS

通过用于安装Amazon SAM CLI 的相同服务升级 CLI。

Homebrew

要使用 macOS 上的Amazon SAM CLI 升级Homebrew，请运行以下命令：

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

> Note
>
> 要升级到Amazon SAM v1.70.1 或更高版本，我们建议brew upgrade aws/tap/aws-sam-cli改为运行。

Package 安装程序

要使用软件包安装程序升级Amazon SAM CLI，请安装最新的软件包版本。有关说明，请参阅 安装 Amazon SAM CLI (p. 15)。

Windows

要升级Amazon SAM CLI，请安装 Amazon SAM CLI (p. 15)再次重复中的 Windows 安装步骤。

卸载Amazon SAM CLI

Linux

要在 Linux 上卸载 CLAmazon SAM I，必须通过运行以下命令删除符号链接和安装目录：

1. 找到符号链接和安装路径。
   - 使用which以下命令查找符号链接：

     ```
     which sam
     ```

     输出显示Amazon SAM二进制文件所在的路径，例如：

     ```
     /usr/local/bin/sam
     ```

   - 使用ls命令查找符号链接指向的目录：

     ```
     ls -l /usr/local/bin/sam
     ```

     在以下示例中，安装目录为`/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. 删除符号链接。

   ```
   sudo rm /usr/local/bin/sam
   ```

3. 删除安装目录。

   ```
   sudo rm -rf /usr/local/aws-sam-cli
   ```

macOS

通过用于安装Amazon SAM CLI 的相同服务卸载 CLI。

Homebrew

如果Amazon SAM CLI 是使用Homebrew或安装的pip，请运行以下命令将其卸载：

```
$ pip uninstall aws-sam-cli && brew uninstall aws-sam-cli
```

运行以下命令验证Amazon SAM CLI 是否已卸载：

```
$ sam --version
command not found: sam
```

Package 安装程序

如果Amazon SAM CLI 是使用软件包安装程序安装的，请使用以下命令将其卸载。

1. 通过修改和运行以下命令删除Amazon SAM CLI 程序：

   ```
   $ sudo rm -rf /path-to/aws-sam-cli
   ```

   a. *sudo* — 如果您的用户对Amazon SAM CLI 程序的安装位置具有写入权限，sudo则不需要。否则，sudo 是必需的。

   b. */path-t* o — 安装Amazon SAM CLI 程序的路径。默认位置是 `/usr/local`。

2. $PATH通过修改和运行以下Amazon SAM命令删除 CLI：

   ```
   $ sudo rm -rf /path-to-symlink-directory/sam
   ```

   a. *sudo* — 如果您的用户具有写入权限$PATH，sudo则不需要。否则，sudo 是必需的。

   b. *path-to-symlink-directory*— 您的$PATH环境变量。默认位置是 `/usr/local/bin`。

3. 运行以下命令验证Amazon SAM CLI 是否已卸载：

   ```
   $ sam --version
   command not found: sam
   ```

Windows

要使用 Windows 设置卸载Amazon SAM CLI，请按照以下步骤操作：

1. 在 “开始” 菜单中，搜索 “添加或删除程序”。
2. 选择名为 Amazon SAMCommand Line Interface 的结果，然后选择卸载以启动卸载程序。
3. 确认您要卸载 Amazon SAM CLI。

安装 CAmazon SAM LI 夜间版本

您可以下载并安装Amazon SAM命令行界面的夜间版本。它包含Amazon SAM CLI 代码的预发行版本，该版本可能不如生产版本稳定。安装后，可以在sam-nightly命令中使用夜间构建。您可以同时安装和使用Amazon SAM CLI 的生产和夜间构建版本。

Note

夜间版本不包含构建映像的预发行版本。因此，使用该--use-container选项构建您的无服务器应用程序会使用构建映像的最新生产版本。

要安装Amazon SAM CLI 夜间版本，请按照以下说明进行操作。

Linux

Command line installer

夜间版本可通过以下下载链接获得：Amazon SAMCLI nightly build。要安装Amazon SAM CLI 的夜间编译版本，请执行与该安装 Amazon SAM CLI (p. 15)部分相同的步骤，但改用夜间编译下载链接。你可以在夜间构建的 Amazon SAMCLI 发行说明中找到夜间编译安装程序文件的哈希值 GitHub。

要验证您是否已安装夜间编译版本，请运行sam-nightly --version命令。此命令的输出格式为`1.X.Y.dev<YYYYMMDDHHmm>`，例如：

```
SAM CLI, version 1.20.0.dev202103151200
```

Homebrew

要在 Linux 上安装Amazon SAM CLI 的夜间编译版本Homebrew，请使用运行以下命令：

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

要验证您是否已安装夜间编译版本，请运行sam-nightly --version命令。此命令的输出格式为`1.X.Y.dev<YYYYMMDDHHmm>`，例如：

```
SAM CLI, version 1.20.0.dev202103151200
```

macOS

要在 macOS 上安装Amazon SAM CLI 的夜间编译版本Homebrew，请使用运行以下命令：

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

要验证您是否已安装夜间编译版本，请运行sam-nightly --version命令。此命令的输出格式为`1.X.Y.dev<YYYYMMDDHHmm>`，例如：

```
SAM CLI, version 1.20.0.dev202103151200
```

Windows

Amazon SAMCLI 的夜间构建版本可通过以下下载链接获得：CL Amazon SAMI nightly build。要在 Windows 上安装夜间版本，请执行与中相同的步骤安装 Amazon SAM CLI (p. 15)，但请改用夜间编译下载链接。

要验证您是否已安装夜间编译版本，请运行sam-nightly --version命令。此命令的输出格式为`1.X.Y.dev<YYYYMMDDHHmm>`，例如：

```
SAM CLI, version 1.20.0.dev202103151200
```

使用以下方法将Amazon SAM CLI 安装到虚拟环境中pip

我们建议使用原生软件包安装程序来安装Amazon SAM CLI。如果您必须使用pip，我们建议您将Amazon SAM CLI 安装到虚拟环境中。这样可以确保在出现错误时提供干净的安装环境和隔离的环境。

将Amazon SAM CLI 安装到虚拟环境中或进行操作

1. 从您选择的起始目录中创建虚拟环境并命名它。

   Linux / macOS

   ```
   $ mkdir project
   $ cd project
   $ python3 -m venv venv
   ```

   Windows

   ```
   > mkdir project
   > cd project
   > py -3 -m venv venv
   ```

2. 激活虚拟环境

   Linux / macOS

   ```
   $ . venv/bin/activate
   ```

   提示符更改为显示您的虚拟环境处于活动状态。

   ```
   (venv) $
   ```

   Windows

   ```
   > venv\Scripts\activate
   ```

   提示符更改为显示您的虚拟环境处于活动状态。

   ```
   (venv) >
   ```

3. 将Amazon SAM CLI 安装到您的虚拟环境中或进行安装。

   ```
   (venv) $ pip install --upgrade aws-sam-cli
   ```

4. 验证 Amazon SAM CLI 是否已正确安装：

   ```
   (venv) $ sam --version
   SAM CLI, version 1.76.0
   ```

5. 您可以使用 `deactivate` 命令退出虚拟环境。不管何时启动新会话，都必须重新激活环境。

问题排查

如果您在安装或使用Amazon SAM CLI 时遇到错误，请参阅Amazon SAMCLI 纠正 (p. 449)。

设置Amazon证书

这些区域有：Amazon SAM要求您设置命令行界面 (CLI)Amazon凭据以便它可以调用Amazon代表您提供服务。例如，Amazon SAMCLI 调用 Amazon S3 和Amazon CloudFormation.

你可能已经设置Amazon要使用的凭据Amazon工具，就像其中一个Amazon开发工具包或Amazon CLI. 如果没有，本主题将向您展示建议的设置方法Amazon凭证。

要设置Amazon凭据，你必须拥有访问密钥 ID和您的私有访问密钥适用于要配置的 IAM 用户。有关访问密钥 ID 和秘密访问密钥的信息，请参阅。管理 IAM 用户的访问密钥中的IAM 用户指南.

接下来，确定你是否有Amazon CLI已安装。然后，按照以下某一部分中的说明操作：

使用 Amazon CLI

如果您有Amazon CLI已安装，请使用`aws configure`命令并按照提示进行操作：

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

有关的信息`aws configure`命令，请参阅快速配置Amazon CLI中的Amazon Command Line Interface用户指南.

不使用Amazon CLI

如果您没有Amazon CLI安装后，您可以创建凭证文件或设置环境变量：

- 凭证文件— 你可以在Amazon本地系统上的凭据文件。此文件必须位于以下位置之一中：
  - `~/.aws/credentials`在 Linux 或 macOS 上
  - Windows 上的 `C:\Users\USERNAME\.aws\credentials`

  此文件应包含以下格式的行：

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

- 环境变量— 您可以将`AWS_ACCESS_KEY_ID`和`AWS_SECRET_ACCESS_KEY`环境变量。

  要在 Linux 或 macOS 上设置这些变量，请使用出口命令：

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

  要在 Windows 上设置这些变量，请使用设置命令：

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

Amazon SAMCLI 纠正

对使用、安装和管理Amazon Serverless Application Model命令行界面 (Amazon SAMCLI) 时的错误消息进行故障排除。

主题

安装错误

Linux

Docker 错误：“无法连接 Docker er 错误 Docker 错误：“无法连接Do docker 守护程序在这台主机上运行吗？”

在某些情况下，要访问 Docker er Docker docker Docker docker Docker er cker docker Dockerec2-user 如果您收到此错误，请尝试重启您的实例。

Shell 错误：“找不到命令”

如果您收到此错误，则说明您的 shell 无法在路径中找到Amazon SAM CLI 可执行文件。验证您安装Amazon SAM CLI 可执行文件的目录的位置，然后验证该目录是否在您的路径中。

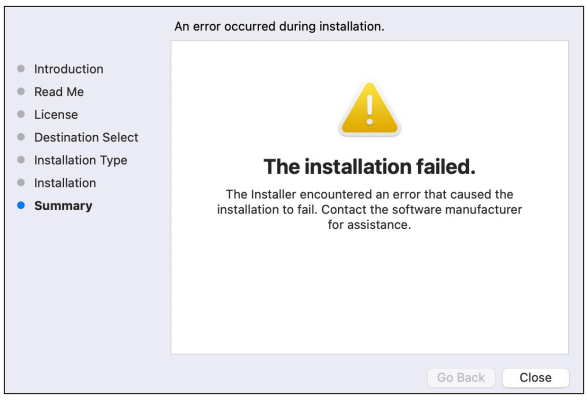
Amazon SAMCLI 错误：“/lib64/libc.so.6：找不到 `GLIBC_2.14'版本（ /usr/local/aws-sam-cli /dist/libz.so.1 需要）”

如果您收到此错误，则说明您使用的是不支持的 Linux 版本，并且内置的 glibc 版本已过时。试以下任一项任一项：

- 将你的 Linux 主机升级到 CentOS、Fedora、Ubuntu 或 Amazon Linux 2 的 64 位版本。
- 按照的说明进行操作安装 Amazon SAM CLI (p. 15)。

macOS

安装失败错误



如果您正在为用户安装Amazon SAM CLI 并选择了一个您没有写入权限的安装目录，则可能会出现此错误。试以下任一项任一项：

1. 选择您具有写入权限的其他安装目录。
2. 删除安装程序。然后，下载并再次运行它。

错误消息

curl 错误：“curl: (6) 无法解析:...”

在尝试调用 API Gateway 终端节点时，您会显示以下错误 API Gateway 终

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

这意味着您试图向无效的域名发送请求。如果您的无服务器应用程序未能成功部署，或者您的curl命令中有拼写错误，就会发生这种情况。使用Amazon CloudFormation控制台或，验证应用程序是否成功部署Amazon CLI，并验证您的curl命令是否正确。

错误：创建托管资源失败：找不到凭证

运行sam deploy命令时，您会看到以下错误：

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

这意味着您尚未设置Amazon凭据以使Amazon SAM CLI 进行Amazon服务调用。要解决这个问题，你必须设置Amazon证书。有关更多信息，请参阅设置Amazon证书 (p. 448)：

错误：pip 的依赖关系解析器...

错误示例文本：

```
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.
```

可能的原因：如果您使用安装软件包pip，则软件包之间的依赖关系可能会发生冲突。

该`aws-sam-cli`软件包的每个版本都依赖于该`aws-sam-translator`软件包的版本。例如，`aws-sam-cli`v1.58.0 可能依赖于`aws-sam-translator` v1.51.0。

如果您使用安装Amazon SAM CLIpip，然后安装另一个依赖于更新版本的软件包`aws-sam-translator`，则会出现以下情况：

- `aws-sam-translator`将安装较新版本的。
- 的当前版本`aws-sam-cli`和的更新版本`aws-sam-translator`可能不兼容。
- 当你使用Amazon SAM CLI 时，会出现依赖关系解析器错误。

解决方案：

1. 使用Amazon SAM CLI 原生软件包安装程序。
   a. Amazon SAMCLI 的权限 CLI 卸CLI。有关说明，请参阅 卸载Amazon SAM CLI (p. 444)。
   b. 使用本机软件包安装程序安装Amazon SAM CLI。有关说明，请参阅 安装 Amazon SAM CLI (p. 15)。
   c. 必要时，使用本机软件包安装程序升级Amazon SAM CLI。有关说明，请参阅 升级 CAmazon SAM LI (p. 443)。
2. 如果您必须使用pip，我们建议您将Amazon SAM CLI 安装到虚拟环境中。这样可以确保在出现错误时提供干净的安装环境和隔离的环境。有关说明，请参阅 使用以下方法将Amazon SAM CLI 安装到虚拟环境中pip (p. 447)。

错误：需要在本地运行Amazon SAM 项目Docker。你安装了吗？

运行sam local start-api命令时，您会看到以下错误：

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

这意味着你没有Docker正确安装。 Docker必须在本地测试应用程序。要解决此问题，请为开发主机的Docker er er Docker er docker。有关更多信息，请参阅安装 Docker (p. 455)：

错误：未满足安全限制

跑步时sam deploy --guided，系统会提示你提出问题`Function may not have authorization defined, Is this okay? [y/N]`。如果您以（默认响应）此提示**N**（默认响应），会显示以下错误：

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

该提示通知您，您要部署的应用程序可能在未经授权的情况下配置了可公开访问的 Amazon API Gateway API。**N**回应此提示，就是说这不行。

要解决此问题，您有以下选项：

- 使用授权配置您的应用程序。有关配置授权的信息，请参阅控制对 API Gateway API 的访问 (p. 330)。
- 如果您打算在未经授权的情况下使用可公开访问的 API 终端节点，请重新启动部署并回答此问题，**Y**以表明您可以进行部署。

消息：缺少身份验证令牌

在尝试调用 API Gateway 终端节点时，您会显示以下错误 API Gateway 终

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

这意味着你试图向正确的域名发送请求，但是 URI 不可识别。要修复此问题，请验证完整 URL，然后使用正确的 URL 更新curl命令。

Amazon SAM连接器参考

该部分包含连接器资源类型的参考信息。Amazon Serverless Application ModelAmazon SAM有关连接器的简介，请参阅使用Amazon SAM连接器管理资源权限 (p. 273)。

连接器支持的源和目标资源类型

`AWS::Serverless::Connector`资源类型支持源资源和目标资源之间一定数量的连接。在Amazon SAM模板中配置连接器时，使用下表引用支持的连接以及需要为每种源和目标资源类型定义的属性。有关在模板中配置连接器的更多信息，请参阅AWS::Serverless::Connector (p. 120)。

对于源资源和目标资源，在同一个模板中定义时，使用`Id`属性。或者，`Qualifier`可以添加以缩小已定义资源的范围。当资源不在同一个模板中时，请组合使用支持的属性。

要请求新连接，请在serverless-application-model Amazon GitHub存储库中提交新问题。

Source type（源类型）	目标类型	权限	源属性	目标属性
AWS::ApiGateway::RestApi	AWS::Lambda::Function	Write	Id或QualifierResource和Type	Id或者Arn和Type以及Type
AWS::ApiGateway::RestApi	AWS::Serverless::Function	Write	Id或QualifierResource和Type	Id或者Arn和Type以及Type
AWS::ApiGatewayV2::Api	AWS::Lambda::Function	Write	Id或QualifierResource和Type	Id或者Arn和Type以及Type
AWS::ApiGatewayV2::Api	AWS::Serverless::Function	Write	Id或QualifierResource和Type	Id或者Arn和Type以及Type
AWS::DynamoDB::Table	AWS::Lambda::Function	Read	Id或者Arn和Type	Id或者RoleName和Type
AWS::DynamoDB::Table	AWS::Serverless::Function	Read	Id或者Arn和Type	Id或者RoleName和Type
AWS::Events::Rule	AWS::Events::EventBus	Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::Events::Rule	AWS::Lambda::Function	Write	Id或者Arn和Type	Id或者Arn和Type
AWS::Events::Rule	AWS::Serverless::Function	Write	Id或者Arn和Type	Id或者Arn和Type
AWS::Events::Rule	AWS::Serverless::StateMachine	Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::Events::Rule	AWS::SNS::Topic	Write	Id或者Arn和Type	Id或者Arn和Type
AWS::Events::Rule	AWS::SQS::Queue	Write	Id或者Arn和Type	Id或ArnQueueUrl，以及Type
AWS::Events::Rule	AWS::StepFunctions::StateMachine	Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::Lambda::Function	AWS::DynamoDB::Table	Read, Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::Lambda::Function	AWS::Events::EventBus	Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::Lambda::Function	AWS::Lambda::Function	Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::Lambda::Function	AWS::Location::PlaceIndex	Read	Id或者RoleName和Type	Id或者Arn和Type
AWS::Lambda::Function	AWS::S3::Bucket	Read, Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::Lambda::Function	AWS::Serverless::Function	Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::Lambda::Function	AWS::Serverless::SimpleTable	Read, Write	Id或者RoleName和Type	Id或者Arn和Type

Source type（源类型）	目标类型	权限	源属性	目标属性
AWS::Lambda::Function	AWS::Serverless::StateMachine	Read, Write	Id或者RoleName和Type	Id或ArnName，以及Type
AWS::Lambda::Function	AWS::SNS::Topic	Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::Lambda::Function	AWS::SQS::Queue	Read, Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::Lambda::Function	AWS::StepFunctions::StateMachine	Read, Write	Id或者RoleName和Type	Id或ArnName，以及Type
AWS::S3::Bucket	AWS::Lambda::Function	Write	Id或者Arn和Type	Id或者Arn和Type
AWS::S3::Bucket	AWS::Serverless::Function	Write	Id或者Arn和Type	Id或者Arn和Type
AWS::Serverless::Api	AWS::Lambda::Function	Write	Id或QualifierResource或者Arn，以及Type	Id或者Arn和Type
AWS::Serverless::Api	AWS::Serverless::Function	Write	Id或QualifierResource或者Arn，以及Type	Id或者Arn和Type
AWS::Serverless::SimpleTable	AWS::DynamoDB::Table	Read, Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::Serverless::Function	AWS::Events::EventBus	Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::Serverless::Function	AWS::Lambda::Function	Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::Serverless::Function	AWS::S3::Bucket	Read, Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::Serverless::Function	AWS::Serverless::Function	Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::Serverless::Function	AWS::Serverless::SimpleTable	Read, Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::Serverless::Function	AWS::StepFunctions::StateMachine	Read, Write	Id或者RoleName和Type	Id或ArnName，以及Type
AWS::Serverless::Function	AWS::SNS::Topic	Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::Serverless::Function	AWS::SQS::Queue	Read, Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::Serverless::StateMachine	AWS::Lambda::Function	Write	Id或者RoleName和Type	Id或ArnName，以及Type
AWS::Serverless::StateMachine	AWS::SNS::Topic	Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::Serverless::StateMachine	AWS::SQS::Queue	Read, Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::Serverless::StateMachine	AWS::StepFunctions::StateMachine	Read, Write	Id或者RoleName和Type	Id或ArnName，以及Type
AWS::StepFunctions::Activity	AWS::Lambda::Function	Write	Id或QualifierResource或者Arn，以及Type	Id或者Arn和Type
AWS::StepFunctions::Activity	AWS::Serverless::Function	Write	Id或QualifierResource或者Arn，以及Type	Id或者Arn和Type

Source type（源类型）	目标类型	权限	源属性	目标属性
AWS::Serverless::SimpleTable	AWS::Lambda::Function	Read	Id或者Arn和Type	Id或者RoleName和Type
AWS::Serverless::SimpleTable	AWS::Serverless::Function	Read	Id或者Arn和Type	Id或者RoleName和Type
AWS::StateMachine	AWS::DynamoDB::Table	Read, Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::StateMachine	AWS::Events::EventBus	Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::StateMachine	AWS::Lambda::Function	Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::StateMachine	AWS::S3::Bucket	Read, Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::StateMachine	AWS::Serverless::Function	Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::StateMachine	AWS::Serverless::SimpleTable	Read, Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::StateMachine	AWS::StateMachine	Read, Write	Id或者RoleName和Type	Id或ArnName，以及Type
AWS::StateMachine	AWS::SNS::Topic	Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::StateMachine	AWS::SQS::Queue	Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::StepFunctions::StateMachine	AWS::StepFunctions::StateMachine	Read, Write	Id或者RoleName和Type	Id或ArnName，以及Type
AWS::SNS::Topic	AWS::Lambda::Function	Write	Id或者Arn和Type	Id或者Arn和Type
AWS::SNS::Topic	AWS::Serverless::Function	Write	Id或者Arn和Type	Id或者Arn和Type
AWS::SNS::Topic	AWS::SQS::Queue	Write	Id或者Arn和Type	Id或ArnQueueUrl，以及Type
AWS::SQS::Queue	AWS::Lambda::Function	Read, Write	Id或者Arn和Type	Id或者RoleName和Type
AWS::SQS::Queue	AWS::Serverless::Function	Read, Write	Id或者Arn和Type	Id或者RoleName和Type
AWS::StepFunctions::StateMachine	AWS::DynamoDB::Table	Read, Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::StepFunctions::StateMachine	AWS::Events::EventBus	Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::StepFunctions::StateMachine	AWS::Lambda::Function	Write	Id或者RoleName和Type	Id或者Arn和Type

Source type（源类型）	目标类型	权限	源属性	目标属性
AWS::StepFunctions::StateMachine	AWS::IAM::Role	Read, Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::StepFunctions::StateMachine	AWS::Serverless::Function	Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::StepFunctions::StateMachine	AWS::Serverless::SimpleTable	Read, Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::StepFunctions::StateMachine	AWS::StepFunctions::StateMachine	Read, Write	Id或者RoleName和Type	Id或ArnName，以及Type
AWS::StepFunctions::StateMachine	AWS::SNS::Topic	Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::StepFunctions::StateMachine	AWS::SQS::Queue	Write	Id或者RoleName和Type	Id或者Arn和Type
AWS::StepFunctions::StateMachine	AWS::StepFunctions::StateMachine	Read, Write	Id或者RoleName和Type	Id或ArnName，以及Type

安装 Docker 以便与Amazon SAM CLI 一起使用

Docker 是一个在你的机器上运行容器的应用程序。使用 Docker，Amazon SAM可以提供类似于容器的本地环境Amazon Lambda，用于构建、测试和调试您的无服务器应用程序。

> **Note**
>
> 只有在本地测试应用程序和使用`--use-container`选项构建部署包时才需要 Docker。

主题

安装 Docker

按照以下说明在操作系统上安装 Docker。

Linux

Docker 适用于许多不同的操作系统，包括大多数现代 Linux 分发版 (如 CentOS) 和 Ubuntu。有关在特定操作系统上安装 Docker 的信息，请参阅 Docker Docs 网站上的 Get Docker。

如果您使用的是亚马逊 Linux 2 (请求使用) 请按照下列步骤安装 Docker：

1. 更新实例上已安装的程序包和程序包缓存。

```
sudo yum update -y
```

2. 安装最新的 Docker Community Edition 程序包。

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

3. 启动 Docker 服务。

```
sudo service docker start
```

4. 将ec2-user添加到该docker组中，这样您就可以在不使用的情况下运行 Docker 命令sudo。

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

5. 注销并重新登录，即可获得新的docker群组权限。为此，请关闭当前的 SSH 终端窗口，然后在新窗口中重新连接到您的实例。您的新 SSH 会话应具有相应的docker群组权限。
6. 验证是否ec2-user能在不使用的情况下运行 Docker 命令sudo。

```
docker ps
```

您应看到以下输出，它表明了 Docker 成功，它表明了 Docker 成功：

```
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
             PORTS               NAMES
```

Note

在 Linux 上，要使用与主机不同的指令集架构构建和运行 Lambda 函数，需要执行其他步骤来配置 Docker。例如，要在x86_64计算机上运行arm64函数，可以运行以下命令来配置 Docker 守护程序：`docker run --rm --privileged multiarch/qemu-user-static --reset -p yes`。

如果您在安装 Docker 时遇到问题，请参阅安装错误 (p. 449)。或者，请参阅 Docker Docs 网站上的 Linux 安装后步骤的疑难解答部分。

macOS

Note

官方支持 Docker Desktop，但从Amazon SAM CLI 版本 1.47.0 开始，你可以使用替代方案，只要它们使用 Docker 运行时即可。

1. 安装 Docker

   Amazon SAMCLI 支持 Docker 在 macOS Sierra 10.12 或更高版本上运行。有关如何安装 Docker，请参阅 Dock er Docs 网站上的 Mac 版 Docker Des ktop。
2. 配置您的共享云端硬盘

   CAmazon SAM LI 要求在共享驱动器中列出项目目录或任何父目录。有关如何在 macOS 上共享驱动器，请参阅 Docker Docs 网站上的文件共享。
3. 验证安装

   安装 Docker 后，验证它是否正常运行。还要确认您可以从命令行运行 Docker 命令（例如，`docker ps`）。您无需安装、提取或提取任何容器，Amazon SAMCLI 会根据需要自动执行此操作。

如果您在安装 Docker 时遇到问题，有关更多故障排除提示，请参阅 Docker Docs 网站的 “疑难解答和诊断” 部分。

Windows

Note

Amazon SAM正式支持 Docker 桌面。但是，从Amazon SAM CLI 版本 1.47.0 开始，您可以使用替代方案，只要它们使用 Docker 运行时即可。

1. 安装 Docker.

   Docker Desktop 支持最新的 Windows 操作系统。对于旧版本的 Windows，Docker 工具箱可用。根据正确的 Docker 安装步骤选择你的 Windows 版本：

   - 要安装适用于 Windows 10 的 Docker，请参阅 Docker Docs 网站上的 “安装适用于 Windows 的 Docker Des ktop”。
   - 要安装适用于早期版本的 Windows 的 Docker，请参阅 Docker Toolb ox GitHub 存储库上的 Docker Toolbox。
2. 配置您的共享云端硬盘。

   CAmazon SAM LI 要求在共享驱动器中列出项目目录或任何父目录。在某些情况下，您必须共享驱动器才能让 Docker 正常运行。
3. 验证安装。

   安装 Docker 后，验证它是否正常运行。还要确认您可以从命令行运行 Docker 命令（例如，docker ps）。您无需安装、提取或提取任何容器，Amazon SAMCLI 会根据需要自动执行此操作。

如果您在安装 Docker 时遇到问题，有关更多故障排除提示，请参阅 Docker Docs 网站的 “疑难解答和诊断” 部分。

后续步骤

有关如何安装Amazon SAM CLI 的信息，请参阅安装 Amazon SAM CLI (p. 15)。

安装Homebrew以便与Amazon SAM CLI 一起使用

您可以选择使用Homebrew在 macOS 和 Linux 计算机上安装和管理Amazon Serverless Application Model命令行界面 (Amazon SAMCLI) 的版本。要使用Homebrew，请按照此处的安装说明进行操作。

主题

安装 Git

要安装 GitHomebrew，您必须先安装 Git。Git 适用于许多不同的操作系统，包括大多数现代 Linux 分发版和 MacOS。有关在特定操作系统上安装 Git 的说明，请参阅在 Git 网站上安装 Git。

安装 Homebrew

Linux

Note

安装Homebrew会将环境的默认 Python 版本更改为已Homebrew安装的版本。

成功安装 Git 后，要安装Homebrew，请运行以下命令：

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

接下来，通过运行以下命令将其Homebrew添加到您的 PATH 中。通过在 Debian 和 Ubuntu 上添加，或者~/.profile在 CentOS、Fedora 和 Red Hat~/.bash_profile 上添加这些命令，这些命令可以在所有主要版本的 Linux 上运行。

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

验证是否Homebrew已安装。

```
brew --version
```

成功安装后Homebrew，您应看到类似下面的输出：

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

macOS

成功安装 Git 后，运行以下命令进行安装Homebrew，确保按照提示进行操作：

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

验证是否Homebrew已安装：

```
brew --version
```

成功安装后Homebrew，您应看到类似下面的输出：

```
Homebrew 2.5.7
Homebrew/homebrew-core (git revision 1be3ad; last commit 2020-10-29)
Homebrew/homebrew-cask (git revision a0cf3; last commit 2020-10-29)
```

后续步骤

要安装Amazon SAM CLI，请参阅安装 Amazon SAM CLI (p. 15)。

镜像

Amazon SAM借助构建Container Re像，简化无服务器应用程序的持续集成和持续交付（CI/CD）。Amazon SAM提供的映像包括Amazon SAM命令行界面 (CLI) 和用于许多支持的Amazon Lambda运行时的构建工具。这使得使用Amazon SAM CLI 构建和打包无服务器应用程序变得更加容易。您可以将这些映像与 CI/CD 系统一起使用来自动构建和部署Amazon SAM应用程序。有关示例，请参阅 使用 CI/CD 系统部署 (p. 366)。

Amazon SAM构建容器镜像 URI 使用该映像中包含的Amazon SAM CLI 版本进行标记。如果您指定未加标签的 URI，则使用最新版本。例如，public.ecr.aws/sam/build-nodejs14.x使用最新的图像。但

是，public.ecr.aws/sam/build-nodejs14.x:1.24.1使用包含Amazon SAM CLI 版本 1.24.1 的图像。

从Amazon SAM CLI 的 1.33.0 版本开始，两个镜像x86_64和arm64容器镜像都可用于支持的运行时。有关更多信息，请参阅Amazon Lambda开发者指南中的 Lambda 运行时。

Note

在 CLI 版本 1.22.0 之前， DockerHub 是Amazon SAM CLI 从中Amazon SAM提取容器映像的默认存储库。从 1.22.0 开始，默认存储库更改为Amazon Elastic Container Registry Public (Amazon ECR Public) 要从当前默认存储库以外的存储库中提取容器映像，可以使用带--build-image选项的山姆·布莱德 (p. 408)命令。本主题末尾的示例显示了如何使用 DockerHub 存储库映像构建应用程序。

镜像

下表列出了 Amazon ECR Public 构建容器映像的 URI，您可以使用这些镜像来构建和打包无服务器应用程序Amazon SAM。

Note

Amazon ECR Public 从Amazon SAM CLI 版本 1.22.0DockerHub 开始取代。如果您正在使用Amazon SAM CLI 的早期版本，我们建议您升级。

运行时	Amazon ECR Public
.NET 7	public.ecr.aws/sam/build-dotnet7
.NET 6	public.ecr.aws/sam/build-dotnet6
.NET Core 3.1	public.ecr.aws/sam/build-dotnetcore3.1
Node.js 18	public.ecr.aws/sam/build-nodejs18.x
Node.js 16	public.ecr.aws/sam/build-nodejs16.x
Node.js 14	public.ecr.aws/sam/build-nodejs14.x
Node.js 12	public.ecr.aws/sam/build-nodejs12.x
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
Python 3.7	public.ecr.aws/sam/build-python3.7
Ruby 2.7	public.ecr.aws/sam/build-ruby2.7
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
Go 1.x	public.ecr.aws/sam/build-go1.x
自定义运行时 (AL2)	public.ecr.aws/sam/build-provided.al2
自定义运行时	public.ecr.aws/sam/build-Provid-

示例

以下两个示例命令使用 DockerHub 存储库中的容器映像构建应用程序：

使用从DockerHub以下位置提取的容器映像构建Node.js 12应用程序：

```
$ sam build --use-container --build-image amazon/aws-sam-cli-build-image-nodejs12.x
```

使用从DockerHub以下位置提取的Python 3.8容器镜像构建函数资源：

```
$ sam build --use-container --build-image Function1=amazon/aws-sam-cli-build-image-
python3.8
```

逐步部署无服务器应用程序

Amazon Serverless Application Model(Amazon SAM) 内置，可CodeDeploy提供渐进Amazon Lambda部署。只需几行配置，Amazon SAM即可为您完成以下操作：

- 部署您的 Lambda 函数的新版本，并自动创建指向新版本的别名。
- 逐渐将客户流量转移到新版本，直到您对其按预期运行感到满意。如果更新无法正常运行，则可以回滚更改。
- 定义流量前和流量后测试函数，以验证新部署的代码配置是否正确以及您的应用程序是否按预期运行。
- 如果触发 CloudWatch 警报，则自动回滚部署。

> **Note**
>
> 如果您通过Amazon SAM模板启用渐进部署，则会自动为您创建 CodeDeploy 资源。您可以直接通过查看 CodeDeploy 资源Amazon Web Services Management Console。

示例

以下示例演示了如何使用逐步 CodeDeploy 将客户转移到您新部署的 Lambda 函数版本：

```
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
```

对Amazon SAM模板的这些修订具有以下作用：

- `AutoPublishAlias`: 通过添加此属性并指定别名，Amazon SAM:
  - 根据 Lambda 函数的 Amazon S3 URI 的更改检测何时部署新代码。
  - 使用最新代码创建并发布该函数的更新版本。
  - 使用您提供的名称创建别名（除非别名已经存在），并指向 Lambda 函数的更新版本。函数调用应该使用别名限定词来利用这一功能。如果您不熟悉 Lambda 函数版本控制和别名，请参阅Amazon Lambda 函数版本控制和别名。
- `Deployment Preference Type`：在前面的示例中，10% 的客户流量会立即转移到您的新版本。10 分钟后，所有流量将转移到新版本。但是，如果您的流量前测试或流量后测试失败，或者触发了 CloudWatch 警报，则回 CodeDeploy 滚您的部署。您可以通过以下方式指定如何通过以下方式在版本之间转移流量：
  - `Canary`：流量在两次增量中转移。您可以从预定义的金丝雀选项中进行选择。这些选项指定在第一次增量中转移到更新的 Lambda 函数版本的流量百分比，以及剩余流量以第二个增量转移之前的间隔（以分钟为单位）。
  - `Linear`：流量使用相等的增量转移，在每次增量之间的分钟数相同。您可以从预定义的线性选项中进行选择，这些选项指定每次增量中转移的流量百分比以及每次增量之间的分钟数。
  - `AllAtOnce`：所有流量立即从原始 Lambda 函数转移到更新后的 Lambda 函数版本。

  下表概述了除示例中使用的流量转移选项以外的其他可用流量转移选项。

部署首选项类型
Canary10Percent30Minutes
Canary10Percent5Minutes
Canary10Percent10Minutes
Canary10Percent15Minutes
Linear10PercentEvery 10 10 分钟
Linear10PercentEvery 1 分钟
Linear10PercentEvery 2 分钟
Linear10PercentEvery 3 分钟
AllAtOnce

- `Alarms`：这些 CloudWatch 警报由部署引发的任何错误触发。遇到它们时，它们会自动回滚您的部署。例如，如果您部署的更新代码导致应用程序内出现错误。另一个示例是，您指定的任何Amazon Lambda或自定义 CloudWatch 指标是否已超过警报阈值。
- `Hooks`：这些是流量转移前和流量转移后的测试功能，它们在流量转移开始到新版本之前和流量转移完成之后运行检查。
  - `PreTraffic`：在流量转移开始之前， CodeDeploy 调用预流量挂钩 Lambda 函数。此 Lambda 函数必须回调 CodeDeploy 并指示成功或失败。如果函数失败，它将中止并将失败报告给Amazon CloudFormation。如果函数成功，则 CodeDeploy 继续进行流量转移。
  - `PostTraffic`：流量转移完成后， CodeDeploy 调用后流量挂钩 Lambda 函数。这与预流量挂钩类似，其中函数必须回调 CodeDeploy 才能报告成功或失败。使用转移流量后挂钩可以运行集成测试或其他验证操作。

  有关更多信息，请参阅 SAM 安全部署参考。

首次逐步部署 Lambda 函数

逐步部署 Lambda 函数时， CodeDeploy 需要先前部署的函数版本才能转移流量。因此，您的首次部署应分两步完成：

- 步骤 1：部署您的 Lambda 函数并使用自动创建别名`AutoPublishAlias`。
- 步骤 2：使用执行逐步部署`DeploymentPreference`。

分两步执行首次逐步部署，可以 CodeDeploy 从之前的 Lambda 函数版本转移流量。

第 1 步：部署 Lambda 函数

```
Resources:
MyLambdaFunction:
  Type: AWS::Serverless::Function
  Properties:
    Handler: index.handler
    Runtime: nodejs12.x
    CodeUri: s3://bucket/code.zip

    AutoPublishAlias: live
```

步骤 2：执行逐步部署

```
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
```

了解更多信息

有关配置逐步部署的实践示例，请参阅 “完整Amazon SAM研讨会” 中的模块 5-Canary 部署。

中的遥测Amazon SAMCLI

在Amazon，我们根据从与客户互动中学到的知识开发和推出服务。我们使用客户反馈来迭代我们的产品。遥测是附加信息，可帮助更好地了解客户需求、诊断问题并提供功能，以改善客户体验。

这些区域有：Amazon SAM命令行接口 (CLI) 收集遥测数据，例如通用使用指标、系统和环境信息以及错误。有关所收集遥测类型的详细信息，请参阅收集的信息类型 (p. 463).

这些区域有：Amazon SAMCLI 确实如此不收集个人信息，例如用户名或电子邮件地址。它也不会提取敏感的项目级信息。

客户控制是否开启遥测功能，他们可以随时更改设置。如果遥测保持开启状态，Amazon SAMCLI 在后台发送遥测数据，无需任何额外的客户互动。

关闭会话的遥测功能

在 macOS 和 Linux 操作系统中，您可以关闭单个会话的遥测功能。要关闭当前会话的遥测，请运行以下命令，以便为环境变量SAM_CLI_TELEMETRY到false. 对每个新终端或会话重复使用以下命令。

```
export SAM_CLI_TELEMETRY=0
```

在所有会话中关闭个人资料的遥测功能

运行以下命令以在运行时关闭所有会话的遥测功能Amazon SAM您的操作系统上的 CLI。

在 Linux 中关闭遥测

1. 运行：

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

2. 运行：

   ```
   source ~/.profile
   ```

在 macOS 中关闭遥测

1. 运行：

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

2. 运行：

   ```
   source ~/.profile
   ```

在 Windows 中关闭遥测

1. 运行：

   ```
   setx SAM_CLI_TELEMETRY 0
   ```

2. 运行：

   ```
   refreshenv
   ```

收集的信息类型

- 用量信息— 客户运行的通用命令和子命令。

- 错误和诊断信息— 客户运行的命令的状态和持续时间，包括退出代码、内部异常名称以及连接到 Docker 时的故障。
- 系统和环境信息— Python 版本、操作系统（Windows、Linux 或 macOS）、其中的环境Amazon SAMCLI 运行（例如Amazon CodeBuild，一个AmazonIDE 工具包或终端）和使用属性的哈希值。

了解更多信息

的遥测数据Amazon SAMCLI 收集的遵守情况是Amazon数据隐私政策。有关更多信息，请参阅下列内容：

- Amazon服务条款
- 数据隐私常见问题

重要提示

本节包含以下方面的重要说明和已知问题Amazon Serverless Application Model。

在 32 位 Windows 上安装Amazon SAM CLI

对 32 位 Windows 上的Amazon SAM CLI 的Support 将很快被弃用。如果您在 32 位系统上运行，我们建议您升级到 64 位系统并按照中的说明进行操作安装 Amazon SAM CLI (p. 15)。

如果您无法升级到 64 位系统，则可以在 32 位系统上使用带有Amazon SAM CLI 的 L egacy Docker Too lbox。但是，这将导致您在Amazon SAM使用 CLI 时遇到某些限制。例如，不能在 32 位系统上运行 64 位 Docker 容器。因此，如果您的 Lambda 函数依赖于 64 位原生编译的容器，则您将无法在 32 位系统上对其进行本地测试。

要在 32 位系统上安装Amazon SAM CLI，请执行以下命令：

```
pip install aws-sam-cli
```

Important

尽管该`pip install aws-sam-cli`命令也适用于 64 位 Windows，但我们建议您使用 64 位 MSI 在 64 位系统上安装Amazon SAM CLI。

Amazon SAM 的文档历史记录

下表描述每个版本的《Amazon Serverless Application Model开发人员指南》中所做的重要更改。如需有关此文档更新的通知，您可以订阅 RSS 源。

- 文档最新更新时间：2023 年 3 月 23 日

变更	说明	日期
添加 sam 同步选项以跳过基础架构同步 (p. 465)	自定义每次运行时`sam sync`是否都需要Amazon CloudFormation部署。要了解更多信息，请参阅跳过初始部Amazon CloudFormation署。	2023 年 3 月 23 日
添加对 DocumentDB 事件源类型的支持 (p. 465)	Amazon SAM模板规范现在支持`AWS::Serverless::Function`资源`DocumentDB`的事件源类型。要了解更多信息，请参阅DocumentenDB。	2023 年 3 月 10 日
使用 Rust 编译 Lambda 函数 Cargo Lambda (p. 465)	使用Amazon SAM CLI 使用以下命令来构建 Rust Lambda 函数Cargo Lambda。要了解更多信息，请参阅使用构建 Rust Lambda 函数Cargo Lambda。	2023 年 2 月 23 日
在外部构建函数资源Amazon SAM (p. 465)	添加了有关在使用sam build命令时跳过函数的指南。要了解更多信息，请参阅在之外构建函数Amazon SAM。	2023 年 2 月 14 日
新的嵌入式连接器语法 (p. 465)	使用新的嵌入式连接器语法来定义您的`AWS::Serverless::Connector`资源。要了解更多信息，请参阅使用Amazon SAM连接器管理资源权限。	2023 年 2 月 8 日
为Amazon SAM CLI 添加了新的 sam list 命令 (p. 465)	sam list用于查看有关无服务器应用程序中资源的重要信息。要了解更多信息，请参阅相同的列表。	2023 年 2 月 2 日
为 esb OutExtension uild 添加了格式和编译属性 (p. 465)	使用 esbuild 构建 Node.js Lambda 函数现在支持`Format`和`OutExtension`构建属性。要了解更多信息，请参阅使用 esbuild 构建 Node.js Lambda 函数。	2023 年 2 月 2 日
在Amazon SAM模板规范中添加了运行时管理选项 (p. 465)	为您的 Lambda 函数配置运行时管理选项。要了解更多信息，请参阅`RuntimeManagementConfig`。	2023 年 1 月 24 日

EventSource 为 AWS::Serverless::StateMachine 资源添加了目标属性。 (p. 465)	AWS::Serverless::StateMachine 资源类型支持EventBridgeRule和Schedule事件源的Target属性。	2023 年 1 月 13 日
为 Lambda 函数配置 SQS 轮询器的扩展性 (p. 465)	使用的ScalingConfig属性配置 SQS 轮询器的缩放比例AWS::Serverless::Function。要了解更多信息，请参阅ScalingConfig。	2023 年 1 月 12 日
使用 cfn-lint 验证Amazon SAM应用程序 (p. 465)	你可以使用 cfn-lint 通过Amazon SAM CLI 验证你的Amazon SAM 模板。要了解更多信息，请参阅使用 cfn-lint 进行验证。	2023 年 1 月 11 日
使用应用程序洞察监控您的无服务器 CloudWatch 应用程序 (p. 465)	配置 Amazon CloudWatch 应用程序洞察以监控您的Amazon SAM 应用程序。要了解更多信息，请参阅使用 Application Ins ights 监控您的无服务器 CloudWatch 应用程序。	2022 年 12 月 19 日
添加了适用于 macOS 的Amazon SAM CLI 包安装程序 (p. 465)	使用新的 macOS 软件包安装程序安装Amazon SAM CLI。要了解更多信息，请参阅安装Amazon SAM CLI。	2022 年 12 月 6 日
增加了对 Lambda 的支持 SnapStart (p. 465)	SnapStart 为您的 Lambda 函数进行配置以创建快照，快照是您初始化函数的缓存状态。要了解更多信息，请参阅AWS::Serverless::Function。	2022 年 11 月 28 日
添加了有关配置访问和权限的指南 (p. 465)	Amazon SAM提供了两个选项，可简化无服务器应用程序的访问和权限管理。要了解更多信息，请参阅管理资源访问和权限。	2022 年 11 月 17 日
添加了Amazon SAM对 nodejs18.x 的 CLI 支持 (p. 465)	Amazon SAMCLI 现在支持 nodejs18.x 运行时。要了解更多信息，请参阅 sam init。	2022 年 11 月 17 日
增加了对使用原生 AOT 编译构建.NET 7 Lambda 函数的支持 (p. 465)	使用原生提前编译 (AOT) 编译来构建和打包.NET 7 Lambda 函数，缩短 Lambda 冷启动时间。Amazon SAM要了解更多信息，请参阅使用原生 AOT 编译来构建 .NET 7 Lambda 函数。	2022 年 11 月 15 日
添加了Amazon SAM CLI Terraform 对本地调试和测试的支持 (p. 465)	在您的 Terraform 项目中使用Amazon SAM CLI 对您的 Lambda 函数和层进行本地调试和测试。要了解更多信息，请参阅 Amazon SAMCLI Terraform 支持。	2022 年 11 月 14 日

增加了对 EventBridge 调度程序的Amazon SAM支持 (p. 465)	Amazon Serverless Application Model(Amazon SAM) 模板规范提供了一种简单、简短的语法，您可以使用该语法通过 S EventBridge cheduler 为Amazon Lambda和调度事件Amazon Step Functions。有关更多信息，请参阅使用S EventBridge cheduling。	2022 年 11 月 10 日
简化了 CAmazon SAM LI 安装说明 (p. 465)	Amazon SAMCLI 必备条件和可选步骤已移至单独的页面。支持的操作系统的安装步骤可以在安装Amazon SAM CLI 中找到。	2022 年 11 月 4 日
添加了允许 Windows 10 用户使用长路径的修复程序 (p. 465)	Amazon SAMCLI 应用程序模板存储库包含一些长文件路径，由`sam init`于 Windows 10 的MAX_PATH限制，这些路径可能会在运行时导致错误。有关更多信息，请参阅安装Amazon SAM CLI。	2022 年 11 月 4 日
更新了首次部署的逐步部署流程 (p. 465)	逐步部署 Lambda 函数Amazon CodeDeploy需要两个步骤。要了解更多信息，请参阅首次逐步部署 Lambda 函数。	2022 年 10 月 13 日
对更多类型事件的额外 Lambda 事件筛选支持 (p. 465)	`FilterCritera`属性已添加到`MSKMQ`、和`SelfManagedKafka`事件源类型。	2022 年 10 月 13 日
增加了对Amazon SAM管道的 OpenID Connect（OIDC）。 (p. 465)	Amazon SAM支持 Bitbucket、Ac GitHub tions、 GitLab 持续集成和持续交付（CI/CD）平台的 OpenID Connect（OIDC）用户身份验证。要了解更多信息，请参阅在Amazon SAM管道中使用 OIDC 用户帐户。	2022 年 10 月 13 日
关于 JwtConfiguration 属性的说明 (p. 465)	在 for 下添加了关于定义`issuer`和`audience`属`JwtConfiguration`性的注释`OAuth2Authorizer`。	2022 年 10 月 7 日
函数和的新属性 StateMachine EventSource (p. 465)	`Enabled`并将`State`属性添加到`CloudWatchEvent`的事件源中`AWS::Serverless::Function`。`State`已将属性添加到`AWS::Serverless::Function`和`Schedule`的事件源中`AWS::Serverless::StateMachine`。	2022 年 10 月 6 日

Amazon SAM连接器现已正式上市 (p. 465)	连接器是一种Amazon SAM抽象资源类型，标识为`AWS::Serverless::Connector`，它提供了一种在无服务器应用程序资源之间配置权限的简单而安全的方法。要了解更多信息，请参阅使用Amazon Serverless Application Model连接器管理资源权限。	2022 年 10 月 6 日
在Amazon SAM CLI 中添加了新的 sam 同步选项 (p. 465)	`--dependency-layer`并将`--use-container`选项添加到sam sync。	2022 年 9 月 20 日
在Amazon SAM CLI 中添加了新的 sam 部署选项 (p. 465)	`--on-failure`选项已添加到sam deploy。	2022 年 9 月 9 日
esbuild 支持现已正式发布 (p. 465)	要构建和打包 Node.js Lambda 函数，你可以将Amazon SAM命令行界面与 esbuild JavaScript 打包器一起使用。	2022 年 9 月 1 日
更新了Amazon SAM CLI 遥测 (p. 465)	收集的系统和环境信息的描述已更新，包括使用属性的哈希值。	2022 年 9 月 1 日
向Amazon SAM CLI 添加了本地环境变量支持 (p. 465)	在本地调用 Lambda 函数和在本地运行 APAmazon SAM l Gateway 时，在 CLI 中使用环境变量。	2022 年 9 月 1 日
Support Lambda 指令集架构 (p. 465)	使用Amazon SAM CLI 为我们的`arm64`指令集架构构构构建 Lambda 函数和 Lambda 层。`x86_64`有关更多信息，请参阅`AWS::Serverless::Function`资源类型的架构属性和`AWS::Serverless::LayerVersion`资源类型的CompatibleArchitectures属性。	2021 年 10 月 1 日
生成管道配置示例 (p. 465)	使用Amazon SAM CLI 使用新的`sam pipeline bootstrap`和`sam pipeline init`命令为多个 CI/CD 系统生成示例管道。有关更多信息，请参阅生成示例 CI/CD 管道。	2021 年 7 月 21 日
Amazon SAMCLIAmazon CDK 集成（预览版，第 2 阶段）(p. 465)	在公共预览版的第 2 阶段中，您现在可以使用Amazon SAM CLI 打包和部署Amazon CDK应用程序。您也可以使用Amazon SAM CLI 直接下载示例Amazon CDK应用程序。有关更多信息，请参阅Amazon Cloud Development Kit (Amazon CDK)（预览）。	2021 年 7 月 13 日

Support RabbitMQ 作为函数的事件源 (p. 465)	添加了支持 RabbitMQ 作为无服务器函数的事件源。有关更多信息，请参阅`AWS::Serverless::Function`资源类型`MQ`的事件源的`SourceAccessConfigurations`属性。	2021 年 7 月 7 日
使用 Amazon ECR 构建容器映像部署无服务器应用程序 (p. 465)	使用 Amazon ECR 构建容器映像来部署带有常见 CI/CD 系统（例如Amazon CodePipeline、Jenkins、C GitLab I/CD 和 Ac GitHub tions）的无服务器应用程序。有关更多信息，请参阅部署无服务器应用程序。	2021 年 6 月 24 日
使用Amazon工具包调试Amazon SAM应用程序 (p. 465)	AmazonToolkit 现在支持分步调试，集成开发环境 (IDE) 和运行时的更多组合。有关更多信息，请参阅使用Amazon工具包。	2021 年 5 月 20 日
Amazon SAMCLIAmazon CDK 集成（预览版）(p. 465)	现在，您可以使用Amazon SAM CLI 在本地测试和构建Amazon CDK应用程序。这是公共预览版。有关更多信息，请参阅 Amazon Cloud Development Kit (Amazon CDK)（预览）。	2021 年 4 月 29 日
默认容器镜像存储库更改为 Amazon ECR 公共镜像存储库 (p. 465)	默认容器镜像存储库从 DockerHub 更改为 Amazon ECR Public。有关更多信息，请参阅镜像存储库。	2021 年 4 月 6 日
每晚Amazon SAM CLI 构建 (p. 465)	现在，您可以安装每晚构建的 Amazon SAM CLI 的预发行版本。有关更多信息，请参阅 “安装Amazon SAM CLI” 下您选择的操作系统子主题的 “夜间构建” 部分。	2021 年 3 月 25 日
构建容器环境变量支持 (p. 465)	现在，您可以传递环境变量来构建容器。有关更多信息，请参阅中的`--container-env-var`和`--container-env-var-file`选项`sam build`。	2021 年 3 月 4 日
新的 Linux 安装流程 (p. 465)	现在，您可以使用本机 Linux 安装程序安装 CLAmazon SAM I。有关更多信息，请参阅在Amazon SAM Linux。	2021 年 2 月 10 日

Support 的死信队列 EventBridge (p. 465)	增加了对无服务器函数 EventBridge 和状态机的死信队列和Schedule事件源的支持。有关更多信息，请参阅EventBridgeRule和AWS::Serverless::StateMachine资源类型的和Schedule事件源的DeadLetterConfig属性。AWS::Serverless::Function	2021 年 1 月 29 日
Support 自定义检查点 (p. 465)	增加了对 DynamoDB 的自定义检查点和无服务器函数的 Kinesis 事件源的支持。有关更多信息，请参阅AWS::Serverless::Function资源类型的FunctionResponseTypes属Kinesis性和DynamoDB数据类型。	2021 年 1 月 29 日
Support 翻滚窗户 (p. 465)	增加了对 DynamoDB 的滚动窗口和无服务器函数的 Kinesis 事件源的支持。有关更多信息，请参阅AWS::Serverless::Function资源类型的TumblingWindowInSeconds属Kinesis性和DynamoDB数据类型。	2020 年 12 月 17 日
Support 温水箱 (p. 465)	在使用Amazon SAM CLI 命令sam local start-api和进行本地测试时，增加了对温容器的支持sam local start-lambda。有关更多信息，请参阅这些命令的--warm-containers选项。	2020 年 12 月 16 日
Support Lambda 容器镜像 (p. 465)	添加了对 Lambda 容器镜像的支持。有关更多信息，请参阅构建应用程序。	2020 年 12 月 1 日
Support 代码签名 (p. 465)	增加了对无服务器应用程序代码的代码签名和可信部署的支持。有关更多信息，请参阅为Amazon SAM应用程序配置代码签名。	2020 年 11 月 23 日
Support parallel 和缓存构建 (p. 465)	通过在sam build命令中添加两个选项来提高无服务器应用程序构建的性能：--parallel，它以parallel 而不是按顺序构建函数和--cached层；以及在未进行任何需要重建的更改时，使用先前版本中的构建工件。	2020 年 11 月 10 日

Support Amazon MQ 和双向 TLS 身份验证 (p. 465)	增加了对 Amazon MQ 作为无服务器函数事件源的支持。有关更多信息，请参阅AWS::Serverless::Function资源类型的EventSource和MQ数据类型。还增加了对 API Gateway API 和 HTTP API 的相互传输层安全 (TLS) 身份验证的支持。有关更多信息，请参阅AWS::Serverless::Api资源类型的DomainConfiguration数据类型或AWS::Serverless::HttpApi资源类型的HttpApiDomainConfiguration数据类型。	2020 年 11 月 5 日
Support Lambda 授权方 (p. 465)	添加了对AWS::Serverless::HttpApi资源类型的 Lambda 授权者的支持。有关更多信息，请参阅 Lambda 授权器示例 (AWS::Serverless::HttpApi)。	2020 年 10 月 27 日
Support 多个配置文件和环境 (p. 465)	增加了对多个配置文件和环境的支持，以存储Amazon SAM CLI 命令的默认参数值。有关，请参阅Amazon SAMCLI。	2020 年 9 月 24 日
在控制 API 访问权限时Support 带有 Step Functions 的 X-Ray 和引用 (p. 465)	添加了对 X-Ray 作为无服务器状态机事件源的支持。有关更多信息，请参阅AWS::Serverless::StateMachine资源类型的Tracing属性。在控制 API 访问权限时，还增加了对引用的支持。有关更多信息，请参阅ResourcePolicyStatement数据类型。	2020 年 9 月 17 日
对亚马逊 MSK 的 Support (p. 465)	增加了对 Amazon MSK 作为无服务器函数事件源的支持。这允许亚马逊 MSK 主题中的记录触发您的 Lambda 函数。有关更多信息，请参阅AWS::Serverless::Function资源类型的EventSource和MSK数据类型。	2020 年 8 月 13 日
Support Amazon EFS (p. 465)	增加了对将 Amazon EFS 文件系统安装到本地目录的支持。这允许您的 Lambda a 函数代码访问和修改共享资源。有关更多信息，请参阅AWS::Serverless::Function资源类型的FileSystemConfigs属性。	2020 年 6 月 16 日

协调无服务器应用程序 (p. 465)	通过使用创建 Step Functions 状态机，增加了对编排应用程序的支持Amazon SAM。有关更多信息，请参阅使用Amazon Step Functions和Amazon资源类型编排`AWS::Serverless::StateMachine`资源。	2020 年 5 月 27 日
构建自定义运行时 (p. 465)	添加了用于构建自定义运行时的功能。有关更多信息，请参阅构建自定义运行时。	2020 年 5 月 21 日
建筑物图层 (p. 465)	添加了用于构建个人`LayerVersion`资源的功能。有关更多信息，请参阅建筑层。	2020 年 5 月 19 日
生成的Amazon CloudFormation资源 (p. 465)	提供了有关Amazon SAM生成的Amazon CloudFormation资源以及如何引用它们的详细信息。有关更多信息，请参阅生成的Amazon CloudFormation资源。	2020 年 4 月 8 日
设置Amazon证书 (p. 465)	如果您尚未将Amazon证书设置为与其他Amazon工具（例如其中一个Amazon SDK 或）一起使用，则添加了设置凭据的说明Amazon CLI。有关，请参阅设置Amazon证书。	2020 年 1 月 17 日
Amazon SAM规格和Amazon SAM CLI 更新 (p. 465)	从迁移了Amazon SAM规范GitHub。有关更多信息，请参阅Amazon SAM规范。还通过更改`sam deploy`命令更新了部署工作流程。	2019 年 11 月 25 日
用于控制 API Gateway API 访问权限的新选项和策略模板更新 (p. 465)	添加了用于控制 API Gateway API 访问权限的新选项：IAM 权限、API 密钥和资源策略。有关更多信息，请参阅控制对 API Gateway API。还更新了两个策略模板： RekognitionFacesPolicy 和 ElasticsearchHttpPostPolicy。有关更多信息，请参阅Amazon SAM策略模板。	2019 年 8 月 29 日
入门更新 (p. 465)	更新了入门章节，改进了Amazon SAM CLI 和 Hello World 教程的安装说明。有关更多信息，请参阅Amazon SAM 入门。	2019 年 7 月 25 日
控制对 API Gateway I 的访问 (p. 465)	添加了对控制 API Gateway API 访问的支持。有关更多信息，请参阅控制对 API Gateway API。	2019 年 3 月 21 日

已添加sam publish到 CAmazon SAM LI (p. 465)	Amazon SAM CLI 中的新 sam publish 命令简化了在 Amazon Serverless Application Repository 中发布无服务器应用程序的过程。有关更多信息，请参阅使用 Amazon SAM CLI 发布无服务器应用程序。	2018 年 12 月 21 日
嵌套应用程序和层支持 (p. 465)	增加了对嵌套应用程序和层的支持。有关更多信息，请参阅使用嵌套应用程序和处理图层。	2018 年 11 月 29 日
已添加sam build到 CAmazon SAM LI (p. 465)	Amazon SAMCLI 中的新sam build命令简化了编译具有依赖关系的无服务器应用程序的过程，因此您可以在本地测试和部署这些应用程序。有关更多信息，请参阅构建应用程序。	2018 年 11 月 19 日
为Amazon SAM CLI 添加了新的安装选项 (p. 465)	为Amazon SAM CLI 添加了 Linuxbrew (Linux)、MSI (Windows) 和 Homebrew (macOS) 安装选项。有关更多信息，请参阅安装 Amazon SAM CLI。	2018 年 11 月 7 日
新指南 (p. 465)	这是 Amazon Serverless Application Model 开发人员指南的首次发布。	2018 年 10 月 17 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。