Amazon Serverless Application Model

开发人员指南

亚马逊云科技

Amazon Serverless Application Model: 开发人员指南

Amazon Web Services 文档中描述的 Amazon Web Services 服务或功能可能因区域而异。要查看适用于中国区域的差异，请参阅中国的 Amazon Web Services 服务入门。

Table of Contents

什么是Amazon Serverless Application Model(Amazon SAM) ?

这些区域有：Amazon Serverless Application Model(Amazon SAM) 是一个开源框架，可用于构建无服务器应用程序上Amazon.

一个无服务器应用是 Lambda 函数、事件源以及共同执行任务的其他资源的组合。请注意，无服务器应用程序不仅仅是一个 Lambda 函数，还可以包含其他资源，例如 API、数据库和事件源映射。

您可以使用Amazon SAM以定义无服务器应用程序。Amazon SAM包括以下组件：

- Amazon SAM模板规格. 您可以使用此规范定义无服务器应用程序。它为您介绍构成无服务器应用程序的函数、API、权限、配置和事件的介绍。你使用Amazon SAM模板文件，可在单个可部署的版本控制实体上操作，这是您的无服务器应用程序。对于完全Amazon SAM模板规格，请参阅Amazon Serverless Application Model(Amazon SAM) 规范 (p. 26).

- Amazon SAM命令行界面 (Amazon SAMCLI). 您可以使用此工具构建无服务器应用程序，这些应用程序由Amazon SAM模板。CLI 提供的命令使您能够验证Amazon SAM模板文件是根据规范编写的，在本地调用 Lambda 函数，逐步调试 Lambda 函数，打包和部署无服务器应用程序到Amazon云等。有关如何使用Amazon SAMCLI，包括完整的Amazon SAMCLI 命令参考，请参阅Amazon SAMCLI 命令参考 (p. 251).

本指南向您演示如何使用Amazon SAM定义、测试和部署简单的无服务器应用程序。它还提供了例应用程序 (p. 15)您可以在本地下载、测试并部署到Amazon云。您可以使用此示例应用程序作为开发自己的无服务器应用程序的起点。

使用 Amazon SAM 的好处

由于Amazon SAM与其他集成Amazon服务，创建无服务器应用程序Amazon SAM具有以下优势：

- 单部署配置.Amazon SAM使您能够轻松组织相关组件和资源，并在单一堆栈上进行操作。您可以使用Amazon SAM在资源之间共享配置（例如内存和超时），并将所有相关资源作为单个版本控制的实体一起部署。

- 扩展Amazon CloudFormation. 由于Amazon SAM是的扩展Amazon CloudFormation，您将获得可靠的部署功能Amazon CloudFormation. 你可以通过使用Amazon CloudFormation在您的Amazon SAM模板。此外，您还可以使用全套资源、内在函数和其他模板功能，这些功能可用于Amazon CloudFormation.

- 内置的最佳实践. 您可以使用Amazon SAM定义和部署基础设施作为配置。这使您可以使用和实施最佳做法，例如代码审查。此外，通过几行配置，您可以通过 CodeDeploy 启用安全部署，还可以通过使用Amazon X-Ray.

- 本地调试和测试. 这些区域有：Amazon SAMCLI 允许您在本地构建、测试和调试无服务器应用程序，这些应用程序由Amazon SAM模板。CLI 在本地提供了类似于 Lambda 的执行环境。它通过提供与实际 Lambda 执行环境的奇偶校验来帮助您提前发现问题。要逐步调试你的代码以了解代码在做什么，你可以使用Amazon SAM和Amazon类似的工具包Amazon Toolkit for JetBrains、AmazonToolkit for

pyCharm、AmazonToolkit for IntelliJ, 和AmazonToolkit for Visual Studio Code. 这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。

- 与开发工具的深度集成. 您可以使用Amazon SAM有一套Amazon构建无服务器应用程序的工具。你可以在Amazon Serverless Application Repository. 用于创作、测试和调试Amazon SAM— 基于无服务器的应用程序，你可以使用Amazon Cloud9IDE. 要为无服务器应用程序构建部署管道，您可以使用CodeBuild、CodeDeploy, 和CodePipeline. 您还可以使用AWS CodeStar开始使用自动为您配置的项目结构、代码库和 CI/CD 管道。要部署无服务器应用程序，您可以使用Jenkins 插件.

下一步

开始使用 Amazon SAM (p. 3)

开始使用 Amazon SAM

开始使用Amazon SAM，使用Amazon SAMCLI 来创建无服务器应用程序，您可以在Amazon云。你可以在Amazon云端或本地在您的开发主机上。

安装Amazon SAMCLI，包括所有需要安装或配置才能使用Amazon SAMCLI，请参阅安装 Amazon SAM CLI (p. 3). AfterAmazon SAMCLI 已安装完毕，你可以运行以下教程。

主题

安装 Amazon SAM CLI

Amazon SAM为您提供命令行工具、Amazon SAMCLI，可让您轻松创建和管理无服务器应用程序。你需要安装和配置一些东西才能使用Amazon SAMCLI。

安装Amazon SAMCLI，请参阅下列适用于您的开发主机的说明：

主题

安装Amazon SAMLinux 上的 CLI

这些区域有：Amazon SAM最近发行的 CentOS、Fedora、Ubuntu 和 Amazon Linux 2 的 64 位版本支持命令行界面 (CLI)。安装Amazon SAMCLI，您必须提取或“解压缩”下载的程序包。如果您的操作系统没有内置的 unzip 命令，请使用等效命令。

安装和配置使用Amazon SAM在 Linux 主机上执行 CLI，请执行以下步骤：

1. 创建一个 Amazon 账户。
2. 配置Amazon Identity and Access Management(IAM) 权限和Amazon凭证。
3. 安装 Docker. 注意：Docker 只是在本地测试应用程序或使用`--use-container`选项。
4. 安装 Amazon SAM CLI。

第 1 步：创建 Amazon 账户

如果您还没有Amazon账户，请参阅aws.amazon.com然后选择创建Amazon账户. 有关详细说明，请参阅如何创建和激活新的Amazon账户？

第 2 步：配置 IAM 权限和Amazon证书

与之一使用的 IAM 用户Amazon SAM必须有足够的权限才能进行必要Amazon服务呼叫和管理Amazon资源的费用。确保用户拥有足够权限的最简单方法是向他们授予管理员权限。有关更多信息，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。

Note

如果您不想向使用Amazon Command Line Interface(Amazon CLI)，您可以向他们授予受限的权限集。有关更多信息，请参阅 权限 (p. 330)。

此外，要启用Amazon SAM要做的 CLIAmazon您必须设置服务电话Amazon凭证。有关更多信息，请参阅 设置Amazon证书 (p. 14)。

第 3 步：安装 Docker（可选）

Note

只有在本地测试应用程序以及使用--use-container选项。如果您最初不打算使用这些功能，则可以跳过本节或稍后安装 Docker。

Docker 是在 Linux 计算机上运行容器的应用程序。Amazon SAM提供了类似于Amazon Lambda用作 Docker 容器。您可以使用此容器构建、测试和调试无服务器应用程序。

使用本地运行无服务器项目和功能Amazon SAMCLI，你必须安装 Docker 并正常工作。这些区域有：Amazon SAMCLI 使用`DOCKER_HOST`与 Docker 守护程序联系环境变量。以下步骤介绍如何安装、配置和验证 Docker 安装以使用。Amazon SAMCLI。

Docker 适用于许多不同的操作系统，包括大多数现代 Linux 发行版，例如 CentOS、Debian 和 Ubuntu。有关在特定操作系统上安装 Docker 的信息，请参阅获取 Docker在 Docker 文档网站上。

如果您使用 Amazon Linux 2，请按照以下步骤安装 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. 添加`ec2-user`到`docker`组，以便您可以在不使用的情况下运行 Docker 命令sudo.

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

5. 拿起新的`docker`通过注销然后重新登录来分组权限。为此，请关闭当前的 SSH 终端窗口并在新终端窗口中重新连接到实例。您的新 SSH 会话应具有相应的`docker`组权限。
6. 验证是否`ec2-user`可以运行 Docker 命令而不使用sudo.

```
docker ps
```

您应看到以下输出，确认已安装并运行 Docker：

```
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，必须采取其他步骤来配置 Docker。例如，要运行`arm64`在上的函数`x86_64`计算机上，您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，请参阅故障排除 (p. 7)本指南后面的部分。或者，请参阅故障排除的部分Linux 的安装后步骤在 Docker 文档网站上。

第 4 步：安装Amazon SAMCLI

安装Amazon SAMCLI 上，请按照以下步骤操作：

x86_64

1. 下载Amazon SAMCLI zip 格式文件到您选择的目录中。
2. 使用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
   ```

   输出应与以下示例类似：

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

   将 64 个字符的 SHA256 哈希值与所需的哈希值进行比较Amazon SAM中的 CLI 版本Amazon SAMCLI 发布说明（位于 GitHub 上）。
3. 将安装文件解压到`sam-installation/`子目录。

   ```
   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
   ```

   在成功安装后，您应看到类似以下内容的输出：

   ```
   SAM CLI, version 1.18.0
   ```

ARM

1. 使用`pip`要安装Amazon SAMCLI。

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

   ```
   sam --version
   ```

   在成功安装后，您应看到类似以下内容的输出：

   ```
   SAM CLI, version 1.18.0
   ```

现在您已准备就绪，可以开始开发。

升级

升级Amazon SAMCLI、执行与中相同的步骤安装Amazon SAMCLI本主题前面的部分，但是添加--update安装命令的选项，如下所示：

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

卸载

卸载Amazon SAMCLI，必须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。
   - 使用查找符号链接which命令：

     ```
     which sam
     ```

     输出将显示的路径Amazon SAM例如，二进制文件位于：

     ```
      /usr/local/bin/sam
     ```
   - 使用查找符号链接指向的目录ls命令：

     ```
     ls -l /usr/local/bin/sam
     ```

     在下面的示例中，安装目录是`/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
   ```

每晚构建

每晚构建Amazon SAMCLI 可供你安装。安装完成后，您可以使用`sam-nightly`命令。您可以安装和使用的生产版本和夜间构建版本Amazon SAM同时提交 CLI。

每晚版本包含一个预发布版本Amazon SAMCLI 代码可能不如生产版本稳定。请注意，夜间构建不包含构建映像的预发行版本，因此使用`--use-container`选项使用构建映像的最新生产版本。

这个下载链接可以使用每晚版本：Amazon SAMCLI 每晚构建. 安装的夜间构建版本Amazon SAMCLI、执行与中相同的步骤第 4 步：安装Amazon SAMCLI (p. 5)本主题前面的部分，但请改用每晚构建下载链接。你可以在Amazon SAM针对夜间构建的 CLI 发布说明（位于 GitHub 上）。

要验证你是否安装了夜间构建版本，请运行`sam-nightly --version`命令。此命令的输出是以格式进行的。`1.X.Y.dev<YYYYMMDDHHmm>`，例如：

```
SAM CLI, version 1.20.0.dev202103151200
```

故障排除

Docker 错误：“无法连接到 Docker 守护程序。docker 守护进程在此主机上运行吗？”

在某些情况下，为提供权限`ec2-user`要访问 Docker 守护程序，您可能需要重启实例。如果收到此错误，请尝试重启实例。

Shell 错误：“找不到命令”

如果你收到此错误，你的 shell 无法找到Amazon SAM路径中的 CLI 可执行文件。验证已安装的目录的位置Amazon SAMCLI 可执行文件，然后验证目录是否在路径上。

Amazon SAMCLI 错误：“/lib64/libc.so.6：找不到版本 `GLIBC_2.14' (由 /usr/local /aws-sam-cli/dist/libz.so.1 要求) ”

如果收到此错误，表示您使用的是不受支持的 Linux 版本，并且内置 glibc 版本已过时。请尝试以下任一操作：

- 将 Linux 主机升级到最近发行的 CentOS、Fedora、Ubuntu 或 Amazon Linux 2 的 64 位版本。
- 按照说明进行操作安装Amazon SAM使用 Linux 上的 CLIHomebrew (p. 7).

后续步骤

现在，您可以使用开始构建自己的无服务器应用程序。Amazon SAM. 要从示例无服务器应用程序开始，请选择以下链接之一：

- 教程：部署 Hello World 应用程序 (p. 15)— 下载、构建和部署简单的无服务器应用程序的分步说明。
- 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，你可以进一步试验。

安装Amazon SAM使用 Linux 上的 CLIHomebrew

安装Amazon SAM在 Linux 上使用 CLI，您可以使用Homebrew程序包管理器。有关 的更多信息Homebrew请参阅Homebrew在 Linux 上在Homebrew文档网站。

> **Note**
>
> 安装Homebrew将环境的默认 Python 版本更改为Homebrew安装。

安装Homebrew。Git 适用于许多不同的操作系统，包括大多数现代 Linux 分发版。有关在特定操作系统上安装 Git 的说明，请参阅安装 Git在 Git 网站上。

安装 Homebrew

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)"
```

下一步，添加Homebrew通过运行以下命令，将其转换为 PATH。这些命令适用于 Linux 的所有主要类型，方法是添加`~/.profile`在 Debian 和 Ubuntu 上，或`~/.bash_profile`在 CentOS、Fedora 和红帽上。

```
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
```

确认 。Homebrew已安装。

```
brew --version
```

成功安装Homebrew将会看到类似下面的输出：

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

安装Amazon SAM使用 CLIHomebrew

安装Amazon SAM使用 CLIHomebrew请运行以下命令：

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

验证安装。

```
sam --version
```

成功安装Amazon SAM将会看到类似下面的输出：

```
SAM CLI, version 1.35.0
```

升级Amazon SAM使用 CLIHomebrew

升级Amazon SAM使用 CLIHomebrew，替换install和upgrade如下所示：

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

每晚使用构建Homebrew

每晚构建Amazon SAMCLI 可供你安装。安装后，您可以使用`sam-nightly`命令。您可以安装和使用的生产版本和夜间构建版本Amazon SAM同时提交 CLI。

每晚版本包含的预发布版本Amazon SAMCLI 代码可能不如生产版本稳定。请注意，夜间构建不包含构建映像的预发行版本，因此使用`--use-container`选项使用构建映像的最新生产版本。

安装的夜间构建版本Amazon SAM请运行以下命令：

```
brew tap aws/tap
```

```
brew install aws-sam-cli-nightly
```

要验证你是否安装了夜间构建版本，请运行`sam-nightly --version`命令。此命令的输出为格式`1.X.Y.dev<YYYYMMDDHHmm>`，例如：

```
SAM CLI, version 1.20.0.dev202103151200
```

故障排除

安装Homebrew消息：“输入密码以安装到 /家/linuxbrew/.linuxbrew”

在安装 Homebrew在默认情况下，系统会提示您提供密码。但是，您可能不想为当前用户设置密码，例如，当您设置 CI/CD 系统等非交互式环境时。

如果不想为当前用户设置密码，可以安装Homebrew在非交互模式下，通过设置环境变量`CI=1`. 例如：

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

安装Amazon SAMCLI 错误：“以下公式不能从瓶子中安装，必须从源代码构建。pkg-config、gdbm、openssl @1 .1、ncurses、xz 和 python @3 .8"

如果在安装Amazon SAMCLI，您没有`gcc`已安装模块。安装`gcc`适用于 Linux 分发版的模块。

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

在安装`gcc`模块中运行命令安装Amazon SAM使用 Homebrew CLI再次部分。

Shell 错误：“找不到命令”

如果你收到此错误，你的 shell 无法找到Amazon SAMPATH 中的 CLI 可执行文件。验证已安装Amazon SAMCLI 可执行文件，然后验证目录是否在 PATH 上。

如果您使用本主题中的说明同时安装Homebrew使用Homebrew要安装Amazon SAM然后Amazon SAM将 CLI 可执行文件安装到以下位置：

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

安装Amazon SAM在 Windows 上执行 CLI

请按照以下步骤安装和配置使用Amazon SAMWindows 主机上的命令行界面 (CLI)：

1. 创建Amazon Identity and Access Management(Amazon) 账户。
2. 配置 IAM 权限和Amazon凭证。
3. 安装 Docker. 注意：Docker 只是在本地测试应用程序或使用`--use-container`选项。
4. 安装 Amazon SAM CLI。

第 1 步：创建 Amazon 账户

如果您还没有Amazon账户，请参阅aws.amazon.com然后选择创建Amazon账户. 有关详细说明，请参阅创建并激活Amazon账户.

第 2 步：配置 IAM 权限和Amazon证书

与之一使用的 IAM 用户Amazon SAM必须有足够的权限才能进行必要Amazon服务呼叫和管理Amazon资源的费用。确保用户拥有足够权限的最简单方法是向他们授予管理员权限。有关更多信息，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。

> Note
>
> 如果您不想将管理员权限授予使用Amazon Command Line Interface(Amazon CLI)，您可以向他们授予受限的权限集。有关更多信息，请参阅 权限 (p. 330)。

此外，要启用Amazon SAM要制作的 CLIAmazon必须设置服务呼叫，您必须设置Amazon凭证。有关更多信息，请参阅 设置Amazon证书 (p. 14)。

第 3 步：安装 Docker（可选）

> Note
>
> Docker 只是在本地测试应用程序以及使用`--use-container`选项。如果您最初不打算使用这些功能，则可以跳过本节或稍后安装 Docker。

> Note
>
> 我们正式支持 Docker Desktop，但是从 SAM CLI 版本 1.47.0 开始，只要使用 Docker 运行时，您就可以使用替代方案。

Docker 是一款在 Linux 计算机上运行容器的应用程序。Amazon SAM提供的本地环境类似于Amazon Lambda用作 Docker 容器。您可以使用此容器构建、测试和调试无服务器应用程序。

使用本地运行无服务器项目和功能Amazon SAMCLI，你必须安装 Docker 并正常工作。这些区域有：Amazon SAMCLI 使用`DOCKER_HOST`用于联系 Docker 守护程序的环境变量。以下步骤介绍如何安装、配置和验证 Docker 安装以使用Amazon SAMCLI。

1. 安装 Docker.

   Docker 桌面支持最新的 Windows 操作系统。对于 Windows 的旧版本，Docker 工具箱可用。选择你的 Windows 版本以获取正确的 Docker 安装步骤：

   - 要安装适用于 Windows 10 的 Docker，请参阅安装适用于 Windows 的 Docker 桌面.
   - 要为旧版本的 Windows 安装 Docker，请参阅在 Windows 上安装 Docker 工具箱.

2. 配置共享驱动器。

   这些区域有：Amazon SAMCLI 要求项目目录或任何父目录都列在共享驱动器中。在某些情况下，您必须共享驱动器才能正常运行 Docker。

   - 如果你在 Hyper-V 模式下使用 Windows 10，请参阅Docker 文件共享.
   - 要在旧版本的 Windows 上共享驱动器，请参阅添加共享目录.

3. 验证安装。

   安装 Docker 之后，验证它是否正常工作。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、提取或拉取任何容器 —Amazon SAMCLI 根据需要自动执行此操作。

如果您在安装 Docker 时遇到问题，请参阅日志和故障排除的 部分Docker 安装指南有关其他故障排除提示。

第 4 步：安装Amazon SAMCLI

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

请按照以下步骤安装Amazon SAMCLI 使用 MSI 文件。

1. 安装Amazon SAMCLI64 位.

   Note

   如果你在 32 位系统上运行，请参阅安装Amazon SAM32 位 Windows 上的 CLI (p. 334).
2. 验证安装。

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

   ```
   sam --version
   ```

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

   ```
   SAM CLI, version 1.35.0
   ```
3. 安装 Git。

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

现在您已准备就绪，可以启动开发。

卸载

卸载Amazon SAM使用 Windows 设置 CLI，请执行以下步骤：

1. 在 “开始” 菜单中，搜索 “添加或删除程序”。
2. 选择名为的条目Amazon SAMCommand Line Interface然后选择卸载来启动卸载程序。
3. 确认您要卸载 Amazon SAM CLI。

每晚构建

每晚构建Amazon SAMCLI 可供你安装。安装完成后，您可以使用每晚使用`sam-nightly`命令。您可以安装和使用Amazon SAM同时提交 CLI。

每晚版本包含一个预发布版Amazon SAMCLI 代码可能不如生产版本稳定。请注意，夜间构建不包含构建映像的预发行版本，因此使用`--use-container`选项使用构建映像的最新生产版本。

这个下载链接可以使用每晚版本：Amazon SAMCLI 每晚构建. 要安装的夜间构建版Amazon SAMCLI、执行与第 4 步：安装Amazon SAMCLI (p. 11)本主题前面的部分，但请改用每晚构建下载链接。

要验证你是否安装了夜间构建版本，请运行`sam-nightly --version`命令。此命令的输出采用格式`1.X.Y.dev<YYYYMMDDHHmm>`例如：

```
SAM CLI, version 1.20.0.dev202103151200
```

后续步骤

现在，您可以使用以下内容来构建自己的无服务器应用程序Amazon SAM！如果您想从示例无服务器应用程序开始，请选择下列链接之一：

- 教程：部署 Hello World 应用程序 (p. 15)— 下载、构建和部署简单的无服务器应用程序的分步说明。
- 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，你可以进一步试验。

安装Amazon SAMmacOS 上的 CLI

执行以下步骤可安装和配置使用Amazon SAMmacOS 主机上的命令行界面 (CLI)：

1. 创建一个 Amazon 账户。
2. 配置Amazon Identity and Access Management(IAM) 权限和Amazon凭证。
3. 安装 Docker. 注意：Docker 只是在本地测试应用程序或使用`--use-container`选项
4. 安装 Homebrew。
5. 安装 Amazon SAM CLI。

第 1 步：创建 Amazon 账户

如果您还没有Amazon账户，请参阅aws.amazon.com然后选择创建Amazon账户. 有关详细说明，请参阅如何创建和激活新的Amazon账户？

第 2 步：配置 IAM 权限和Amazon证书

与之一使用的 IAM 用户Amazon SAM必须有足够的权限才能进行必要Amazon服务呼叫和管理Amazon资源的费用。确保用户拥有足够权限的最简单方法是向他们授予管理员权限。有关更多信息，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。

> Note
>
> 如果您不想向使用Amazon Command Line Interface(Amazon CLI)，您可以向他们授予受限的权限集。有关更多信息，请参阅 权限 (p. 330)。

此外，要启用Amazon SAM要制作的 CLIAmazon您必须设置服务呼叫Amazon凭证。有关更多信息，请参阅设置Amazon证书 (p. 14)。

第 3 步：安装 Docker（可选）

> Note
>
> Docker 只是在本地测试应用程序以及使用`--use-container`选项。如果您最初不打算使用这些功能，则可以跳过本部分或在以后安装 Docker。

> Note
>
> 我们正式支持 Docker Desktop，但是从 SAM CLI 版本 1.47.0 开始，只要使用 Docker 运行时，您就可以使用替代方案。

Docker 是一款在 macOS 计算机上运行容器的应用程序。Amazon SAM提供的本地环境类似于Amazon Lambda用作 Docker 容器。您可以使用此容器构建、测试和调试无服务器应用程序。

使用本地运行无服务器项目和功能Amazon SAMCLI，你必须安装 Docker 并正常工作。这些区域有：Amazon SAMCLI 使用DOCKER_HOST与 Docker 守护程序联系的环境变量。以下步骤介绍如何安装、配置和验证 Docker 安装以与Amazon SAMCLI。

1. 安装 Docker

   这些区域有：Amazon SAMCLI 支持在 macOS Sierra 10.12 或更高版本上运行的 Docker。要安装 Docker，请参阅安装适用于 Mac 的 Docker 桌面.
2. 配置共享驱动器

   这些区域有：Amazon SAMCLI 要求项目目录或任何父目录都列在共享驱动器中。要在 macOS 上共享驱动器，请参阅文件共享.
3. 验证安装

   安装 Docker 之后，验证它是否正常工作。还要确认您可以通过命令行运行 Docker 命令（例如，`docker ps`）。您无需安装、提取或拉取任何容器 ——Amazon SAMCLI 根据需要自动执行此操作。

如果您在安装 Docker 时遇到问题，请参阅日志和故障排除的 部分Docker 安装指南有关其他故障排除提示。

第 4 步：安装 Homebrew

推荐的安装方法Amazon SAMmacOS 上的 CLI 是使用Homebrew程序包管理器。有关 Homebrew 的更多信息，请参阅 Homebrew 文档。

安装Homebrew，您必须首先安装 Git。有关 Git 的更多信息，请参阅Git 文档. Git 适用于许多不同的操作系统，包括 macOS。有关在特定操作系统上安装 Git 的说明，请参阅安装 Git.

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)"
```

确认 。Homebrew已安装：

```
brew --version
```

在成功安装时，您应看到类似下面的输出，类似于Homebrew：

```
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)
```

第 5 步：安装Amazon SAMCLI

按照以下步骤安装Amazon SAM使用 CLIHomebrew：

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

验证安装：

```
sam --version
```

在成功安装后，您应看到类似下面的输出，类似于以下Amazon SAMCLI：

```
SAM CLI, version 1.35.0
```

现在，您已准备好启动开发。

升级

升级Amazon SAM使用 CLIHomebrew中，运行以下命令：

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

卸载

卸载Amazon SAM使用 CLIHomebrew中，运行以下命令：

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

每晚构建

每晚构建Amazon SAMCLI 可供你安装。安装完成后，您可以使用`sam-nightly`命令。您可以安装和使用Amazon SAM同时提交 CLI。

每晚版本包含一个预发布版本Amazon SAMCLI 代码可能不如生产版本稳定。请注意，夜间构建不包含构建映像的预发行版本，因此使用`--use-container`选项使用构建映像的最新生产版本。

要安装的每晚构建版本Amazon SAMCLI、运行以下命令：

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

要验证你是否安装了夜间构建版本，请运行`sam-nightly --version`命令。此命令的输出为格式。`1.X.Y.dev<YYYYMMDDHHmm>`例如：

```
SAM CLI, version 1.20.0.dev202103151200
```

后续步骤

现在，您可以使用此方式开始构建自己的无服务器应用程序Amazon SAM！如果您要从示例无服务器应用程序开始，请选择以下链接之一：

- 教程：部署 Hello World 应用程序 (p. 15)— 下载、构建和部署简单的无服务器应用程序的分步说明。
- 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，你可以进一步试验。

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：

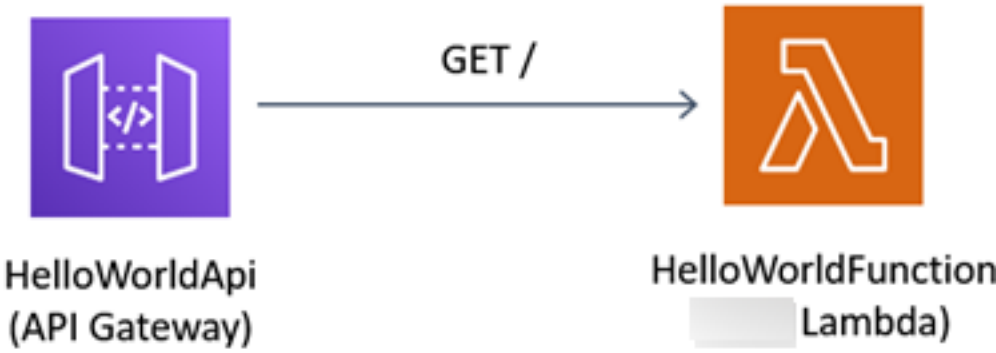
```
set AWS_ACCESS_KEY_ID=your_access_key_id
set AWS_SECRET_ACCESS_KEY=your_secret_access_key
```

教程：部署 Hello World 应用程序

在本指南中，您可以使用下载、构建和部署示例 Hello World 应用程序Amazon SAM. 然后，您可以在Amazon云，可以选择在开发主机上本地测试它。

此应用程序实现了基本的 API 后端。它由 Amazon API Gateway 终端节点和Amazon Lambdafunction. 向 API Gateway 终端节点发送 GET 请求时，会调用 Lambda 函数。此函数返回`hello world`消息。

下图显示此应用程序的组件：



初始化示例应用程序时，您可以选择 Lambda 部署包类型，`Zip`要么`Image`. 有关包装类型的更多信息，请参阅Lambda 部署程序包中的Amazon Lambda开发人员指南.

以下是为创建 Hello World 应用程序而运行的命令的预览。有关其中每个命令的更多信息，请参阅本教程后面的各个部分。

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

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

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

先决条件

本指南假定您已完成中的操作系统的步骤。安装 Amazon SAM CLI (p. 3)，其中包括：

1. 创建Amazonaccount.
2. 配置Amazon Identity and Access Management(IAM) 权限。
3. 安装 Docker。注意：Docker 只是在本地测试应用程序的先决条件。
4. 安装 Homebrew。注意：自制软件只是 Linux 和 macOS 的先决条件。
5. 安装Amazon SAM命令行界面 (CLI)。注意：请确保您具有版本 1.13.0 或更高版本。通过运行`sam --version`命令。

第 1 步：下载示例Amazon SAM应用程序

要运行的命令：

```
sam init
```

按照屏幕上的提示操作。对于本教程，建议您选择Amazon Quick Start Templates，Zip软件包类型、您选择的运行时间以及Hello World Example.

输出示例：

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

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

什么是Amazon SAM在执行此操作：

此命令以您提供的名称作为项目名称创建一个目录。项目目录的内容类似于：

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

> **Note**
>
> 选择其中一个 Python 运行时和Hello World Example.

有三个特别重要的文件：

- template.yaml：包含Amazon SAM定义应用程序的模板Amazon资源的费用。
- hello_world/app.py：包含实际的 Lambda 处理程序逻辑。
- hello_world/requirements.txt：包含应用程序所需的任何 Python 依赖项，用于sam build.

第 2 步：构建你的应用

要运行的命令：

首先，切换到项目目录，其中`template.yaml`已找到示例应用程序的文件。（默认情况下，此目录为`sam-app`。）然后运行以下命令：

```
sam build
```

输出示例：

```
Build Succeeded

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

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

什么是Amazon SAM在执行此操作：

这些区域有：Amazon SAMCLI 提供多个 Lambda 运行时的抽象，以构建您的依赖项，并将源代码复制到暂存文件夹中，以便所有内容都准备好打包和部署。这些区域有：`sam build`命令将构建应用程序具有的任何依赖项，并将应用程序源代码复制到下的文件夹`.aws-sam/build`要压缩并上传到 Lambda。

您可以在下面看到以下顶级树`.aws-sam`：

```
.aws-sam/
  ### build/
      ### HelloWorldFunction/
      ### template.yaml
```

`HelloWorldFunction`是一个包含您的目录`app.py`文件，以及应用程序使用的第三方依赖项。

第 3 步：将应用程序部署到Amazon云

要运行的命令：

```
sam deploy --guided
```

按照屏幕上的提示操作。要接受交互式体验中提供的默认选项，请使用`Enter`.

> **Note**
>
> 对于提示`HelloWorldFunction may not have authorization defined, Is this okay? [y/N]`、Amazon SAM通知您示例应用程序在未经授权的情况下配置了 API Gateway API。部署示例应用程序时，Amazon SAM创建公开可用的 URL。
> 您可以通过对提示回答 “Y” 来确认此通知。有关配置授权的信息，请参阅控制对 API Gateway API 的访问 (p. 192).

输出示例：

```
Deploying with following values
===============================
Stack name                 : sam-app
```

```
    Region                    : us-east-1
    Confirm changeset         : False
    Deployment s3 bucket      : sam-bucket
    Capabilities              : ["CAPABILITY_IAM"]
    Parameter overrides       : {}

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

 Waiting for changeset to be created..

 CloudFormation stack changeset
 ---------------------------------------------------------------------------------------------------------
 Operation                                          LogicalResourceId
          ResourceType
 ---------------------------------------------------------------------------------------------------------
 + Add
 HelloWorldFunctionHelloWorldPermissionProd         AWS::Lambda::Permission
 + Add                                              ServerlessRestApiDeployment47fc2d5f9d
          AWS::ApiGateway::Deployment
 + Add                                              ServerlessRestApiProdStage
          AWS::ApiGateway::Stage
 + Add                                              ServerlessRestApi
          AWS::ApiGateway::RestApi
 * Modify                                           HelloWorldFunctionRole
          AWS::IAM::Role
 * Modify                                           HelloWorldFunction
          AWS::Lambda::Function
 ---------------------------------------------------------------------------------------------------------

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

 CloudFormation events from changeset
 ---------------------------------------------------------------------------------------------------------
 ResourceStatus                       ResourceType
 LogicalResourceId                    ResourceStatusReason
 ---------------------------------------------------------------------------------------------------------
 UPDATE_IN_PROGRESS                   AWS::IAM::Role
 HelloWorldFunctionRole               -
 UPDATE_COMPLETE                      AWS::IAM::Role
 HelloWorldFunctionRole               -
 UPDATE_IN_PROGRESS                   AWS::Lambda::Function
 HelloWorldFunction                   -
 UPDATE_COMPLETE                      AWS::Lambda::Function
 HelloWorldFunction                   -
 CREATE_IN_PROGRESS                   AWS::ApiGateway::RestApi
 ServerlessRestApi                    -
 CREATE_COMPLETE                      AWS::ApiGateway::RestApi
 ServerlessRestApi                    -
 CREATE_IN_PROGRESS                   AWS::ApiGateway::RestApi
 ServerlessRestApi                    Resource creation Initiated
 CREATE_IN_PROGRESS                   AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment47fc2d5   Resource creation Initiated
                                                                                  f9d
 CREATE_IN_PROGRESS                   AWS::Lambda::Permission
 HelloWorldFunctionHelloWorldPermis   Resource creation Initiated
                                                                                  sionProd
 CREATE_IN_PROGRESS                   AWS::Lambda::Permission
 HelloWorldFunctionHelloWorldPermis   -
                                                                                  sionProd
 CREATE_IN_PROGRESS                   AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment47fc2d5   -
                                                                                  f9d
CREATE_COMPLETE                      AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment47fc2d5   -
                                                                                  f9d
```

```
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
HelloWorldFunctionHelloWorldPermis  -
                                                                    sionProd
UPDATE_COMPLETE_CLEANUP_IN_PROGRES  AWS::CloudFormation::Stack      sam-app
                    -
S
UPDATE_COMPLETE                     AWS::CloudFormation::Stack      sam-app
                    -
-------------------------------------------------------------------------------------------------

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

HelloWorldFunction-1TY92MJX0BXU5
-------------------------------------------------------------------------------------------------

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

什么是Amazon SAM在执行此操作：

此命令将应用程序部署到Amazon云。它需要使用你构建的部署工件`sam build`命令，将其打包并上传到 Amazon Simple Storage Storage Storage Storage Storage Service (Amazon S3) 存储桶中Amazon SAMCLI 使用创建和部署应用程序Amazon CloudFormation. 在输出中`sam deploy`命令中，您可以看到正在对您的 Amazon CloudFormation堆栈。

如果您的应用程序创建了 HTTP 终端节点，则输出`sam deploy`生成还会向您显示测试应用程序的终端节点 URL。您可以使用`curl`使用该终端节点 URL 向应用程序发送请求。例如：

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

成功部署应用程序后，您会看到类似以下内容的输出：

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

如果你明白`{"message": "hello world"}`执行之后`curl`命令，您已成功将无服务器应用程序部署到 Amazon，而且你正在调用你的实时 Lambda 函数。否则，请参阅故障排除 (p. 23)本教程后面的部分。

第 4 步：（可选）在本地测试应用程序

当你开发应用程序时，你可能会发现在本地进行测试很有用。这些区域有：Amazon SAMCLI 提供`sam local`命令来使用模拟 Lambda 执行环境的 Docker 容器运行应用程序。有两种方式可执行此操作：

- 在本地托管你的 API
- 直接调用 Lambda 函数

此步骤介绍了两个选项。

在本地托管你的 API

要运行的命令：

```
sam local start-api
```

输出示例：

```
 2019-07-12 15:27:58 Mounting HelloWorldFunction at http://127.0.0.1:3000/hello [GET]
 2019-07-12 15:27:58 You can now browse to the above endpoints to invoke your functions.
 You do not need to restart/reload SAM CLI while working on your functions, changes will
 be reflected instantly/automatically. You only need to restart SAM CLI if you update your
 Amazon SAM template
 2019-07-12 15:27:58  * Running on http://127.0.0.1:3000/ (Press CTRL+C to quit)

 Fetching lambci/lambda:python3.7 Docker container
 image.....................................................................................
 2019-07-12 15:28:56 Mounting /<working-development-path>/sam-app/.aws-sam/build/
HelloWorldFunction as /var/task:ro,delegated inside runtime container
 START RequestId: 52fdfc07-2182-154f-163f-5f0f9a621d72 Version: $LATEST
 END RequestId: 52fdfc07-2182-154f-163f-5f0f9a621d72
 REPORT RequestId: 52fdfc07-2182-154f-163f-5f0f9a621d72  Duration: 4.42 ms       Billed
 Duration: 100 ms Memory Size: 128 MB     Max Memory Used: 22 MB
 2019-07-12 15:28:58 No Content-Type given. Defaulting to 'application/json'.
 2019-07-12 15:28:58 127.0.0.1 - - [12/Jul/2019 15:28:58] "GET /hello HTTP/1.1" 200 -
```

加载 Docker 映像可能需要一段时间。加载后，你可以使用`curl`向在本地主机上运行的应用程序发送请求：

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

输出示例：

```
 2019-07-12 15:29:57 Invoking app.lambda_handler (python3.7)
 2019-07-12 15:29:57 Found credentials in shared credentials file: ~/.aws/credentials

 Fetching lambci/lambda:python3.7 Docker container image......
 2019-07-12 15:29:58 Mounting /<working-development-path>/sam-app/.aws-sam/build/
HelloWorldFunction as /var/task:ro,delegated inside runtime container
 START RequestId: 52fdfc07-2182-154f-163f-5f0f9a621d72 Version: $LATEST
 END RequestId: 52fdfc07-2182-154f-163f-5f0f9a621d72
 REPORT RequestId: 52fdfc07-2182-154f-163f-5f0f9a621d72  Duration: 7.92 ms       Billed
 Duration: 100 ms Memory Size: 128 MB     Max Memory Used: 22 MB
 {"statusCode":200,"body":"{\"message\": \"hello world\"}"}
```

什么是Amazon SAM在执行此操作：

这些区域有：`start-api`命令启动复制 REST API 终端节点的本地终端节点。它会下载一个执行容器，您可以在本地运行函数。最终结果与你在Amazon云。

直接调用 Lambda 函数

要运行的命令：

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

输出示例：

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

 Fetching lambci/lambda:python3.7 Docker container
 image.....................................................................................................
 2019-07-01 14:09:39 Mounting /<working-development-path>/sam-app/.aws-sam/build/
HelloWorldFunction as /var/task:ro,delegated inside runtime container
 START RequestId: 52fdfc07-2182-154f-163f-5f0f9a621d72 Version: $LATEST
 END RequestId: 52fdfc07-2182-154f-163f-5f0f9a621d72
 REPORT RequestId: 52fdfc07-2182-154f-163f-5f0f9a621d72    Duration: 3.51 ms    Billed
 Duration: 100 ms    Memory Size: 128 MB    Max Memory Used: 22 MB
 {"statusCode":200,"body":"{\"message\": \"hello world\"}"}
```

什么是Amazon SAM在执行此操作：

这些区域有：`invoke`命令直接调用您的 Lambda 函数，并且可以传递您提供的输入事件有效负载。使用此命令，您可以在文件中传递事件负载`event.json`该示例应用程序提供的。

您的初始化应用程序带有默认值`aws-proxy`API Gateway 的事件。已为您预先填充多个值。在这种情况下，`HelloWorldFunction`不关心特定的值，所以一个固定的请求是可以的。您可以指定一些要替代请求的值，以模拟实际请求所期望的内容。以下是生成自己的输入事件并将输出与默认值进行比较的示例`event.json`对象：

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

输出示例：

```
 <   "body": "",
 ---
 >   "body": "{\"message\": \"hello world\"}",
 4,6c4,6
 <   "path": "/hello",
 <   "httpMethod": "GET",
 <   "isBase64Encoded": true,
 ---
 >   "path": "/path/to/resource",
 >   "httpMethod": "POST",
 >   "isBase64Encoded": false,
 11c11
 <     "proxy": "/hello"
 ---
 >     "proxy": "/path/to/resource"
 56c56
 <     "path": "/prod/hello",
 ---
 >     "path": "/prod/path/to/resource",
 58c58
```

```
<     "httpMethod": "GET",
---
>     "httpMethod": "POST",
```

故障排除

Amazon SAMCLI 错误：“安全限制未满足”

运行时sam deploy --guided，系统会提示你问题`HelloWorldFunction may not have authorization defined, Is this okay? [y/N]`. 如果你回应此提示**N**（默认响应），您会看到以下错误：

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

该提示通知您，您即将部署的应用程序可能配置了未经授权的 Amazon API Gateway API。通过响应**N**对于这个提示，你说这不好。

要解决此问题，您可使用以下选项：

- 通过授权配置应用程序。有关配置授权的信息，请参阅控制对 API Gateway API 的访问 (p. 192).
- 用回答这个问题**Y**表明您可以部署未经授权配置了 API Gateway API 的应用程序。

Amazon SAMCLI 错误：“没有此类选项：—app 模板”

执行 `sam init` 时，您会看到以下错误：

```
Error: no such option: --app-template
```

这意味着您使用的是不支持 `--app-template` 参数的旧版本 Amazon SAM CLI。要解决此问题，您可以将 Amazon SAM CLI 版本更新为 0.33.0 或更高版本，也可以省略 `sam init` 命令中的 `--app-template` 参数。

Amazon SAMCLI 错误：“没有此类选项：—guide”

执行 `sam deploy` 时，您会看到以下错误：

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

这意味着您使用的是不支持 `--guided` 参数的旧版本 Amazon SAM CLI。要解决此问题，您可以将 Amazon SAM CLI 版本更新为 0.33.0 或更高版本，也可以省略 `sam deploy` 命令中的 `--guided` 参数。

Amazon SAMCLI 错误：“无法创建托管资源：无法找到凭证”

执行 `sam deploy` 时，您会看到以下错误：

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

这意味着您还没有设置Amazon用于启用Amazon SAM要制作的 CLIAmazon服务电话。要解决此问题，您必须设置Amazon凭证。有关更多信息，请参阅 设置Amazon证书 (p. 14)。

Amazon SAMCLI 错误：“正在运行Amazon本地 SAM 项目需要 Docker。你安装了吗？”

执行 `sam local start-api` 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，您没有正确安装 Docker。Docker 需要在本地测试应用程序。要修复此问题，请按照为开发主机安装 Docker 的说明进行操作。转到安装 Amazon SAM CLI (p. 3)，选择适当的平台，然后按照标题为的部分中的说明进行操作安装 Docker.

卷曲错误：“缺少身份验证令牌”

尝试调用 API Gateway 终端节点时，会显示以下错误：

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

这意味着您尝试向正确的域发送请求，但 URI 无法识别。要修复此问题，请验证完整的 URL，然后更新`curl`具有正确 URL 的命令。

卷曲错误：“curl：(6) 无法解决：...”

尝试调用 API Gateway 终端节点时，会显示以下错误：

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

这意味着您试图向无效域发送请求。如果您的无服务器应用程序未能成功部署，或者如果您在`curl`命令。通过使用Amazon CloudFormation控制台或Amazon CLI，然后验证你的`curl`命令是正确的。

清除

如果您不再需要Amazon通过运行本教程创建的资源，则可以通过删除Amazon CloudFormation你部署的堆栈。

删除Amazon CloudFormation使用堆栈Amazon Web Services Management Console按照以下步骤操作：

1. 登录到 Amazon Web Services Management Console 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在左侧导航窗格中，选择 Stacks (堆栈)。
3. 在堆栈列表中，选择sam-app（或者您创建的堆栈的名称）。
4. 选择 Delete。

完成后，堆栈的状态更改为DELETE_完成.

或者，您也可以删除Amazon CloudFormation通过运行以下命令来堆叠Amazon CLI命令：

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

验证已删除的堆栈

对于删除的两种方法Amazon CloudFormation堆栈，你可以通过转到Amazon CloudFormation控制台. 在左侧导航窗格中，选择堆栈，然后在搜索框旁边的下拉列表中，选择Deleted (已删除). 您应该在已删除堆栈的列表中看到您的堆栈的名称。

结论

在本教程中，您完成了以下操作：

1. 创建、构建和部署了无服务器应用程序Amazon使用Amazon SAM.
2. 使用在本地测试应用程序Amazon SAMCLI 和 Docker。
3. 已删除Amazon您不再需要的资源。

后续步骤

现在，您可以使用Amazon SAMCLI。

为了帮助您入门，您可以从Amazon Serverless Application Repository示例GitHub 上的仓库。

Amazon Serverless Application Model(Amazon SAM) 规范

您将Amazon SAM定义您的无服务器应用程序的规范。此部分提供有关Amazon SAM模板部分、资源类型、资源属性、数据类型、资源属性、资源属性、内部函数和 API Gateway 扩展，您可以在Amazon SAMTemplates。

Amazon SAM模板是的扩展Amazon CloudFormation模板，包括一些额外的组件，使它们更易于使用。有关的完整参考Amazon CloudFormation模板，请参阅Amazon CloudFormation模板参考中的Amazon CloudFormation用户指南.

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，请参阅的 “全局变量” 部分Amazon SAM模板 (p. 28).
- 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. 31)。
- 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. 257)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. 28)

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. 257)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. 281)。

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，请参阅开始使用 Amazon SAM (p. 3)然后按照中的说明操作教程：部署 Hello World 应用程序 (p. 15).

的 “全局变量” 部分Amazon SAM模板

有时候你在Amazon SAM模板具有通用的配置。例如，您可能拥有具有多个应用程序AWS::Serverless::Function具有相同的资源Runtime、Memory、VPCConfig、Environment，和Cors配置。您可以在每个资源中声明一次，而不是在每个资源中复制这些信息Globals部分，然后让你的资源继承它们。

这些区域有：Globals部分受AWS::Serverless::Function、AWS::Serverless::Api、AWS::Serverless::HttpApi，和AWS::Serverless::SimpleTable资源的费用。

例如：

```
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，两种方法`HelloWorldFunction`和`ThumbnailFunction`使用 “nodejs12.x”`Runtime`，“180” 秒`Timeout`和 “index.handler”`Handler`.`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前面列表中的资源和属性。不支持他们的一些原因包括：1) 他们打开了潜在的安全问题，或者 2) 它们使模板难以理解。

隐式 API

Amazon SAM创建隐式 API当你在`Events`部分。您可以使用`Globals`以覆盖隐式 API 的所有属性。

可覆盖的属性

资源可以覆盖您在`Globals`部分。例如，您可以向环境变量映射添加新变量，也可以覆盖全局声明的变量。但是，资源无法删除在`Globals`部分。

更一般而言，`Globals`部分声明了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，但无法删除它们。如果有些资源使用属性但其他资源不使用属性，那么您不能在`Globals`部分。

下面几节介绍如何针对不同的数据类型进行覆盖。

原始数据类型被替换

原始数据类型包括字符串、数字、布尔值等。

中指定的值`Resources`部分将替换`Globals`部分。

例如：

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

Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      Runtime: python3.6
```

这些区域有：`Runtime`为了`MyFunction`设置为`python3.6`.

地图已合并

地图也称为字典或键值对集合。

映射中的条目`Resources`部分与全局地图条目合并。如果有重复项，`Resource`部分条目覆盖`Globals`部分条目。

例如：

```
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MyFunction设置为以下内容：

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

列表是附加的

列表也称为数组。

列出Globals部分在列表中的前面Resources部分。

例如：

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

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

这些区域有：SecurityGroupIds为了MyFunction的VpcConfig设置为以下内容：

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

Amazon SAM资源和属性参考

本节包含的参考信息Amazon SAM资源和属性类型。

主题

- AWS::Serverless::Application (p. 64)
- AWS::Serverless::Function (p. 67)
- AWS::Serverless::HttpApi (p. 132)
- AWS::Serverless::LayerVersion (p. 152)
- AWS::Serverless::SimpleTable (p. 155)
- AWS::Serverless::StateMachine (p. 158)

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。

网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的AWS::Serverless::Api (p. 32)资源不必显式添加到Amazon无服务器应用程序定义模板。这种类型的资源是通过上定义的 Api 事件联合隐式创建的AWS::Serverless::Function (p. 67)模板中定义的资源不引用AWS::Serverless::Api (p. 32)资源。

网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的AWS::Serverless::Api (p. 32)应该使用资源来定义和记录 API OpenApi，它提供了配置底层Amazon API Gateway 资源的更多功能。

建议使用Amazon CloudFormation挂钩或 IAM 策略来验证 API Gateway 资源是否附加了授权方来控制对它们的访问。

有关的更多信息Amazon CloudFormation挂钩，请参阅注册挂钩中的Amazon CloudFormationCLI 用户指南和apigw-enforce-authorizer GitHub 存储库。

有关如何使用的IAM;; 策略的更多信息，请参阅。要求 API 路由具有授权中的API Gateway 开发人员指南.

语法

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

YAML

```
Type: AWS::Serverless::Api
Properties:
  AccessLogSetting: AccessLogSetting
  ApiKeySourceType: String
  Auth: ApiAuth (p. 41)
  BinaryMediaTypes: List
  CacheClusterEnabled: Boolean
  CacheClusterSize: String
  CanarySetting: CanarySetting
  Cors: String | CorsConfiguration (p. 57)
  DefinitionBody: JSON
  DefinitionUri: String | ApiDefinition (p. 56)
  Description: String
  DisableExecuteApiEndpoint: Boolean
  Domain: DomainConfiguration (p. 59)
  EndpointConfiguration: EndpointConfiguration (p. 63)
  FailOnWarnings: Boolean
  GatewayResponses: Map
  MethodSettings: MethodSettings
  MinimumCompressionSize: Integer
  Mode: String
  Models: Map
  Name: String
  OpenApiVersion: String
  StageName: String
  Tags: Map
```

```
  TracingEnabled: Boolean
  Variables: Map
```

属性

`AccessLogSetting`

配置阶段的访问日志设置。

类型：AccessLogSetting

必需：否

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

`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：此属性将直接传递给 Day`ApiKeySourceType`一个的财产`AWS::ApiGateway::RestApi`资源。

`Auth`

配置授权以控制对 API Gateway API 的访问。

有关配置访问权限的更多信息Amazon SAM看到控制对 API Gateway API 的访问 (p. 192).

类型：ApiAuth (p. 41)

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且不提供Amazon CloudFormation等效函数。

`BinaryMediaTypes`

您的 API 可能返回的 MIME 类型列表。使用此选项可以启用 API 的二进制支持。在 mime 类型中使用 ~1 代替/。

类型：List

必需：否

Amazon CloudFormation兼容性：此属性类似于`BinaryMediaTypes`一个的财产`AWS::ApiGateway::RestApi`资源。列表 BinaryMediaTypes 被添加到两个Amazon CloudFormation资源和 OpenAPI 文档。

`CacheClusterEnabled`

指示是否为阶段启用缓存。要缓存响应，还必须设置`CachingEnabled`到`true`下`MethodSettings`.

类型：布尔值

必需：否

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

`CacheClusterSize`

阶段的缓存群集大小。

类型：字符串

必需：否

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

`CanarySetting`

将 canary 设置配置为常规部署的某个阶段。

类型：CanarySetting

必需：否

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

`Cors`

管理所有 API Gateway API 的跨源资源共享 (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：CORS 需要Amazon SAM以修改您的 OpenAPI 定义。所以，它只有在内联时才起作用 OpenApi 定义为 DefinitionBody.

有关 CORS 的更多信息，请参阅。为 API Gateway 的 CORS 资源启用 CORS中的API Gateway 开发人员指南.

类型：字符串 |CorsConfiguration (p. 57)

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且不提供Amazon CloudFormation等效函数。

`DefinitionBody`

描述你的 API 的 OpenAPI 规范。如果两者都没有`DefinitionUri`也不`DefinitionBody`指定后，SAM 将生成一个`DefinitionBody`根据您的模板配置为您提供。

类型：JSON

必需：否

Amazon CloudFormation兼容性：此属性类似于`Body`一个的财产`AWS::ApiGateway::RestApi`资源。如果提供了某些属性，则可以将内容插入或修改到 DefinitionBody 在被传递给 CloudFormation. 房产包括`Auth`、`BinaryMediaTypes`、`Cors`、`GatewayResponses`、`Models`，还有一个`EventSource`对应的 Api 类型`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。如果两者都没有`DefinitionUri`也不`DefinitionBody`指定后，SAM 将生成一个`DefinitionBody`根据您的模板配置为您提供。

如果提供了本地文件路径，则模板必须通过包含`sam deploy`要么`sam package`命令，以便正确转换定义。

外部不支持内部函数 OpenApi 由引用了文件`DefinitionUri`. 请改用`DefinitionBody`属性带有包括转换导入 OpenApi 定义到模板中。

类型：字符串 |ApiDefinition (p. 56)

必需：否

Amazon CloudFormation兼容性：此属性类似于`BodyS3Location`一个的财产`AWS::ApiGateway::RestApi`资源。嵌套的 Amazon S3 属性的名称不同。

`Description`

Api 资源的描述。

类型：字符串

必需：否

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

`DisableExecuteApiEndpoint`

指定客户端是否可以使用默认`execute-api`终端节点`https://{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来调用 API，请禁用默认终端节点。

类型：布尔值

必需：否

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

`Domain`

为此 API Gateway API 配置自定义域。

类型：DomainConfiguration (p. 59)

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且不提供Amazon CloudFormation等效函数。

`EndpointConfiguration`

REST API 的终端节点类型。

类型：EndpointConfiguration (p. 63)

必需：否

Amazon CloudFormation兼容性：此属性类似于`EndpointConfiguration`一个的财产`AWS::ApiGateway::RestApi`资源。嵌套配置属性的名称不同。

`FailOnWarnings`

指定是否回滚 API 创建 (`true`) 还是不是 (`false`) 在遇到警告时。默认值为 `false`。

类型：布尔值

必需：否

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

`GatewayResponses`

配置 API 的网关响应。网关响应是 API Gateway 直接或通过使用 Lambda 授权方返回的响应。有关更多信息，请参阅文档。Ap网关 OpenApi 网响应.

类型：映射

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且不提供Amazon CloudFormation等效函数。

`MethodSettings`

配置 API 阶段的所有设置，包括日志记录、指标、CacheTTL、限制。

类型：MethodSettings

必需：否

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

`MinimumCompressionSize`

允许根据客户端的 Accept-Encoding 标头压缩响应正文。当响应正文大小大于或等于配置的阈值时，将触发压缩。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 和标识。

类型：整数

必需：否

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

`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 。Mode (模式)的财产AWS::ApiGateway::RestApi资源类型。

有效值：`overwrite` 或 `merge`

类型：字符串

必需：否

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

`Models`

您的 API 方法要使用的架构。可以使用 JON 格式或 YAML 描述这些模式。有关示例模型的更多信息，请参阅本页底部的示例部分。

类型：映射

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且不提供Amazon CloudFormation等效函数。

`Name`

API Gateway 的名称 RestApi 资源

类型：字符串

必需：否

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

OpenApiVersion

版本 OpenApi 来使用。此值可以是唯一的2.0对于 Swagger 规范，或者其中一个 OpenApi 3.0 版本，比如3.0.1. 有关 OpenAPI 的更多信息，请参阅。OpenAPI 规规规规.

注意：将此属性设置为任何有效值也会移除舞台StageSAM 创建的。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性对是唯一的Amazon SAM而且不提供Amazon CloudFormation等效函数。

StageName

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

要引用阶段资源，请使用*<api-logical-id>*.Stage. 有关引用的资源的更多信息，AWS::Serverless::Api (p. 32)资源已指定，请参阅Amazon CloudFormation指定 Amazon# Serverless# Api 时生成的资源 (p. 179). 有关的一般信息Amazon CloudFormation资源，请参阅生成 Amazon CloudFormation资源 (p. 177).

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性类似于StageName一个的财产AWS::ApiGateway::Stage资源。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 的阶段名称为 “Prod”。

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。有关标签的有效键和值的详细信息，请参阅资源标签中的Amazon CloudFormation用户指南.

类型：映射

必需：否

Amazon CloudFormation兼容性：此属性类似于Tags一个的财产AWS::ApiGateway::Stage资源。SAM 中的 Tags 属性由 Key: Value 对组成；在 CloudFormation 它包含Tag 对象的列表。

TracingEnabled

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

类型：布尔值

必需：否

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

`Variables`

一个映射（字符串到字符串），定义阶段变量，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：映射

必需：否

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

返回值

Ref

当该资源的逻辑 ID 提供给`Ref`内部函数时，它返回底层 API Gateway API 的 ID。

有关如何使用的更多信息`Ref`函数，请参阅`Ref`中的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。

有关的更多信息`Fn::GetAtt`，请参阅`Fn::GetAtt`中的Amazon CloudFormation用户指南.

`RootResourceId`

`RestApi` 资源的根资源 ID，例如 `a0bc123d4e`。

示例

SimpleApiExample

HelldAmazon SAM模板文件，其中包含带有 API 终端节点的 Lambda 函数。这是完整的Amazon SAM正在运行的无服务器应用程序的模板文件。

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

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模板代码段，其中包含在外部 Swagger 文件中定义的 API 以及 Lambda 集成和 CORS 配置。这只是其中的一部分Amazon SAM模板文件显示AWS::Serverless::Api (p. 32)定义。

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

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模板代码段，其中包含使用 Amazon Cognito 对该 API 的请求进行授权的 API。这只是其中的一部分Amazon SAM模板文件显示AWS::Serverless::Api (p. 32)定义。

YAML

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

ApiModelsExample

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包含模型架构的 API 的模板片段。这只是其中的一部分Amazon SAM模板文件，显示一个AWS::Serverless::Api (p. 32)定义有两个模型架构。

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

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

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。

有关使用配置访问的更多信息和示例Amazon SAM看到控制对 API Gateway API 的访问 (p. 192).

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

```
 AddDefaultAuthorizerToCorsPreflight: Boolean
 ApiKeyRequired: Boolean
 Authorizers: CognitoAuthorizer (p. 45) | LambdaTokenAuthorizer (p. 51)
| LambdaRequestAuthorizer (p. 48)
 DefaultAuthorizer: String
 InvokeRole: String
 ResourcePolicy: ResourcePolicyStatement (p. 53)
 UsagePlan: ApiUsagePlan (p. 43)
```

属性

`AddDefaultAuthorizerToCorsPreflight`

如果`DefaultAuthorizer`和`Cors`设置了属性，然后设置`AddDefaultAuthorizerToCorsPreflight`将导致默认授权者被添加到`Options`属性在 OpenAPI 部分中。

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

`ApiKeyRequired`

如果设置为 true，则所有 API 事件都需要 API 密钥。有关 API 键的更多信息，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.

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

`Authorizers`

用于控制对 API Gateway API 的访问的授权方。

有关更多信息，请参阅 控制对 API Gateway API 的访问 (p. 192)。

类型：Cognito 授权者 (p. 45)|lambdatoken授权者 (p. 51)|lambdaRequest 授权者 (p. 48)

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：SAM 将授权者添加到 Api 的 OpenAPI 定义中。

`DefaultAuthorizer`

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

注意：如果与此 API 关联的函数的 Api EventSource 配置为使用 IAM 权限，则必须将此属性设置为`AWS_IAM`，否则将会导致出现错误。

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

`InvokeRole`

将所有资源和方法的集成凭据设置为此值。

`CALLER_CREDENTIALS`映射到`arn:aws:iam::*:user/*`，它使用调用者凭据来调用终端节点。

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

类型：字符串

必需：否

默认值：`CALLER_CREDENTIALS`

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

`ResourcePolicy`

为 API 上的所有方法和路径配置资源策略。

类型：资源策略声明 (p. 53)

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

附加说明：此设置也可以在个人身上定义`AWS::Serverless::Function`使用ApiFunctionAuth (p. 89). 这对于具有`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创建和使用带 API 密钥的使用计划中的API Gateway 开发人员指南.

该Amazon SAM属性生成三个额外的Amazon CloudFormation设置此属性时的资源：AWS::ApiGateway::UsagePlan，一个AWS::ApiGateway::UsagePlanKey，还

有AWS::ApiGateway::ApiKey. 有关此方案的信息，请参阅。指定了 UsagePlan 属性 (p. 180). 有关生成的一般信息Amazon CloudFormation资源，请参阅生成Amazon CloudFormation资源 (p. 177).

类型：APIUSAGE 计划 (p. 43)

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

CognitoAuth

Cognito Auth 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`已在的 “全局变量” 部分Amazon SAM模板 (p. 28).

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. 192)。

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. 46)
  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. 46)

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. 192)。

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
  FunctionArn: String
  FunctionInvokeRole: String
  FunctionPayloadType: String
  Identity: LambdaRequestAuthorizationIdentity (p. 49)
```

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

`FunctionArn`

指定提供 API 授权的 Lambda 函数的函数 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

`FunctionInvokeRole`

将授权者凭据添加到 Lambda 授权者的 OpenAPI 定义中。

类型：字符串

必需：否

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

`FunctionPayloadType`

此属性可用于定义 API 的 Lambda 授权方类型。

有效值：`TOKEN` 或 `REQUEST`

类型：字符串

必需：否

默认值：`TOKEN`

Amazon CloudFormation兼容性：该属性对是唯一的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。此属性只有在`FunctionPayloadType`属性将设定为`REQUEST`.

类型：lambda 请求授权身份 (p. 49)

必需：条件

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

lambda RequestAuth

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. 192)。

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

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：该属性是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`FunctionArn`

指定提供 API 授权的 Lambda 函数的函数 arn。

类型：字符串

必需：是

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

`FunctionInvokeRole`

将授权者凭据添加到 Lambda 授权者的 OpenAPI 定义中。

类型：字符串

必需：否

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

`FunctionPayloadType`

此属性可用于定义 API 的 Lambda 授权方的类型。

有效值：`TOKEN` 或 `REQUEST`

类型：字符串

必需：否

默认值：`TOKEN`

Amazon CloudFormation兼容性：该属性是唯一的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。此属性只有在`FunctionPayloadType`属性将设定为`REQUEST`.

类型：lambdatoken授权身份 (p. 52)

必需：条件

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

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

此属性可用于在授权者的传入请求中指定 IdentitySource。有关 IdentitySource 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

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

属性

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

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

示例

lambdatoken身份

YAML

```
Identity:
  Header: Auth
  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.

语法

要在您的Amazon Serverless Application Model(Amazon SAM) 模板，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`

这些区域有：Amazon要阻止的账户。

类型：List

必需：否

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

`AwsAccountWhitelist`

这些区域有：Amazon允许的账户。有关此属性的示例使用，请参阅此页面底部的示例部分。

类型：List

必需：否

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

`CustomStatements`

要应用于此 API 的自定义资源策略声明的列表。有关此属性的示例使用，请参阅此页面底部的示例部分。

类型：List

必需：否

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

`IntrinsicVpcBlacklist`

要阻止的虚拟私有云 (VPC) 列表，其中每个 VPC 都被指定为参考，例如动态参考或者`Ref` 内部函数. 有关此属性的示例使用，请参阅此页面底部的示例部分。

类型：List

必需：否

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

`IntrinsicVpcWhitelist`

允许的 VPC 列表，其中每个 VPC 都被指定为参考，例如动态参考或者`Ref` 内部函数.

类型：List

必需：否

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

`IntrinsicVpceBlacklist`

要阻止的 VPC 终端节点列表，其中每个 VPC 终端节点都被指定为参考，例如动态参考或者`Ref` 内部函数.

类型：List

必需：否

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

`IntrinsicVpceWhitelist`

要允许的 VPC 终端节点列表，其中每个 VPC 终端节点都被指定为参考，例如动态参考或者`Ref` 内部函数. 有关此属性的示例使用，请参阅此页面底部的示例部分。

类型：List

必需：否

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

`IpRangeBlacklist`

要阻止的 IP 地址或地址范围。有关此属性的示例使用，请参阅此页面底部的示例部分。

类型：List

必需：否

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

`IpRangeWhitelist`

允许的 IP 地址或地址范围。

类型：List

必需：否

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

`SourceVpcBlacklist`

要阻止的源 VPC 或 VPC 终端节点。源 VPC 名称必须以`"vpc-"`并且源 VPC 终端节点名称必须以`"vpce-"`. 有关此属性的示例使用，请参阅此页面底部的示例部分。

类型：List

必需：否

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

`SourceVpcWhitelist`

要允许的源 VPC 或 VPC 终端节点。源 VPC 名称必须以`"vpc-"`并且源 VPC 终端节点名称必须以`"vpce-"`.

类型：List

必需：否

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

示例

资源策略示例

以下示例阻止两个 IP 地址和一个源 VPC，并允许Amazonaccount.

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. 32)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. 61)
  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`BasePath`的财产`AWS::ApiGateway::BasePathMapping`资源。Amazon SAM创建多个`AWS::ApiGateway::BasePathMapping`资源，每个一个`BasePath`此属性中指定。

`CertificateArn`

的 Amazon 资源名称 (ARN)Amazon托管证书此域名的终端节点。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`CertificateArn`的财产`AWS::ApiGateway::DomainName`资源。如果`EndpointConfiguration`设置为`REGIONAL`（默认值），`CertificateArn`映射到RegionalCertificateArn在`AWS::ApiGateway::DomainName`. 如果`EndpointConfiguration`设置为`EDGE`、`CertificateArn`映射到CertificateArn在`AWS::ApiGateway::DomainName`.

附加说明：对于`EDGE`终端节点，您必须在`us-east-1` Amazon区域。

`DomainName`

API Gateway API 的自定义域名。不支持大写字母。

Amazon SAM生成AWS::ApiGateway::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. 180). 有关生成的信息Amazon CloudFormation资源，请参阅生成 Amazon CloudFormation资源 (p. 177).

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::ApiGateway::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.

有效值：`REGIONAL` 或 `EDGE`

类型：字符串

必需：否

默认值：`REGIONAL`

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

`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::ApiGateway::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::ApiGateway::DomainName资源。

`Route53`

定义亚马逊路线 53 配置。

类型：Route53 配置 (p. 61)

必需：否

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

`SecurityPolicy`

此域名的 TLS 版本加密套件。

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::ApiGateway::DomainName`资源。

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

从Amazon Serverless Application Repository或从 Amazon S3 存储桶中将其作为嵌套应用程序。嵌套应用程序作为嵌套部署AWS::CloudFormation::Stack资源，可以包含多个其他资源，包括其他资源AWS::Serverless::Application (p. 64)资源的费用。

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

```
Type: AWS::Serverless::Application
Properties:
  Location: String | ApplicationLocationObject (p. 66)
  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 文档并包含有效的 CloudFormation 或 SAM 模板。网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的ApplicationLocationObject (p. 66)可用于指定已发布到Amazon Serverless Application Repository.

如果提供了本地文件路径，则模板必须通过包含`sam deploy`要么`sam package`命令，以便正确转换应用程序。

类型：字符串 |应用位置对象 (p. 66)

必需：是

Amazon CloudFormation兼容性：此属性类似于`TemplateURL`的财产`AWS::CloudFormation::Stack`资源。CloudFormation 版本不需要ApplicationLocationObject (p. 66)从中检索应用程序Amazon Serverless Application Repository.

`NotificationARNs`

将发送有关堆栈事件的通知的现有 Amazon SNS 主题的列表。

类型：List

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`NotificationARNs`的财产`AWS::CloudFormation::Stack`资源。

`Parameters`

应用程序参数值。

类型：Map

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`Parameters`的财产`AWS::CloudFormation::Stack`资源。

`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。键和值只能包含字母数字字字字符。密钥的长度可以为 1 到 127 个 Unicode 字符并且不能带有前缀 aws:。该值的长度可以为 1 到 255 个 Unicode 字符。

类型：Map

必需：否

Amazon CloudFormation兼容性：此属性类似于`Tags`的财产`AWS::CloudFormation::Stack`资源。SAM 中的 Tags 属性由键:Value 对组成；在 CloudFormation 中，它由标签对象的列表组成。创建堆栈时，SAM 将自动添加`lambda:createdBy:SAM`标签到此应用程序。此外，如果此应用程序来自Amazon Serverless Application Repository，那么 SAM 还会自动添加两个额外的标签`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：此属性将直接传递给`TimeoutInMinutes`的财产`AWS::CloudFormation::Stack`资源。

返回值

Ref

当该资源的逻辑 ID 提供给`Ref`内在函数，它返回底层的资源名称`AWS::CloudFormation::Stack`资源。

有关如何使用的更多信息`Ref`函数，请参阅`Ref`中的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`中的Amazon CloudFormation用户指南.

`Outputs.ApplicationOutputName`

具有名称的堆栈输出的值*`ApplicationOutputName`*.

示例

SAR 申请

使用无服务器应用程序存储库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
```

普通应用程序

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::Function

创建Amazon Lambda函数，Amazon Identity and Access Management(IAM) 执行角色和触发函数的事件源映射。

这些区域有：AWS::Serverless::Function (p. 67)资源也支持`Metadata`资源属性，所以你可以指示Amazon SAM以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的更多信息，请参阅构建自定义运行时 (p. 211).

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

```
Type: AWS::Serverless::Function
Properties:
  Architectures: List
  AssumeRolePolicyDocument: JSON
  AutoPublishAlias: String
  AutoPublishCodeSha256: String
  CodeSigningConfigArn: String
  CodeUri: String | FunctionCode (p. 130)
  DeadLetterQueue: Map | DeadLetterQueue (p. 77)
  DeploymentPreference: DeploymentPreference (p. 78)
  Description: String
  Environment: Environment
  EphemeralStorage: EphemeralStorage
  EventInvokeConfig: EventInvokeConfiguration (p. 81)
  Events: EventSource (p. 86)
  FileSystemConfigs: List
  FunctionName: String
  FunctionUrlConfig: FunctionUrlConfig (p. 131)
  Handler: String
  ImageConfig: ImageConfig
  ImageUri: String
  InlineCode: String
  KmsKeyArn: String
  Layers: List
  MemorySize: Integer
  PackageType: String
  PermissionsBoundary: String
  Policies: String | List | Map
  ProvisionedConcurrencyConfig: ProvisionedConcurrencyConfig
  ReservedConcurrentExecutions: Integer
  Role: String
  Runtime: String
```

```
  Tags: Map
  Timeout: Integer
  Tracing: String
  VersionDescription: String
  VpcConfig: VpcConfig
```

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Lambda 指令集架构中的Amazon Lambda开发人员指南.

有效值：其中一个`x86_64`要么`arm64`

类型：列表

必需：否

默认值：`x86_64`

Amazon CloudFormation兼容性：此属性将直接传递给`Architectures`的财产`AWS::Lambda::Function`资源。

`AssumeRolePolicyDocument`

添加 AssumeRolePolicyDocument 对于默认创建`Role`对于此函数。如果未指定此属性，Amazon SAM为此函数添加了默认代入角色。

类型：JSON

必需：否

Amazon CloudFormation兼容性：此属性类似于`AssumeRolePolicyDocument`的财产`AWS::IAM::Role`资源。Amazon SAM将此属性添加到此函数生成的 IAM 角色中。如果为此函数提供了角色的 Amazon 资源名称 (ARN)，则该属性不执行任何操作。

`AutoPublishAlias`

Lambda 别名的名称。有关 Lambda 别名的更多信息，请参阅Lambda 函数别名中的Amazon Lambda开发人员指南. 有关使用此属性的示例，请参阅逐步部署无服务器应用程 (p. 327).

Amazon SAM生成AWS::Lambda::Version和AWS::Lambda::Alias设置此属性时的资源。有关此方案的信息，请参阅。指定了 AutoPublishAlias 属性 (p. 181). 有关生成的一般信息Amazon CloudFormation资源，请参阅生成Amazon CloudFormation资源 (p. 177).

类型：字符串

必需：否

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

`AutoPublishCodeSha256`

使用的字符串值以及中的值。`CodeUri`，以确定是否应发布新的 Lambda 版本。

此属性解决了当Amazon SAM模板具有以下特性：`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 (p. 327))，`AutoPublishAlias`属性已设置且不会在部署之间更改，而`CodeUri`属性已设置，不会在部署之间更改。

当存储在 Amazon 简单存储服务 (Amazon S3) 位置的部署软件包被包含更新的 Lambda 函数代码的新部署包替换时，可能会发生这种情况，但`CodeUri`属性保持不变（与将新的部署包上传到新的 Amazon S3 位置和`CodeUri`正在更改为新位置）。

在这种情况下，要成功触发逐步部署，必须为`AutoPublishCodeSha256`.

类型：字符串

必需：否

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

`CodeSigningConfigArn`

的 ARNAWS::Lambda::CodeSigningConfig资源，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 配置代码签名Amazon SAM应用程序 (p. 201).

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`CodeSigningConfigArn`的财产`AWS::Lambda::Function`资源。

`CodeUri`

函数代码的 Amazon S3 URI、本地文件夹的路径或FunctionCode (p. 130)对象。此属性仅适用于`PackageType`属性将设定为`Zip`，否则将忽略它。

备注:

1. 如果`PackageType`属性将设定为`Zip`（默认值），然后是其中之一`CodeUri`要么`InlineCode`是必需的。

2. 如果是 Amazon S3 URI 或FunctionCode (p. 130)已提供对象，则引用的 Amazon S3 对象必须是有效的Lambda 部署程序包.

3. 如果提供了本地文件夹的路径，为了正确转换代码，模板必须通过包括sam build (p. 252)然后是sam deploy (p. 257)要么sam package (p. 273). 默认情况下，相对路径是相对于Amazon SAM模板的位置。

类型：字符串 |FunctionCode (p. 130)

必需：条件

Amazon CloudFormation兼容性：此属性类似于`Code`的财产`AWS::Lambda::Function`资源。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 将在队列中发送无法处理的事件。有关死信队列功能的更多信息，请参阅Amazon Lambda函数死信队列中的Amazon Lambda开发人员指南.

注意：如果 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。您为函数配置的死信队列用于函数的异步调用队列，不适用于事件源队列。

类型：| MapDeadLetter 队列 (p. 77)

必需：否

Amazon CloudFormation兼容性：此属性类似于DeadLetterConfig的财产AWS::Lambda::Function资源。InAmazon CloudFormation此类型派生自TargetArn，而在Amazon SAM你必须将类型与TargetArn.

DeploymentPreference

用于启用渐进 Lambda 部署的设置。

如果DeploymentPreference已指定对象，Amazon SAM创建AWS::CodeDeploy::Application叫ServerlessDeploymentApplication（每个堆栈一个），AWS::CodeDeploy::DeploymentGroup叫*<function-logical-id>*DeploymentGroup，还有AWS::IAM::Role叫CodeDeployServiceRole.

类型：DeploymentPreference (p. 78)

必需：否

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

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 (p. 327)。

Description

该函数的描述。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给Description的财产AWS::Lambda::Function资源。

Environment

运行时环境的配置。

类型：Environment

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给Environment的财产AWS::Lambda::Function资源。

EphemeralStorage

一个对象，它在中指定可供您的 Lambda 函数使用的磁盘空间（以 MB 为单位）/tmp.

有关此属性的更多信息，请参阅Lambda 执行环境中的Amazon Lambda开发人员指南.

类型：EphemeralStorage

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给EphemeralStorage的财产AWS::Lambda::Function资源。

EventInvokeConfig

描述 Lambda 函数上的事件调用配置的对象。

类型：事件调用配置 (p. 81)

必需：否

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

`Events`

指定触发此函数的事件。事件由一种类型和一组取决于类型的属性组成。

类型：EventSource (p. 86)

必需：否

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

`FileSystemConfigs`

的列表FileSystemConfig对象，指定 Amazon Elastic File System (Amazon EFS) 文件系统的连接设置。

如果你的模板包含AWS::EFS::MountTarget资源，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：列表

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`FileSystemConfigs`的财产`AWS::Lambda::Function`资源。

`FunctionName`

函数的名称。如果您没有指定名称，将为您生成一个唯一名称。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`FunctionName`的财产`AWS::Lambda::Function`资源。

`FunctionUrlConfig`

描述函数 URL 的对象。函数 URL 是可用于调用函数的 HTTPS 终端节点。

有关更多信息，请参阅 。函数 URL中的Amazon Lambda开发人员指南.

类型：函数 URLConfig (p. 131)

必需：否

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

`Handler`

代码中被调用以开始执行的函数。此属性只有在`PackageType`属性将设定为`Zip`.

类型：字符串

必需：条件

Amazon CloudFormation兼容性：此属性将直接传递给`Handler`的财产`AWS::Lambda::Function`资源。

`ImageConfig`

用于配置 Lambda 容器映像设置的对象。有关更多信息，请参阅 。在 Lambda 中使用容器镜像中的Amazon Lambda开发人员指南.

类型：ImageConfig

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给ImageConfig的财产AWS::Lambda::Function资源。

ImageUri

Lambda 函数的容器镜像的 Amazon Elastic Container Registry (Amazon ECR) 存储库的 URI。此属性仅适用于PackageType属性将设定为Image否则将忽略它。有关更多信息，请参阅 。在 Lambda 中使用容器镜像中的Amazon Lambda开发人员指南.

注意：如果PackageType属性将设定为Image，那么ImageUri是必需的，或者必须在必要的情况下构建应用程序Metadata中的条目Amazon SAM模板文件。有关更多信息，请参阅 构建应用程序 (p. 204)。

使用必要的方式构建应用Metadata条目优先于ImageUri，所以如果你同时指定两者那么ImageUri将忽略。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给ImageUri的财产AWS::Lambda::Function Code数据类型。

InlineCode

直接写入模板中的 Lambda 函数代码。此属性仅适用于PackageType属性将设定为Zip否则将忽略它。

注意：如果PackageType属性将设定为Zip（默认值），然后是其中之一CodeUri要么InlineCode是必需的。

类型：字符串

必需：条件

Amazon CloudFormation兼容性：此属性将直接传递给ZipFile的财产AWS::Lambda::Function Code数据类型。

KmsKeyArn

一个 ARNAmazon Key Management Service(Amazon KMS) Lambda 用于加密和解密函数环境变量的密钥。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给KmsKeyArn的财产AWS::Lambda::Function资源。

Layers

列表LayerVersion此函数应该使用的 ARN。此处指定的顺序是运行 Lambda 函数时它们的导入顺序。

类型：列表

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给Layers的财产AWS::Lambda::Function资源。

`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：此属性将直接传递给`MemorySize`的财产`AWS::Lambda::Function`资源。

`PackageType`

Lambda 函数的部署程序包类型。有关更多信息，请参阅 。Lambda 部署程序包中的Amazon Lambda开发人员指南.

备注:

1. 如果将此属性设置为`Zip`（默认值），然后`CodeUri`要么`InlineCode`适用，和`ImageUri`将忽略。

2. 如果将此属性设置为`Image`，仅限`ImageUri`适用，而且两者都有`CodeUri`和`InlineCode`将忽略。存储功能容器映像所需的 Amazon ECR 存储库可以由Amazon SAMCLI。有关更多信息，请参阅 sam deploy (p. 257)。

有效值：`Zip` 或 `Image`

类型：字符串

必需：否

默认值：`Zip`

Amazon CloudFormation兼容性：此属性将直接传递给`PackageType`的财产`AWS::Lambda::Function`资源。

`PermissionsBoundary`

用于此函数执行角色的权限边界的 ARN。只有在为您生成角色时，此属性才有效。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`PermissionsBoundary`的财产`AWS::IAM::Role`资源。

`Policies`

该函数需要的一个或多个策略。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，并且可以是Amazon托管策略或Amazon SAM策略模板或以 YAML 格式化的内联 IAM 策略文档。

有关 的更多信息Amazon托管策略，请参阅Amazon托管策略(在 IAM 用户指南中)。有关 的更多信息Amazon SAM策略模板，请参阅Amazon SAM策略模板 (p. 284)中的Amazon Serverless Application Model开发人员指南 的第一个版本。有关内联策略的更多信息，请参阅内联策略(在 IAM 用户指南中)。

注意：如果`Role`属性已设置，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`Policies`的财产`AWS::IAM::Role`资源。Amazon SAM支持Amazon托管策略名称Amazon SAM策略模板，以及 JSON 策略文档。Amazon CloudFormation仅接受 JSON 策略文档。

`ProvisionedConcurrencyConfig`

预置函数别名的并发配置。

注意：`ProvisionedConcurrencyConfig`只有在`AutoPublishAlias`已设置。否则将导致出错。

类型：ProvisionedConcurrencyConfig

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`ProvisionedConcurrencyConfig`的财产`AWS::Lambda::Alias`资源。

`ReservedConcurrentExecutions`

您想为函数预留的最大并发执行次数。

有关此属性的更多信息，请参阅Lambda 函数扩展中的Amazon Lambda开发人员指南.

类型：整数

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`ReservedConcurrentExecutions`的财产`AWS::Lambda::Function`资源。

`Role`

用作此函数执行角色的 IAM 角色的 ARN。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性类似于`Role`的财产`AWS::Lambda::Function`资源。此项为必填项Amazon CloudFormation但不是在Amazon SAM. 如果未指定角色，则会为您创建一个角色，其逻辑 ID 为*`<function-logical-id>`*`Role`.

`Runtime`

函数的运行时的标识符。此属性只有在`PackageType`属性将设定为`Zip`.

注意：如果你指定`provided`该属性的标识符，您可以使用`Metadata`要指示的资源属性Amazon SAM来构建此函数所需的自定义运行时。有关构建自定义运行时的更多信息，请参阅构建自定义运行时 (p. 211).

类型：字符串

必需：条件

Amazon CloudFormation兼容性：此属性将直接传递给`Runtime`的财产`AWS::Lambda::Function`资源。

`Tags`

一个映射（字符串到字符串），指定添加到此函数的标签。有关标签的有效密钥和值的详细信息，请参阅标签键和值要求中的Amazon Lambda开发人员指南.

创建堆栈时，Amazon SAM会自动添加`lambda:createdBy:SAM`标签到此 Lambda 函数，以及为此函数生成的默认角色。

类型：映射

必需：否

Amazon CloudFormation兼容性：此属性类似于Tags的财产AWS::Lambda::Function资源。这些区域有：Tags房地产在Amazon SAM由键/值对组成（而在Amazon CloudFormation此属性包含一系列Tag对象）。另请参阅，Amazon SAM会自动添加lambda:createdBy:SAM标签到此 Lambda 函数，以及为此函数生成的默认角色。

`Timeout`

函数在停止之前可以运行的最长时间（以秒为单位）。

类型：整数

必需：否

默认值：3

Amazon CloudFormation兼容性：此属性将直接传递给Timeout的财产AWS::Lambda::Function资源。

`Tracing`

指定函数的 X-Ray 跟踪模式的字符串。有关 X-Ray 的更多信息，请参阅使用Amazon Lambda和Amazon X-Ray中的Amazon Lambda开发人员指南.

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TracingConfig的财产AWS::Lambda::Function资源。如果Tracing属性将设定为Active和Role没有指定属性，那么Amazon SAM添加arn:aws:iam::aws:policy/AWSXrayWriteOnlyAccess针对它为您创建的 Lambda 执行角色的策略。

`VersionDescription`

指定Description添加到新 Lambda 版本资源中的字段。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给Description的财产AWS::Lambda::Version资源。

`VpcConfig`

使该函数能够访问虚拟私有云 (VPC) 中的私有资源的配置。

类型：VpcConfig

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给VpcConfig的财产AWS::Lambda::Function资源。

返回值

Ref

当该资源的逻辑 ID 提供给Ref内部函数，它返回底层 Lambda 函数的资源名称。

有关如何使用的更多信息`Ref`函数，请参阅`Ref`中的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`中的Amazon CloudFormation用户指南.

`Arn`

基础 Lambda 函数的 ARN。

示例

简单的函数

以下是一个基本示例：AWS::Serverless::Function (p. 67)程序包类型的资源`Zip`Amazon S3 存储桶中的函数代码和函数代码。

YAML

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

函数属性示例

以下是一个示例配置文件。AWS::Serverless::Function (p. 67)程序包的类型`Zip`（默认）使用`InlineCode`、`Layers`、`Tracing`、`Policies`、`Amazon EFS`，还有`Api`事件源。

YAML

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

```
          Resource: 'arn:aws:s3:::my-bucket/*'
  Events:
    ApiEvent:
      Type: Api
      Properties:
        Path: /path
        Method: get
```

示例：ImageConfig

以下是一个示例配置文件。`ImageConfig`对于包装类型的 Lambda 函数`Image`.

YAML

```
HelloWorldFunction:
  Type: AWS::Serverless::Function
  Properties:
    PackageType: Image
    ImageUri: account-id.dkr.ecr.region.amazonaws.com/ecr-repo-name:image-name
    ImageConfig:
      Command:
        - "app.lambda_handler"
      EntryPoint:
        - "entrypoint1"
      WorkingDirectory: "workDir"
```

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

指定启用渐进式 Lambda 部署的配置。有关配置渐进式 Lambda 部署的更多信息，请参阅逐步部署无服务器应用程 (p. 327).

注意：您必须指定`AutoPublishAlias`在你的AWS::Serverless::Function (p. 67)要使用`DeploymentPreference`对象，否则将会导致出现错误。

语法

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

YAML

```
  Alarms: List
  Enabled: Boolean
  Hooks: Hooks (p. 80)
  PassthroughCondition: Boolean
  Role: String
  TriggerConfigurations: List
  Type: String
```

属性

`Alarms`

列表 CloudWatch 警报您希望由部署中出现的任何错误触发的警报。

该酒店接受`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：列表

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

`Enabled`

是否启用此部署首选项。

类型：布尔值

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

`Hooks`

验证在流量转移之前和之后运行的 Lambda 函数。

类型：Hook (p. 80)

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

`PassthroughCondition`

如果为 True，且如果启用了此部署首选项，则函数的条件将传递给生成的 CodeDeploy 资源。通常，您应将此设置为 True。否则为 CodeDeploy 即使函数的条件解析为 False，也会创建资源。

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

`Role`

IAM 角色 ARN CodeDeploy 将用于交通转移。如果提供了 IAM 角色，将不会创建。

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

`TriggerConfigurations`

要与部署组关联的触发器配置列表。用于向 SNS 主题通知生命周期事件。

类型：列表

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`TriggerConfigurations`一个的财产`AWS::CodeDeploy::DeploymentGroup`资源。

`Type`

目前，有两种类别的部署类型：线性和金丝雀。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 (p. 327).

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

DeploymentPreference 使用交通前后的钩子。

包含流量前后挂钩的示例部署首选项。

YAML

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

DeploymentPreference 与 Fn# If 内部函数

使用的示例部署首选项`Fn::If`用于配置警报。在此示例中，`Alarm1`将在以下情况下配置`MyCondition`是`true`, 和`Alarm2`和`Alarm5`将在以下情况下配置`MyCondition`是`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。

注意：此属性中引用的 Lambda 函数配置`CodeDeployLambdaAliasUpdate`结果的对象AWS::Lambda::Alias资源。有关更多信息，请参阅 。CodeDeployLambdaAliasUpdate 策略中的Amazon CloudFormation用户指南.

语法

要在您的Amazon Serverless Application Model(Amazon SAM) 模板，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

`PreTraffic`

在流量转移之前运行的 Lambda 函数。

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

挂钩

挂钩函数示例

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. 82)
  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. 82)

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. 83)
  OnSuccess: OnSuccess (p. 84)
```

属性

`OnFailure`

处理失败的事件的目标。

类型：OnFailure (p. 83)

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. 84)

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。每个事件都由一个类型和一组依赖于该类型的属性组成。有关每个事件源的属性的更多信息，请参阅与该类型对应的主题。

语法

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

YAML

```
 Properties: S3 (p. 119) | SNS (p. 126) | Kinesis (p. 112) | DynamoDB (p. 99)
| SQS (p. 129) | Api (p. 88) | Schedule (p. 120) | CloudWatchEvent (p. 96)
| EventBridgeRule (p. 103) | CloudWatchLogs (p. 97) | IoTRule (p. 111)
| AlexaSkill (p. 87) | Cognito (p. 98) | HttpApi (p. 107) | MSK (p. 117) | MQ (p. 115)
| SelfManagedKafka (p. 124)
 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 Type。

类型：S3 (p. 119)|SNS (p. 126)|Kinesis (p. 112)|DynamoDB (p. 99)|SQS (p. 129)|API (p. 88)|Schedule (p. 120)|CloudW
事件 (p. 96)|EventBridgerRule (p. 103)|CloudWatchLogs (p. 97)|iotrRule (p. 111)|AlexasKill (p. 87)|Cognito (p. 98)|httPa
我管理的 Kafka (p. 124)

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

`Type`

事件类型。

有效值：`S3、SNS、Kinesis、DynamoDB、SQS、Api、Schedule、CloudWatchEvent、CloudWatchLogs、IoTRu`

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

apeVent

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. 32)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. 32)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. 89)
Method: String
Path: String
RequestModel: RequestModel (p. 94)
RequestParameters: String | RequestParameter (p. 95)
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. 89)

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. 32)资源。

类型：RequestModel (p. 94)

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. 95)

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. 32)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. 32)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. 32)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. 91)
```

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. 91)

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.

语法

要在您的Amazon Serverless Application Model(Amazon SAM) 模板，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

这些区域有：Amazon要阻止的账户。

类型：List

必需：否

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

`AwsAccountWhitelist`

这些区域有：Amazon允许的账户。有关此属性的示例，请参阅此页面底部的示例部分。

类型：List

必需：否

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

`CustomStatements`

要应用于此 API 的自定义资源策略声明的列表。有关此属性的示例，请参阅此页面底部的示例部分。

类型：List

必需：否

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

`IntrinsicVpcBlacklist`

要阻止的虚拟私有云 (VPC) 列表，其中每个 VPC 都被指定为参考，例如动态参考或者`Ref` 内部函数. 有关此属性的示例，请参阅此页面底部的示例部分。

类型：List

必需：否

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

`IntrinsicVpcWhitelist`

允许的 VPC 列表，其中每个 VPC 都被指定为参考，例如动态参考或者`Ref` 内部函数.

类型：List

必需：否

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

`IntrinsicVpceBlacklist`

要阻止的 VPC 终端节点列表，其中每个 VPC 终端节点都被指定为参考，例如动态参考或者`Ref` 内部函数.

类型：List

必需：否

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

`IntrinsicVpceWhitelist`

要允许的 VPC 终端节点列表，其中每个 VPC 终端节点都被指定为参考，例如动态参考或者`Ref` 内部函数. 有关此属性的示例，请参阅此页面底部的示例部分。

类型：List

必需：否

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

`IpRangeBlacklist`

要阻止的 IP 地址或地址范围。有关此属性的示例，请参阅此页面底部的示例部分。

类型：List

必需：否

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

`IpRangeWhitelist`

允许的 IP 地址或地址范围。

类型：List

必需：否

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

`SourceVpcBlacklist`

要阻止的源 VPC 或 VPC 终端节点。源 VPC 名称必须以`"vpc-"`并且源 VPC 终端节点名称必须以`"vpce-"`. 有关此属性的示例，请参阅此页面底部的示例部分。

类型：List

必需：否

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

`SourceVpcWhitelist`

要允许的源 VPC 或 VPC 终端节点。源 VPC 名称必须以`"vpc-"`并且源 VPC 终端节点名称必须以`"vpce-"`.

类型：List

必需：否

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

示例

资源策略示例

以下示例阻止两个 IP 地址和一个源 VPC，并允许Amazonaccount.

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. 32).

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。

Amazon Serverless Application Model(Amazon SAM) 生成AWS::Events::Rule设置此事件类型时的资源。

重要提示：EventBridgeRule (p. 103)是要使用的首选事件源类型，而不是`CloudWatchEvent`.`EventBridgeRule`和`CloudWatchEvent`使用相同的底层服务、API 和Amazon CloudFormation资源的费用。但是，Amazon SAM仅将对新功能的支持添加到`EventBridgeRule`.

语法

要在您的中声明该实体Amazon Serverless Application Model(Amazon SAM) 模板，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。如果您忽略该属性，Amazon SAM使用默认的事件总线。

类型：字符串

必需：否

默认值：默认事件总线

Amazon CloudFormation兼容性：此属性将直接传递给`EventBusName`的财产`AWS::Events::Rule`资源。

`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：此属性将直接传递给`Input`的财产`AWS::Events::Rule Target`资源。

`InputPath`

当您不希望将整个匹配的事件传递到目标时，请使用`InputPath`属性来描述要通过的事件的哪一部分。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`InputPath`的财产`AWS::Events::Rule Target`资源。

`Pattern`

描述哪些事件路由到指定目标。有关更多信息，请参阅 。EventBridge 中的事件和事件模式中的Amazon EventBridge 用户指南.

类型：EventPattern

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给`EventPattern`的财产`AWS::Events::Rule`资源。

示例

CloudWatch 事件

以下是的示例。`CloudWatchEvent`事件源类型。

YAML

```
CWEvent:
  Type: CloudWatchEvent
  Properties:
    Input: '{"Key": "Value"}'
    Pattern:
      detail:
        state:
          - terminated
```

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
```

DynamoDB

描述`DynamoDB`事件源类型。有关更多信息，请参阅 。使用Amazon Lambda使用 Amazon DynamoDB中的Amazon Lambda开发人员指南.

Amazon SAM生成AWS::Lambda::EventSourceMapping设置此事件类型时的资源。

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: Integer
  BisectBatchOnFunctionError: Boolean
  DestinationConfig: DestinationConfig
  Enabled: Boolean
  FilterCriteria: FilterCriteria
  FunctionResponseTypes: List
  MaximumBatchingWindowInSeconds: Integer
  MaximumRecordAgeInSeconds: Integer
  MaximumRetryAttempts: Integer
  ParallelizationFactor: Integer
  StartingPosition: String
  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：此属性将直接传递给`BatchSize`的财产`AWS::Lambda::EventSourceMapping`资源。

最低：`1`

最高：`1000`

`BisectBatchOnFunctionError`

如果函数返回错误，则将批次拆分为两批并重试。

类型：Boolean

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`BisectBatchOnFunctionError`的财产`AWS::Lambda::EventSourceMapping`资源。

`DestinationConfig`

用于丢弃记录的 Amazon Simple Queue Service (Amazon SQS) 队列或 Amazon Simple Notification Service (Amazon SNS) 主题目标。

类型：DestinationConfig

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`DestinationConfig`的财产`AWS::Lambda::EventSourceMapping`资源。

`Enabled`

禁用事件源映射以暂停轮询和调用。

类型：Boolean

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`Enabled`的财产`AWS::Lambda::EventSourceMapping`资源。

`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筛选事件中的Amazon Lambda开发人员指南.

类型：FilterCriteria

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`FilterCriteria`的财产`AWS::Lambda::EventSourceMapping`资源。

`FunctionResponseTypes`

当前应用于事件源映射的响应类型的列表。有关更多信息，请参阅 。报告批处理项目失败中的Amazon Lambda开发人员指南.

有效值：`ReportBatchItemFailures`

类型：List

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`FunctionResponseTypes`的财产`AWS::Lambda::EventSourceMapping`资源。

`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：此属性将直接传递给`MaximumBatchingWindowInSeconds`的财产`AWS::Lambda::EventSourceMapping`资源。

`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：此属性将直接传递给`MaximumRecordAgeInSeconds`的财产`AWS::Lambda::EventSourceMapping`资源。

`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：此属性将直接传递给`MaximumRetryAttempts`的财产`AWS::Lambda::EventSourceMapping`资源。

`ParallelizationFactor`

要从每个分片中同时处理的批次数。

类型：整数

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`ParallelizationFactor`的财产`AWS::Lambda::EventSourceMapping`资源。

`StartingPosition`

在流中开始读取数据的位置。

有效值：`TRIM_HORIZON` 或 `LATEST`

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给`StartingPosition`的财产`AWS::Lambda::EventSourceMapping`资源。

`Stream`

DynamoDB 流的 Amazon 资源名称 (ARN)。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给`EventSourceArn`的财产`AWS::Lambda::EventSourceMapping`资源。

`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 。滚动窗口中的Amazon Lambda开发人员指南.

类型：整数

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`TumblingWindowInSeconds`的财产`AWS::Lambda::EventSourceMapping`资源。

示例

现有 DynamoDB 表的 DynamoDB 事件源

DynamoDB 表的 DynamoDB 事件源，该表已存在于Amazonaccount.

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

用于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，该类型将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 。什么是 Amazon EventBridge？中的Amazon EventBridge 用户指南.

Amazon SAM生成AWS::Events::Rule设置此事件类型时的资源。

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

```
  DeadLetterConfig: DeadLetterConfig (p. 105)
  EventBusName: String
  Input: String
  InputPath: String
  Pattern: EventPattern
  RetryPolicy: RetryPolicy
  Target: Target (p. 106)
```

属性

`DeadLetterConfig`

在目标调用失败后配置 Amazon Simple Queue Service (Amazon SQS) 队列，其中 EventBridge 将事件发送到该队列。例如，在向不存在的 Lambda 函数发送事件时，或者当 EventBridge 没有足够的权限调用 Lambda 函数时，调用可能会失败。有关更多信息，请参阅 。事件重试策略和使用死信队列中的Amazon EventBridge 用户指南.

注意：这些区域有：AWS::Serverless::Function (p. 67)资源类型具有类似的数据类型，`DeadLetterQueue`，它处理成功调用目标 Lambda 函数后发生的故障。这些类型的失败示例包括 Lambda 限制或 Lambda 目标函数返回的错误。有关函数的更多信息`DeadLetterQueue`属性，请参阅Amazon Lambda函数死信队列中的Amazon Lambda开发人员指南.

类型：DeadLetterConfig (p. 105)

必需：否

Amazon CloudFormation兼容性：此属性类似于`DeadLetterConfig`的财产`AWS::Events::Rule Target`数据类型。这些区域有：Amazon SAM此属性的版本包括额外的子属性，以防你想Amazon SAM为您创建死信队列。

`EventBusName`

要与该规则关联的事件总线。如果您忽略此属性，Amazon SAM使用默认事件总线。

类型：字符串

必需：否

默认值：默认事件总线

Amazon CloudFormation兼容性：此属性将直接传递给`EventBusName`的财产`AWS::Events::Rule`资源。

`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：此属性将直接传递给`Input`的财产`AWS::Events::Rule Target`资源。

`InputPath`

当您不希望将整个匹配的事件传递到目标时，请使用`InputPath`属性来描述要通过的事件的哪一部分。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`InputPath`的财产`AWS::Events::Rule Target`资源。

`Pattern`

描述哪些事件路由到指定目标。有关更多信息，请参阅 。EventBridge 中的事件和事件模式中的Amazon EventBridge 用户指南.

类型：EventPattern

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给`EventPattern`的财产`AWS::Events::Rule`资源。

`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 。事件重试策略和使用死信队列中的Amazon EventBridge 用户指南.

类型：RetryPolicy

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`RetryPolicy`的财产`AWS::Events::Rule Target`数据类型。

`Target`

这些区域有：AmazonEventBridge 在触发规则时调用的资源。您可以使用此属性指定目标的逻辑 ID。如果未指定此属性，那么Amazon SAM生成目标的逻辑 ID。

类型：目标 (p. 106)

必需：否

Amazon CloudFormation兼容性：此属性类似于`Targets`的财产`AWS::Events::Rule`资源。这些区域有：Amazon SAM此属性的版本只允许您指定单个目标的逻辑 ID。

示例

EventBridgerRule

以下是的示例。`EventBridgeRule`事件源类型。

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

在目标调用失败后，EventBridge 将事件发送到的 Amazon Simple Queue Service (Amazon SQS) 队列的对象。例如，在向不存在的 Lambda 函数发送事件或者调用 Lambda 函数的权限不足时，调用可能会失败。有关更多信息，请参阅 。事件重试策略和使用死信队列中的Amazon EventBridge 用户指南.

注意：这些区域有：AWS::Serverless::Function (p. 67)资源类型具有类似的数据类型，`DeadLetterQueue`它处理成功调用目标 Lambda 函数后发生的故障。此类失败的示例包括 Lambda 限制或 Lambda 目标函数返回的错误。有关函数的更多信息`DeadLetterQueue`属性，请参阅Amazon Lambda函数死信队列中的Amazon Lambda开发人员指南 的第一个版本。

语法

要在您的Amazon Serverless Application Model(Amazon SAM) 模板，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 Amazon 资源名称 (ARN)。

注意：指定`Type`财产或`Arn`财产，但不能同时提供两者。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`Arn`的财产`AWS::Events::Rule DeadLetterConfig`数据类型。

`QueueLogicalId`

死信队列的自定义名称Amazon SAM创建如果`Type`已指定。

注意：如果`Type`未设置属性，将忽略此属性。

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

`Type`

队列的类型。设置此属性后，Amazon SAM自动创建死信队列并附加必要的基于资源的策略授予规则资源向队列发送事件的权限。

注意：指定`Type`财产或`Arn`财产，但不能同时提供两者。

有效值：`SQS`

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

DeadLetterConfig

DeadLetterConfig

YAML

```
DeadLetterConfig:
  Type: SQS
  QueueLogicalId: MyDLQ
```

Target

配置AmazonEventBridge 在触发规则时调用的资源。

语法

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

YAML

```
  Id: String
```

属性

`Id`

目标的逻辑 ID。

的价值`Id`可以包括字母数字字符、句点 (`.`)、连字符 (-) 和下划线 (_)。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给`Id`的财产`AWS::Events::Rule Target`数据类型。

示例

目标

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. 109)
  Method: String
  Path: String
  PayloadFormatVersion: String
  RouteSettings: RouteSettings
  TimeoutInMillis: Integer
```

属性

`ApiId`

的标识符AWS::Serverless::HttpApi (p. 132)在此模板中定义的资源。

如果没有定义，则默认值AWS::Serverless::HttpApi (p. 132)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. 132)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. 109)

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 的 “全局变量” 部分Amazon SAM模板 (p. 28)。

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。有关更多信息，请参阅 。使用Amazon Lambda使用 Amazon Kinesis中的Amazon Lambda开发人员指南.

Amazon SAM生成AWS::Lambda::EventSourceMapping设置此事件类型时的资源。

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: Integer
  BisectBatchOnFunctionError: Boolean
  DestinationConfig: DestinationConfig
  Enabled: Boolean
  FilterCriteria: FilterCriteria
  FunctionResponseTypes: List
  MaximumBatchingWindowInSeconds: Integer
  MaximumRecordAgeInSeconds: Integer
  MaximumRetryAttempts: Integer
  ParallelizationFactor: Integer
  StartingPosition: String
  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：此属性将直接传递给`BatchSize`一个的财产`AWS::Lambda::EventSourceMapping`资源。

最低：1

最高：10000

`BisectBatchOnFunctionError`

如果函数返回错误，则将批次拆分为两批并重试。

类型：Boolean

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`BisectBatchOnFunctionError`一个的财产`AWS::Lambda::EventSourceMapping`资源。

`DestinationConfig`

用于丢弃记录的 Amazon Simple Queue Service (Amazon SQS) 队列或 Amazon Simple Notification Service (Amazon SNS) 主题目标。

类型：DestinationConfig

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`DestinationConfig`一个的财产`AWS::Lambda::EventSourceMapping`资源。

`Enabled`

禁用事件源映射以暂停轮询和调用。

类型：Boolean

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`Enabled`一个的财产`AWS::Lambda::EventSourceMapping`资源。

`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筛选事件中的Amazon Lambda开发人员指南.

类型：FilterCriteria

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`FilterCriteria`一个的财产`AWS::Lambda::EventSourceMapping`资源。

`FunctionResponseTypes`

当前应用于事件源映射的响应类型的列表。有关更多信息，请参阅 。报告批处理项目失败中的Amazon Lambda开发人员指南.

有效值：`ReportBatchItemFailures`

类型：List

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`FunctionResponseTypes`一个的财产`AWS::Lambda::EventSourceMapping`资源。

`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：此属性将直接传递给`MaximumBatchingWindowInSeconds`一个的财产`AWS::Lambda::EventSourceMapping`资源。

`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：此属性将直接传递给`MaximumRecordAgeInSeconds`一个的财产`AWS::Lambda::EventSourceMapping`资源。

`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：此属性将直接传递给`MaximumRetryAttempts`一个的财产`AWS::Lambda::EventSourceMapping`资源。

`ParallelizationFactor`

要从每个分片中同时处理的批次数。

类型：整数

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`ParallelizationFactor`一个的财产`AWS::Lambda::EventSourceMapping`资源。

`StartingPosition`

在流中开始读取数据的位置。

有效值：`TRIM_HORIZON` 或 `LATEST`

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给`StartingPosition`一个的财产`AWS::Lambda::EventSourceMapping`资源。

`Stream`

数据流或流使用者的 Amazon 资源名称 (ARN)。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给`EventSourceArn`一个的财产`AWS::Lambda::EventSourceMapping`资源。

`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 。滚动窗口中的Amazon Lambda开发人员指南.

类型：整数

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`TumblingWindowInSeconds`一个的财产`AWS::Lambda::EventSourceMapping`资源。

示例

Kinesis 事件源

以下是 Kinesis 事件源示例。

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。有关更多信息，请参阅 。结合 Amazon MQ 使用 Lambda中的Amazon Lambda开发人员指南.

Amazon SAM生成AWS::Lambda::EventSourceMapping设置此事件类型时的资源。

注意：要在虚拟私有云 (VPC) 中拥有 Amazon MQ 队列，但您的
Lambda 函数在公共网络中，您的函数的执行角色必须包含以下权限：`ec2:CreateNetworkInterface`、`ec2:DeleteNetworkInterface`、`ec2:DescribeNetworkInterfaces`、
有关更多信息，请参阅 。执行角色权限中的Amazon Lambda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

```
  BatchSize: Integer
  Broker: String
  Enabled: Boolean
  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：此属性将直接传递给`BatchSize`一个的财产`AWS::Lambda::EventSourceMapping`资源。

最低：`1`

最高：`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：此属性将直接传递给`EventSourceArn`一个的财产`AWS::Lambda::EventSourceMapping`资源。

`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：Boolean

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`Enabled`一个的财产`AWS::Lambda::EventSourceMapping`资源。

`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：此属性将直接传递给`MaximumBatchingWindowInSeconds`一个的财产`AWS::Lambda::EventSourceMapping`资源。

`Queues`

要使用的 Amazon MQ 代理目标队列的名称。

类型：List

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给`Queues`一个的财产`AWS::Lambda::EventSourceMapping`资源。

`SecretsManagerKmsKeyId`

这些区域有：Amazon Key Management Service(Amazon KMS) 客户托管密钥的密钥 IDAmazon Secrets Manager. 如果您使用来自 Secret Manager 的客户托管密钥，则必须使用此属性，但是您的 Lambda 执行角色不包括`kms:Decrypt`权限。

此属性的价值是 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而且没有Amazon CloudFormation等效项

`SourceAccessConfigurations`

身份验证协议或虚拟主机的阵列。请使用指定此项SourceAccessConfigurations数据类型。

注意：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，凭证必须采用以下格式：`{"username": "your-username", "password": "your-password"}`. 只有一个类型的对象`BASIC_AUTH`允许使用。

`VIRTUAL_HOST`-您的 RabbitMQ 代理中虚拟主机的名称。Lambda 将使用此 Rabbit MQ 的主机作为事件源。只有一个类型的对象`VIRTUAL_HOST`允许使用。

类型：List

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给`SourceAccessConfigurations`一个的财产`AWS::Lambda::EventSourceMapping`资源。

示例

Amazon MQ 事件源

以下是一个示例配置文件`MQ`Amazon MQ 代理的事件源类型。

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

描述一个`MSK`事件源类型。有关更多信息，请参阅 。使用Amazon Lambda使用Amazon MSK中的Amazon Lambda开发人员指南.

Amazon SAM生成AWS::Lambda::EventSourceMapping设置此事件类型时的资源。

语法

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

YAML

```
  ConsumerGroupId: String
  MaximumBatchingWindowInSeconds: Integer
  StartingPosition: String
  Stream: String
  Topics: List
```

属性

`ConsumerGroupId`

一个字符串，用于配置如何从 Kafka 主题中读取事件。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`AmazonManagedKafkaConfiguration`一个的财产`AWS::Lambda::EventSourceMapping`资源。

`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：此属性将直接传递给`MaximumBatchingWindowInSeconds`一个的财产`AWS::Lambda::EventSourceMapping`资源。

`StartingPosition`

在流中开始读取数据的位置。

有效值：`TRIM_HORIZON` 或 `LATEST`

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给`StartingPosition`一个的财产`AWS::Lambda::EventSourceMapping`资源。

`Stream`

数据流或流使用者的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`EventSourceArn`一个的财产`AWS::Lambda::EventSourceMapping`资源。

`Topics`

Kafka 主题的名称。

类型：List

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给`Topics`一个的财产`AWS::Lambda::EventSourceMapping`资源。

示例

现有集群的 Amazon MSK 示例

以下是的示例`MSK`Amazon MSK 集群的事件源类型，该集群已存在于Amazonaccount.

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

以下是的示例`MSK`在同一模板文件中声明的 Amazon MSK 集群的事件源类型。

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，它将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 。什么是 Amazon EventBridge？中的Amazon EventBridge 用户指南.

Amazon Serverless Application Model(Amazon SAM) 生成AWS::Events::Rule设置此事件类型时的资源。

语法

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

YAML

```
DeadLetterConfig: DeadLetterConfig (p. 123)
Description: String
Enabled: Boolean
Input: String
Name: String
RetryPolicy: RetryPolicy
Schedule: String
```

属性

`DeadLetterConfig`

在目标调用失败后，配置 Amazon Simple Queue Service (Amazon SQS) 队列，其中 EventBridge 将发送事件。例如，在向不存在的 Lambda 函数发送事件时，或者当 EventBridge 没有足够的权限调用 Lambda 函数时，调用可能会失败。有关更多信息，请参阅 。事件重试策略和使用死信队列中的Amazon EventBridge 用户指南.

注意：这些区域有：AWS::Serverless::Function (p. 67)资源类型有类似的数据类型，`DeadLetterQueue`，它处理成功调用目标 Lambda 函数后发生的故障。这些类型的失败示例包括 Lambda 限制或 Lambda 目标函数返回的错误。有关函数的更多信息`DeadLetterQueue`属性，请参阅Amazon Lambda函数死信队列中的Amazon Lambda开发人员指南.

类型：DeadLetterConfig (p. 123)

必需：否

Amazon CloudFormation兼容性：此属性类似于`DeadLetterConfig`的财产`AWS::Events::RuleTarget`数据类型。这些区域有：Amazon SAM此属性的版本包括额外的子属性，以防你想Amazon SAM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：此属性将直接传递给`Description`的财产`AWS::Events::Rule`资源。

`Enabled`

指示是否启用规则。

要禁用规则，请将此属性设置为`false`.

类型：Boolean

必需：否

Amazon CloudFormation兼容性：此属性类似于`State`的财产`AWS::Events::Rule`资源。如果此属性设置为`true`然后Amazon SAM通行证`ENABLED`，否则它会通过`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：此属性将直接传递给`Target`的财产`AWS::Events::Rule Target`资源。

`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：此属性将直接传递给`Name`的财产`AWS::Events::Rule`资源。

`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 。事件重试策略和使用死信队列中的Amazon EventBridge 用户指南.

类型：RetryPolicy

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`RetryPolicy`的财产`AWS::Events::Rule Target`数据类型。

`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：此属性将直接传递给`ScheduleExpression`的财产`AWS::Events::Rule`资源。

示例

CloudWatch Event Event

CloudWatch Schedule Event Event

YAML

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

DeadLetterConfig

用于指定在失败的目标调用后，EventBridge 将事件发送的 Amazon Simple Queue Service (Amazon SQS) 队列。例如，在向不存在的 Lambda 函数发送事件或者调用 Lambda 函数的权限不足时，调用可能会失败。有关更多信息，请参阅 。事件重试策略和使用死信队列中的Amazon EventBridge 用户指南.

注意：这些区域有：AWS::Serverless::Function (p. 67)资源类型具有类似的数据类型，`DeadLetterQueue`它处理成功调用目标 Lambda 函数后发生的故障。此类失败的示例包括 Lambda 限制或 Lambda 目标函数返回的错误。有关函数的更多信息`DeadLetterQueue`属性，请参阅Amazon Lambda函数死信队列中的Amazon Lambda开发人员指南 的第一个版本。

语法

要在您的Amazon Serverless Application Model(Amazon SAM) 模板，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 Amazon 资源名称 (ARN)。

注意：指定`Type`财产或`Arn`财产，但不是两者。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`Arn`的财产`AWS::Events::Rule DeadLetterConfig`数据类型。

`QueueLogicalId`

死信队列的自定义名称Amazon SAM创建如果`Type`已指定。

注意：如果`Type`未设置属性，将忽略此属性。

类型：字符串

必需：否

Amazon CloudFormation兼容性：该物业对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

`Type`

队列的类型。设置此属性时，Amazon SAM自动创建死信队列并附加必要的基于资源的策略授予规则资源向队列发送事件的权限。

注意：指定`Type`财产或`Arn`财产，但不是两者。

有效值：`SQS`

类型：字符串

必需：否

Amazon CloudFormation兼容性：该物业对是唯一的Amazon SAM而且没有Amazon CloudFormation等效项

示例

DeadLetterConfig

DeadLetterConfig

YAML

```
DeadLetterConfig:
  Type: SQS
  QueueLogicalId: MyDLQ
```

SelfManagedKafka

描述一个`SelfManagedKafka`事件源类型。有关更多信息，请参阅 。使用Amazon Lambda使用自行管理的Apache Kafka中的Amazon Lambda开发人员指南.

Amazon SAM生成AWS::Lambda::EventSourceMapping设置此事件类型时的资源。

语法

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

YAML

```
  BatchSize: Integer
  ConsumerGroupId: String
  Enabled: Boolean
  KafkaBootstrapServers: List
  SourceAccessConfigurations: SourceAccessConfiguration
  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：此属性将直接传递给`BatchSize`一个的财产`AWS::Lambda::EventSourceMapping`资源。

最小值：`1`

最大值：`10000`

`ConsumerGroupId`

一个字符串，用于配置如何从 Kafka 主题中读取事件。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`SelfManagedKafkaConfiguration`一个的财产`AWS::Lambda::EventSourceMapping`资源。

`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：此属性将直接传递给`Enabled`一个的财产`AWS::Lambda::EventSourceMapping`资源。

`KafkaBootstrapServers`

适用于Kafka代理的 Bootstrap 服务器列表。例如，包括端口`broker.example.com:xxxx`

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等效函数。

`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。

类型：SourceAccessConfiguration

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给`SourceAccessConfigurations`一个的财产`AWS::Lambda::EventSourceMapping`资源。

`Topics`

Kafka 主题的名称。

类型：List

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给`Topics`一个的财产`AWS::Lambda::EventSourceMapping`资源。

示例

自行管理的 Kafka 事件源

以下是的示例`SelfManagedKafka`事件源类型。

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。

SAM 生成AWS::SNS::Subscription设置此事件类型时的资源

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

```
  FilterPolicy: SnsFilterPolicy
  Region: String
  SqsSubscription: Boolean | SqsSubscriptionObject (p. 127)
  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 。GetSubscriptionAttributes请参阅 Amazon Simple Notification Service API 参考中的。

类型：SNS 过滤器策略

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`FilterPolicy`一个的财产`AWS::SNS::Subscription`资源。

`Region`

对于跨区域订阅，为主题所在的区域。

如果没有指定区域，CloudFormation 将使用发起人的区域作为默认区域。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`Region`一个的财产`AWS::SNS::Subscription`资源。

`SqsSubscription`

将此属性设置为 true，或者指定`SqsSubscriptionObject`以在 SQS 队列中启用对 SNS 主题通知进行批处理。将此属性设置为`true`创建一个新的 SQS 队列，而指定`SqsSubscriptionObject`使用现有 SQS 队列。

类型：布尔值 |SQSSS订阅对象 (p. 127)

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。

`Topic`

要订阅的主题的 ARN。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给`TopicArn`一个的财产`AWS::SNS::Subscription`资源。

示例

SNS 事件源示例

SNS 事件源示例

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。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。

SAM 生成AWS::Lambda::EventSourceMapping设置此事件类型时的资源

语法

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

YAML

```
  BatchSize: Integer
  Enabled: Boolean
  FilterCriteria: FilterCriteria
  MaximumBatchingWindowInSeconds: Integer
  Queue: String
```

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：此属性将直接传递给`BatchSize`的财产`AWS::Lambda::EventSourceMapping`资源。

最低：`1`

最高：`10000`

`Enabled`

禁用事件源映射以暂停轮询和调用。

类型：Boolean

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`Enabled`的财产`AWS::Lambda::EventSourceMapping`资源。

`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筛选事件中的Amazon Lambda开发人员指南.

类型：FilterCriteria

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`FilterCriteria`的财产`AWS::Lambda::EventSourceMapping`资源。

`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：此属性将直接传递给`MaximumBatchingWindowInSeconds`的财产`AWS::Lambda::EventSourceMapping`资源。

`Queue`

队列的 ARN。

类型：字符串

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给`EventSourceArn`的财产`AWS::Lambda::EventSourceMapping`资源。

示例

SQS 事件

SQS 事件

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"]}'
```

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，使用该 API，您可以创建比 REST API 具有更低延迟和更低成本的 RESTful API。有关更多信息，请参阅 。使用 HTTP API中的API Gateway 开发人员指南.

建议使用Amazon CloudFormation挂钩或 IAM 策略，以验证 API Gateway 资源是否附加了授权者以控制对它们的访问。

有关使用的更多信息Amazon CloudFormation挂钩，请参阅注册挂钩中的Amazon CloudFormationCLI 用户指南和apigw 执行授权者 GitHub 存储库。

有关使用 IAM 策略的更多信息，请参阅要求 API 路由具有授权中的API Gateway 开发人员指南.

语法

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

```
Type: AWS::Serverless::HttpApi
Properties:
  AccessLogSettings: AccessLogSettings
  Auth: HttpApiAuth (p. 139)
  CorsConfiguration: String | HttpApiCorsConfiguration (p. 145)
  DefaultRouteSettings: RouteSettings
  DefinitionBody: JSON
  DefinitionUri: String | HttpApiDefinition (p. 147)
  Description: String
  DisableExecuteApiEndpoint: Boolean
  Domain: HttpApiDomainConfiguration (p. 148)
  FailOnWarnings: Boolean
  RouteSettings: RouteSettings
  StageName: String
  StageVariables: Json
  Tags: Map
```

属性

`AccessLogSettings`

阶段访问日志记录的设置。

类型：AccessLogSettings

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`AccessLogSettings`的财产`AWS::ApiGatewayV2::Stage`资源。

`Auth`

配置授权以控制对 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：httpPiAuth (p. 139)

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

`CorsConfiguration`

管理所有 API Gateway HTTP API 的跨源资源共享 (CORS)。指定要允许的域作为字符串，或者指定`HttpApiCorsConfiguration`对象。请注意，CORS 需要Amazon SAM修改你的 OpenAPI 定义，所以 CORS 只有在`DefinitionBody`属性已指定。

有关更多信息，请参阅 。为 HTTP API 配置 CORS中的API Gateway 开发人员指南.

注意：如果CorsConfiguration在 OpenAPI 定义和属性级别都设置了，然后Amazon SAM将两个配置源与优先属性合并。

注意：如果此属性设置为true，那么所有的起源都被允许。

类型：字符串 |httpPicors配置 (p. 145)

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

`DefaultRouteSettings`

此 HTTP API 的默认路由设置。除非被覆盖，否则这些设置适用于所有路由。RouteSettings某些路线的属性。

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。

`DefinitionBody`

描述你的 HTTP API API 的 OpenAPI 定义。如果不指定DefinitionUri或者DefinitionBody、Amazon SAM生成DefinitionBody根据你的模板配置。

类型：JSON

必需：否

Amazon CloudFormation兼容性：此属性类似于Body的财产AWS::ApiGatewayV2::Api资源。如果提供了某些房产，Amazon SAM可以将内容插入或修改DefinitionBody在传递给之前Amazon CloudFormation. 属性包括Auth和EventSource的类型 HttpApi 对于相应的AWS::Serverless::Function资源。

`DefinitionUri`

定义 HTTP API 的 OpenAPI 定义的 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。如果不指定DefinitionUri或者DefinitionBody已指定，Amazon SAM生成DefinitionBody根据你的模板配置。

如果您提供了本地文件路径，则模板必须通过包含sam deploy要么sam package命令以便正确转换定义。

外部不支持内部函数 OpenApi 你引用的定义文件DefinitionUri. 导入 OpenApi 定义到模板中，使用DefinitionBody属性与包括转换.

类型：字符串 |HTTPAPI 定义 (p. 147)

必需：否

Amazon CloudFormation兼容性：此属性类似于BodyS3Location的财产AWS::ApiGatewayV2::Api资源。嵌套的 Amazon S3 属性的命名不同。

`Description`

对的描述 HttpApi 资源。

注意：此属性需要Amazon SAM修改 HttpApi 资源的 OpenAPI 定义，用于设置description字段中返回的子位置类型。以下两种情况导致出现错误：(1) 该DefinitionBody属性是

用`description`OpenAPI 定义中设置的字段（因为这是一个冲突Amazon SAM不会解决），或者 2）`DefinitionUri`属性被指定（因为Amazon SAM不会修改它从 Amazon S3 中检索的 OpenAPI 定义）。

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

`DisableExecuteApiEndpoint`

指定客户端是否可以使用默认设置调用您的 HTTP API。`execute-api`终端节点`https://{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来调用 API，请禁用默认终端节点。

类型：布尔值

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`DisableExecuteApiEndpoint`的财产`AWS::ApiGatewayV2::Api`资源。

`Domain`

为此 API Gateway HTTP API 配置自定义域。

类型：httpPid域名配置 (p. 148)

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

`FailOnWarnings`

指定是否回滚 HTTP API 创建 (`true`) 或不 (`false`) 遇到警告时。原定设置值为 `false`。

类型：布尔值

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`FailOnWarnings`的财产`AWS::ApiGatewayV2::Api`资源。

`RouteSettings`

此 HTTP API 的路由设置（每条路由）。有关更多信息，请参阅 。使用 HTTP API 的路由中的API Gateway 开发人员指南.

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。

`StageName`

API 阶段的名称。如果未指定名称，Amazon SAM使用`$default`来自 API Gateway 的阶段。

类型：字符串

必需：否

默认值：$default

Amazon CloudFormation兼容性：此属性将直接传递给`StageName`的财产`AWS::ApiGatewayV2::Stage`资源。

`StageVariables`

一个定义阶段变量的映射。变量名称可以有字母数字和下划线字符。值必须匹配 [A-Za-z0-9-._~:/? #&=,]+。

类型：Json

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`StageVariables`的财产`AWS::ApiGatewayV2::Stage`资源。

`Tags`

指定要添加到 API Gateway 阶段的标签的映射（字符串到字符串）。键的长度可以是 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。该值的长度可以是 1 到 256 个 Unicode 字符。

类型：映射

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

附加说明：这些区域有：`Tags`属性需要Amazon SAM修改 OpenAPI 定义，因此只有在`DefinitionBody`属性已指定-如果`DefinitionUri`属性已指定。Amazon SAM自动添加`httpapi:createdBy:SAM`标签。标签也被添加到`AWS::ApiGatewayV2::Stage`资源和`AWS::ApiGatewayV2::DomainName`资源（如果`DomainName`已指定）。

返回值

Ref

将此资源的逻辑 ID 传递给内部`Ref`函数，`Ref`返回底层证券的 API ID`AWS::ApiGatewayV2::Api`例如，资源，`a1bcdef2gh`.

有关使用的更多信息`Ref`函数，请参阅`Ref`中的Amazon CloudFormation用户指南.

示例

Simple 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。此示例使用默认 HTTP API，Amazon SAM创建。

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
```

带身份验证的 HTTPi

以下示例说明如何在 HTTP API 终端节点上设置授权。

YAML

```
Properties:
  FailOnWarnings: true
  Auth:
    DefaultAuthorizer: OAuth2
    Authorizers:
      OAuth2:
        AuthorizationScopes:
          - scope4
        JwtConfiguration:
          issuer: "https://www.example.com/v1/connect/oauth2"
          audience:
            - MyApi
        IdentitySource: "$request.querystring.param"
      OpenIdAuth:
        AuthorizationScopes:
          - scope1
          - scope2
        OpenIdConnectUrl: "https://www.example.com/v1/connect/oidc/.well-known/openid-
configuration"
        JwtConfiguration:
          issuer: "https://www.example.com/v1/connect/oidc"
          audience:
            - MyApi
        IdentitySource: "$request.querystring.param"
```

带 OpenAPI 定义的 HTTPi

以下示例说明如何将 OpenAPI 定义添加到模板。

请注意Amazon SAM填写任何缺少的 Lambda 集成 HttpApi 引用此 HTTP API 的事件。Amazon SAM还会添加任何缺失的路径 HttpApi 参考事件。

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
```

带配置设置的 HTTPi

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. 144) | LambdaAuthorizer (p. 140)
  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. 144)|LambdaAuthorizer (p. 140)

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. 142)
```

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. 142)

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 授权方的定义，也称为 JSON Web Token (JWT) 授权方。

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
  IdentitySource: String
  JwtConfiguration: Map
```

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

`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而且没有Amazon CloudFormation等效项

`JwtConfiguration`

此授权者的 JWT 配置。

这将传递到`jwtConfiguration`的部分`x-amazon-apigateway-authorizer`中的`securitySchemes`OpenAPI 定义的部分。

类型：Map

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

OAuth 2.0 授权方

OAuth 2.0 授权方示例

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. 150)
  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. 185). 有关生成的信息Amazon CloudFormation资源，请参阅生成 Amazon CloudFormation资源 (p. 177).

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. 150)

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：域名称CloudFront 分配.

EvaluateTargetHealth

如果 EvaluateTargetHealth 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
```

AWS::Serverless::LayerVersion

创建一个 Lambda LayerVersion，其中包含 Lambda 函数所需的库或运行时代码。

这些区域有：AWS::Serverless::LayerVersion (p. 152)资源也支持Metadata资源属性，所以你可以指示Amazon SAM以构建应用程序中包含的图层。有关构建层的更多信息，请参阅。构建层 (p. 209).

重要提示：自发布以来UpdateReplacePolicy中的资源属性Amazon CloudFormation、AWS::Lambda::LayerVersion（推荐）提供与AWS::Serverless::LayerVersion (p. 152).

转换无服务器 LayerVersion 时，SAM 还会转换资源的逻辑 ID，以便在更新资源时，CloudFormation 不会自动删除旧的 LayerVersion。

语法

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

YAML

```
Type: AWS::Serverless::LayerVersion
Properties:
  CompatibleArchitectures: List
  CompatibleRuntimes: List
  ContentUri: String | LayerContent (p. 154)
  Description: String
  LayerName: String
  LicenseInfo: String
  RetentionPolicy: String
```

属性

`CompatibleArchitectures`

指定图层版本支持的指令集体系结构。

有关此属性的更多信息，请参阅Lambda 指令集架构中的Amazon Lambda开发人员指南.

有效值：`x86_64`、`arm64`

类型：List

必需：否

默认值：`x86_64`

Amazon CloudFormation兼容性：此属性将直接传递给`CompatibleArchitectures`的财产`AWS::Lambda::LayerVersion`资源。

`CompatibleRuntimes`

与此 LayerVersion 兼容的运行时列表。

类型：List

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`CompatibleRuntimes`的财产`AWS::Lambda::LayerVersion`资源。

`ContentUri`

Amazon S3 Uri、本地文件夹的路径或图层代码的 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.

如果提供了本地文件夹的路径，为了正确转换内容，模板必须通过包括sam build (p. 252)然后是sam deploy (p. 257)要么sam package (p. 273). 默认情况下，相对路径是相对于Amazon SAM模板的位置。

类型：字符串 |图层内容 (p. 154)

必需：是

Amazon CloudFormation兼容性：此属性类似于`Content`的财产`AWS::Lambda::LayerVersion`资源。嵌套的 Amazon S3 属性的命名不同。

`Description`

此图层的描述。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`Description`的财产`AWS::Lambda::LayerVersion`资源。

`LayerName`

层的名称或 Amazon Resource Name (ARN)。

类型：字符串

必需：否

默认值：资源逻辑 ID

Amazon CloudFormation兼容性：此属性类似于`LayerName`的财产`AWS::Lambda::LayerVersion`资源。如果您没有指定名称，则将使用资源的逻辑 ID 作为名称。

`LicenseInfo`

有关此 LayerVersion 的许可证的信息。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`LicenseInfo`的财产`AWS::Lambda::LayerVersion`资源。

`RetentionPolicy`

指定更新后是保留还是删除 LayerVersion 的旧版本。

有效值：`Retain` 或 `Delete`

类型：字符串

必需：否

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

附加说明：当你指定`Retain`、Amazon SAM添加资源属性 (p. 176)的`DeletionPolicy: Retain`对变换`AWS::Lambda::LayerVersion`资源。

返回值

Ref

当此资源的逻辑 ID 提供给`Ref`内部函数，它返回底层 Lambda LayerVersion 的资源 ARN。

有关如何使用的更多信息`Ref`函数，请参阅`Ref`中的Amazon CloudFormation用户指南.

示例

图层版本示例

LayerVersion 的示例

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

使用单个属性主密钥创建 DynamoDB 表。当只需要通过主键访问数据时，它非常有用。

要使用 DynamoDB 的更高级功能，请使用AWS::DynamoDB::Table请改用资源。

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

```
Type: AWS::Serverless::SimpleTable
Properties:
  PrimaryKey: PrimaryKeyObject (p. 157)
  ProvisionedThroughput: ProvisionedThroughput
  SSESpecification: SSESpecification
```

```
  TableName: String
  Tags: Map
```

属性

`PrimaryKey`

要用作表主键的属性名称和类型。如果未提供，则主键将是`String`价值为`id`.

注意：创建此资源后，无法修改此属性的值。

类型：PrimaryKeyObject (p. 157)

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

`ProvisionedThroughput`

读取和写入吞吐量配置信息。

如果`ProvisionedThroughput`未指定`BillingMode`将被指定为`PAY_PER_REQUEST`.

类型：ProvisionedThroughput

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`ProvisionedThroughput`一个的财产`AWS::DynamoDB::Table`资源。

`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：此属性将直接传递给`SSESpecification`一个的财产`AWS::DynamoDB::Table`资源。

`TableName`

DynamoDB 表的名称。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`TableName`一个的财产`AWS::DynamoDB::Table`资源。

`Tags`

一个映射（字符串到字符串），指定要添加到此 SimpleTable 的标签。有关标签的有效密钥和值的详细信息，请参阅资源标签中的Amazon CloudFormation用户指南.

类型：映射

必需：否

Amazon CloudFormation兼容性：此属性类似于`Tags`一个的财产`AWS::DynamoDB::Table`资源。SAM 中的 Tags 属性由键:Value 对组成； CloudFormation 它包含 Tag 对象的列表。

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 DynamoDB 表的资源名称。

有关如何使用的更多信息`Ref`函数，请参阅`Ref`中的Amazon CloudFormation用户指南.

示例

简单的表示例

SimpleTTable 的示例

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，你可以用它来编排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。

注意：要管理Amazon SAM包含 Step Functions 状态机的模板，您必须使用版本 0.52.0 或更高版本Amazon SAMCLI。要检查您具有的版本，请运行命令`sam --version`.

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

```
Type: AWS::Serverless::StateMachine
Properties:
  Definition: Map
  DefinitionSubstitutions: Map
  DefinitionUri: String | S3Location
  Events: EventSource (p. 162)
  Logging: LoggingConfiguration
  Name: String
  PermissionsBoundary: String
  Policies: String | List | Map
  Role: String
  Tags: Map
  Tracing: TracingConfiguration
  Type: String
```

属性

`Definition`

状态机定义是一个对象，其中的对象格式与您的格式相匹配Amazon SAM模板文件，例如 JSON 或 YAML。状态机定义遵循 Amazon 状态语言。

有关内联状态机定义的示例，请参阅示例 (p. 161).

您必须提供`Definition`或者`DefinitionUri`.

类型：Map

必需：条件

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

`DefinitionSubstitutions`

一个字符串到字符串映射，它指定状态机定义中占位符变量的映射。这使您能够将在运行时获得的值（例如从内部函数）注入状态机定义。

类型：Map

必需：否

Amazon CloudFormation兼容性：此属性类似于`DefinitionSubstitutions`的财产`AWS::StepFunctions::StateMachine`资源。如果在内联状态机定义中指定了任何内部函数，Amazon SAM向此属性添加条目以将它们注入到状态机定义中。

`DefinitionUri`

写入的状态机定义的 Amazon Simple Storage Service (Amazon S3) URI 或本地文件路径Amazon States Language.

如果您提供了本地文件路径，则模板必须通过包含`sam deploy`要么`sam package`命令来正确转换定义。为此，您必须使用版本 0.52.0 或更高版本Amazon SAMCLI。

您必须提供`Definition`或者`DefinitionUri`.

类型：字符串 |S3 位置

必需：条件

Amazon CloudFormation兼容性：此属性将直接传递给`DefinitionS3Location`的财产`AWS::StepFunctions::StateMachine`资源。

`Events`

指定触发此状态机的事件。事件由一种类型和一组取决于类型的属性组成。

类型：EventSource (p. 162)

必需：否

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

`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：此属性将直接传递给`LoggingConfiguration`的财产`AWS::StepFunctions::StateMachine`资源。

`Name`

状态机的名称。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`StateMachineName`的财产`AWS::StepFunctions::StateMachine`资源。

`PermissionsBoundary`

要用于此状态机执行角色的权限边界的 ARN。只有在为您生成角色时，此属性才有效。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`PermissionsBoundary`的财产`AWS::IAM::Role`资源。

`Policies`

此状态机的执行角色需要的一个或多个策略。

此属性接受单个字符串或字符串列表。该属性可以是Amazon管理Amazon Identity and Access Management(IAM) 策略，Amazon SAM策略模板，或者格式化为地图的一个或多个内联策略文档。

你可以提供`Role`要么`Policies`.

如果`Role`属性已设置，将忽略该属性。

类型：字符串 | 列表 | 地图

必需：条件

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

`Role`

用作此状态机执行角色的 IAM 角色的 ARN。

您必须提供`Role`要么`Policies`.

类型：字符串

必需：条件

Amazon CloudFormation兼容性：此属性将直接传递给`RoleArn`的财产`AWS::StepFunctions::StateMachine`资源。

`Tags`

字符串到字符串映射，该映射指定添加到状态机的标签以及相应的执行角色。有关标签的有效键词和值的信息，请参阅标签的财产AWS::StepFunctions::StateMachine资源。

类型：Map

必需：否

Amazon CloudFormation兼容性：此属性类似于`Tags`的财产`AWS::StepFunctions::StateMachine`资源。Amazon SAM会自动添加`stateMachine:createdBy:SAM`标签到此资源以及为其生成的默认角色。

`Tracing`

选择是否Amazon X-Ray已为状态机启用。有关将 X-Ray 结合使用的更多信息，请参阅。Amazon X-Ray和 Step Functions中的Amazon Step Functions开发人员指南.

类型：TracingConfiguration

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给TracingConfiguration的财产AWS::StepFunctions::StateMachine资源。

Type

状态机的类型。

有效值：STANDARD 或 EXPRESS

类型：字符串

必需：否

默认值：STANDARD

Amazon CloudFormation兼容性：此属性将直接传递给StateMachineType的财产AWS::StepFunctions::StateMachine资源。

返回值

Ref

当向 Ref 内部函数提供此资源的逻辑 ID 时，Ref 将返回底层证券的 Amazon 资源名称 (ARN)。AWS::StepFunctions::StateMachine资源。

有关如何使用的更多信息Ref函数，请参阅Ref中的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中的Amazon CloudFormation用户指南.

Name

返回状态机的名称，如HelloWorld-StateMachine.

示例

状态机定义文件

以下是使用定义文件定义的状态机的示例。这些区域有：my_state_machine.asl.json必须写入文件Amazon States Language.

在此示例中，DefinitionSubstitution条目允许状态机包含在Amazon SAM模板文件。

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
```

内联状态机定义

以下是内联状态机定义的示例。

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，请写下Amazon SAMJSON 中的模板文件。

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。每个事件都由一个类型和一组依赖于该类型的属性组成。有关每个事件源的属性的更多信息，请参阅与该类型对应的副主题。

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

```
  Properties: Schedule (p. 173) | CloudWatchEvent (p. 168) | EventBridgeRule (p. 170)
 | Api (p. 163)
  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`Type`.

类型：Schedule (p. 173)|CloudWatch 事件 (p. 168)|EventBridgerRule (p. 170)|API (p. 163)

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

`Type`

事件类型。

有效值：`Api`、`Schedule`、`CloudWatchEvent`、`EventBridgeRule`

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

示例

API

以下是事件的示例。API。

YAML

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

Api

描述Api事件源类型。如果AWS::Serverless::Api (p. 32)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。

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

```
  Auth: ApiStateMachineAuth (p. 164)
  Method: String
  Path: String
  RestApiId: String
```

属性

`Auth`

此 API、路径和方法的授权配置。

使用此属性覆盖 APIDefaultAuthorizer设置单个路径，如果没有DefaultAuthorizer已指定，或覆盖默认设置ApiKeyRequired设置。

类型：apistateMachineAuth (p. 164)

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

`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。该值必须以开头`/`.

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

`RestApiId`

对的标识符`RestApi`资源，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AWS::Serverless::Api (p. 32)在此模板中定义的资源。

如果不定义此属性，Amazon SAM创建默认设置AWS::Serverless::Api (p. 32)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`.

此属性无法引用AWS::Serverless::Api (p. 32)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而且没有Amazon CloudFormation等效项

示例

APEvent

下面是一个事件的示例：`Api`。

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. 166)
```

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. 166)

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.

语法

要在您的Amazon Serverless Application Model(Amazon SAM ）模板，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`

这些区域有：Amazon要阻止的账户。

类型：List

必需：否

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

`AwsAccountWhitelist`

这些区域有：Amazon允许的账户。有关此属性的示例，请参阅此页面底部的示例部分。

类型：List

必需：否

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

`CustomStatements`

要应用于此 API 的自定义资源策略声明的列表。有关此属性的示例，请参阅此页面底部的示例部分。

类型：List

必需：否

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

`IntrinsicVpcBlacklist`

要阻止的虚拟私有云 (VPC) 列表，其中每个 VPC 都被指定为参考，例如动态参考或者`Ref` 内部函数. 有关此属性的示例，请参阅此页面底部的示例部分。

类型：List

必需：否

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

`IntrinsicVpcWhitelist`

允许的 VPC 列表，其中每个 VPC 都被指定为参考，例如动态参考或者`Ref` 内部函数.

类型：List

必需：否

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

`IntrinsicVpceBlacklist`

要阻止的 VPC 终端节点列表，其中每个 VPC 终端节点都被指定为参考，例如动态参考或者`Ref` 内部函数.

类型：List

必需：否

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

`IntrinsicVpceWhitelist`

要允许的 VPC 终端节点列表，其中每个 VPC 终端节点都被指定为参考，例如动态参考或者`Ref` 内部函数. 有关此属性的示例，请参阅此页面底部的示例部分。

类型：List

必需：否

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

`IpRangeBlacklist`

要阻止的 IP 地址或地址范围。有关此属性的示例，请参阅此页面底部的示例部分。

类型：List

必需：否

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

`IpRangeWhitelist`

允许的 IP 地址或地址范围。

类型：List

必需：否

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

`SourceVpcBlacklist`

要阻止的源 VPC 或 VPC 终端节点。源 VPC 名称必须以`"vpc-"`并且源 VPC 终端节点名称必须以`"vpce-"`. 有关此属性的示例，请参阅此页面底部的示例部分。

类型：List

必需：否

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

`SourceVpcWhitelist`

要允许的源 VPC 或 VPC 终端节点。源 VPC 名称必须以`"vpc-"`并且源 VPC 终端节点名称必须以`"vpce-"`.

类型：List

必需：否

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

示例

资源策略示例

以下示例阻止两个 IP 地址和一个源 VPC，并允许Amazonaccount.

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。

Amazon Serverless Application Model(Amazon SAM) 将生成AWS::Events::Rule设置此事件类型时的资源。

重要提示：EventBridgeRule (p. 170)是要使用的首选事件源类型，而不是`CloudWatchEvent`.`EventBridgeRule`和`CloudWatchEvent`使用相同的底层服务、API 和Amazon CloudFormation资源的费用。但是，Amazon SAM将仅将对新功能的支持添加到`EventBridgeRule`.

语法

要在您的中声明该实体Amazon Serverless Application Model(Amazon SAM) 模板，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。如果您忽略此属性，Amazon SAM使用默认事件总线。

类型：字符串

必需：否

默认值：默认事件总线

Amazon CloudFormation兼容性：此属性将直接传递给`EventBusName`的财产`AWS::Events::Rule`资源。

`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：此属性将直接传递给`Input`的财产`AWS::Events::Rule Target`资源。

`InputPath`

当您不希望将整个匹配的事件传递到目标时，请使用`InputPath`属性来描述要通过的事件的哪一部分。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`InputPath`的财产`AWS::Events::Rule Target`资源。

`Pattern`

描述哪些事件路由到指定目标。有关更多信息，请参阅 。EventBridge 中的事件和事件模式中的Amazon EventBridge 用户指南.

类型：EventPattern

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给`EventPattern`的财产`AWS::Events::Rule`资源。

示例

CloudWatch 事件

以下是的示例`CloudWatchEvent`事件源类型。

YAML

```
CWEvent:
  Type: CloudWatchEvent
  Properties:
    Input: '{"Key": "Value"}'
    Pattern:
      detail:
        state:
```

```
        - terminated
```

EventBridgeRule

描述EventBridgeRule事件源类型，它将状态机设置为 Amazon EventBridge 规则的目标。有关更多信息，请参阅 。什么是 Amazon EventBridge？中的Amazon EventBridge 用户指南.

Amazon SAM生成AWS::Events::Rule设置此事件类型时的资源。

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

```
DeadLetterConfig: DeadLetterConfig (p. 171)
EventBusName: String
Input: String
InputPath: String
Pattern: EventPattern
RetryPolicy: RetryPolicy
```

属性

`DeadLetterConfig`

配置 Amazon Simple Queue Service (Amazon SQS) 队列，其中 EventBridge 将事件发送在失败目标调用后发送事件。例如，在向不存在的 Lambda 函数发送事件时，或者当 EventBridge 没有足够的权限调用 Lambda 函数时，调用可能会失败。有关更多信息，请参阅 。事件重试策略和使用死信队列中的Amazon EventBridge 用户指南.

类型：DeadLetterConfig (p. 171)

必需：否

Amazon CloudFormation兼容性：此属性类似于DeadLetterConfig的财产AWS::Events::RuleTarget数据类型。这些区域有：Amazon SAM此属性的版本包括额外的子属性，以防你想Amazon SAM为您创建死信队列。

`EventBusName`

要与该规则关联的事件总线。如果您忽略此属性，Amazon SAM使用默认事件总线。

类型：字符串

必需：否

默认值：默认事件总线

Amazon CloudFormation兼容性：此属性将直接传递给EventBusName一个的财产AWS::Events::Rule资源。

`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：此属性将直接传递给Input一个的财产AWS::Events::RuleTarget资源。

`InputPath`

当您不希望将整个匹配的事件传递到目标时，请使用`InputPath`属性来描述要通过的事件的哪一部分。

类型：字符串

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`InputPath`一个的财产`AWS::Events::Rule Target`资源。

`Pattern`

描述哪些事件路由到指定目标。有关更多信息，请参阅 。EventBridge 中的事件和事件模式中的Amazon EventBridge 用户指南.

类型：EventPattern

必需：是

Amazon CloudFormation兼容性：此属性将直接传递给`EventPattern`一个的财产`AWS::Events::Rule`资源。

`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 。事件重试策略和使用死信队列中的Amazon EventBridge 用户指南.

类型：RetryPolicy

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给`RetryPolicy`的财产`AWS::Events::Rule Target`数据类型。

示例

EventBridgerRule

以下是的示例。`EventBridgeRule`事件源类型。

YAML

```
EBRule:
  Type: EventBridgeRule
  Properties:
    Input: '{"Key": "Value"}'
    Pattern:
      detail:
        state:
          - terminated
```

DeadLetterConfig

对象用于指定 Amazon Simple Queue Service (Amazon SQS) 队列的对象，在目标调用失败后，EventBridge 将事件发送。例如，在向不存在的状态机发送事件或者调用状态机的权限不足时，调用可能会失败。有关更多信息，请参阅 。事件重试策略和使用死信队列中的Amazon EventBridge 用户指南.

语法

要在您的Amazon Serverless Application Model(Amazon SAM) 模板，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 Amazon 资源名称 (ARN)。

注意：指定指定`Type`财产或`Arn`财产，但不能同时提供两者。

类型：字符串

必需：否

Amazon CloudFormation兼容性：该属性将直接传递给`Arn`的财产`AWS::Events::Rule DeadLetterConfig`数据类型。

`QueueLogicalId`

死信队列的自定义名称Amazon SAM创建如果`Type`已指定。

注意：如果`Type`未设置属性，将忽略该属性。

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

`Type`

队列的类型。设置此属性后，Amazon SAM自动创建死信队列并附加必要的基于资源的策略授予规则资源向队列发送事件的权限。

注意：指定指定`Type`财产或`Arn`财产，但不能同时提供两者。

有效值：`SQS`

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

DeadLetterConfig

DeadLetterConfig

YAML

```
DeadLetterConfig:
  Type: SQS
```

```
  QueueLogicalId: MyDLQ
```

Schedule

描述Schedule事件源类型，它将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 。什么是 Amazon EventBridge？中的Amazon EventBridge 用户指南.

Amazon Serverless Application Model(Amazon SAM) 会生成AWS::Events::Rule设置此事件类型时的资源。

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

```
  DeadLetterConfig: DeadLetterConfig (p. 175)
  Description: String
  Enabled: Boolean
  Input: String
  Name: String
  RetryPolicy: RetryPolicy
  Schedule: String
```

属性

`DeadLetterConfig`

在目标调用失败后，配置 Amazon Simple Queue Service (Amazon SQS) 队列，其中 EventBridge 将发送事件。例如，在向不存在的 Lambda 函数发送事件时，或者当 EventBridge 没有足够的权限调用 Lambda 函数时，调用可能会失败。有关更多信息，请参阅 。事件重试策略和使用死信队列中的Amazon EventBridge 用户指南.

类型：DeadLetterConfig (p. 175)

必需：否

Amazon CloudFormation兼容性：此属性类似于DeadLetterConfig的财产AWS::Events::Rule Target数据类型。这些区域有：Amazon SAM此属性的版本包括额外的子属性，以防你想Amazon SAM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：此属性将直接传递给Description一个的财产AWS::Events::Rule资源。

`Enabled`

指示是否启用规则。

要禁用规则，请将此属性设置为false.

类型：Boolean

必需：否

Amazon CloudFormation兼容性：此属性类似于State一个的财产AWS::Events::Rule资源。如果此属性设置为true然后Amazon SAM通行证ENABLED，否则它会通过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：此属性将直接传递给Target一个的财产AWS::Events::Rule Target资源。

`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：此属性将直接传递给Name一个的财产AWS::Events::Rule资源。

`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 。事件重试策略和使用死信队列中的Amazon EventBridge 用户指南.

类型：RetryPolicy

必需：否

Amazon CloudFormation兼容性：此属性将直接传递给RetryPolicy的财产AWS::Events::Rule Target数据类型。

`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：此属性将直接传递给ScheduleExpression一个的财产AWS::Events::Rule资源。

示例

CloudWatch Events

CloudWatch Events Event Event

YAML

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

```
    Enabled: false
```

DeadLetterConfig

用于指定 Amazon Simple Queue Service (Amazon SQS) 队列的对象，在目标调用失败后，EventBridge 将事件发送。例如，在向不存在的状态机发送事件或者调用状态机的权限不足时，调用可能会失败。有关更多信息，请参阅 。事件重试策略和使用死信队列中的Amazon EventBridge 用户指南.

语法

要在您的Amazon Serverless Application Model(Amazon SAM) 模板，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 Amazon 资源名称 (ARN)。

注意：指定`Type`财产或`Arn`房地产，但不是两者。

类型：字符串

必需：否

Amazon CloudFormation兼容性：该属性将直接传递给`Arn`的财产`AWS::Events::Rule DeadLetterConfig`数据类型。

`QueueLogicalId`

死信队列的自定义名称Amazon SAM创建如果`Type`已指定。

注意：如果`Type`未设置属性，将忽略该属性。

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

`Type`

队列的类型。设置此属性后，Amazon SAM自动创建死信队列并附加必要的基于资源的策略授予规则资源向队列发送事件的权限。

注意：指定`Type`财产或`Arn`房地产，但不是两者。

有效值：`SQS`

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

DeadLetterConfig

DeadLetterConfig

YAML

```
DeadLetterConfig:
  Type: SQS
  QueueLogicalId: MyDLQ
```

有关所有的参考信息Amazon支持的资源和属性类型Amazon CloudFormation和Amazon SAM，请参阅Amazon资源和属性类型参考中的Amazon CloudFormation用户指南.

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。生成Amazon CloudFormation资源 (p. 177).

下表汇总了资源属性支持的方式。Amazon SAM，取决于异常 (p. 176)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Amazon CloudFormation资源方案 (p. 178).
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Amazon CloudFormation资源方案 (p. 178).

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. 211).
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。构建层 (p. 209).

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.

生成Amazon CloudFormation资源

何时Amazon Serverless Application Model(Amazon SAM) 处理你的Amazon SAM模板文件，它会生成一个或多个Amazon CloudFormation资源的费用。这套Amazon CloudFormation那些资源Amazon SAM根据您指定的方案，生成的不同。一个方案是的组合Amazon SAM模板文件中指定的资源和属性。你可以引用生成的Amazon CloudFormation模板文件中其他地方的资源，类似于引用模板文件中明确声明的资源的方式。

例如，如果您指定`AWS::Serverless::Function`您的资源Amazon SAM模板文件，Amazon SAM始终生成`AWS::Lambda::Function`基本资源。如果您还指定可选`AutoPublishAlias`财产，Amazon SAM此外生成`AWS::Lambda::Alias`和`AWS::Lambda::Version`资源的费用。

本部分列出了方案和Amazon CloudFormation他们生成的资源，并显示如何引用生成的Amazon CloudFormation您的资源Amazon SAM模板文件。

生成引用Amazon CloudFormation资源

生成了引用您有两个选项Amazon CloudFormation你的资源Amazon SAM模板文件，`LogicalId`或者通过可参考的财产。

生成引用Amazon CloudFormation按 LogicalID 分类的资源

这些区域有：Amazon CloudFormation那些资源Amazon SAM生成每个都有`LogicalId`，它是模板文件中唯一的字母数字（A-Z、a-z、0-9）标识符。Amazon SAM使用`LogicalIds`的Amazon SAM模板文件中的资源以构造`LogicalIds`的Amazon CloudFormation它产生的资源。您可以使用`LogicalId`生成的Amazon CloudFormation资源来访问模板文件中该资源的属性，就像在Amazon CloudFormation你明确声明的资源。有关 的更多信息`LogicalIds`在Amazon CloudFormation和Amazon SAM模板，请参阅资源中的Amazon CloudFormation用户指南.

Note

这些区域有：LogicalIds的一些生成的资源包含唯一的哈希值，以避免命名空间冲突。这些区域有：LogicalIds的这些资源是在创建堆栈时派生的。您只能在创建堆栈后使用Amazon Web Services Management Console、Amazon CLI，或者其中一个Amazon开发工具包。我们建议您不要通过引用这些资源LogicalId因为哈希值可能会改变。

生成引用Amazon CloudFormation按可参考属性划分的资源

对于一些生成的资源，Amazon SAM提供的可参考属性Amazon SAM资源。您可以使用此属性来引用生成的Amazon CloudFormation资源及其在你的属性Amazon SAM模板文件。

Note

不是全部生成Amazon CloudFormation资源具有可参考的属性。对于这些资源，您必须使用LogicalId.

生成Amazon CloudFormation资源方案

下表汇总了Amazon SAM构成生成场景的资源和属性Amazon CloudFormation资源的费用。中的主题方案列提供有关附加的详细信息Amazon CloudFormation那些资源Amazon SAM为该场景生成。

Amazon SAM 资源	BaseAmazon CloudFormation 资源	方案
AWS::Serverless::Api	AWS::ApiGateway::RestApi	• 指定了 DomainName 属性 (p. 180) • 指定了 UsagePlan 属性 (p. 180)
Amazon# # #### # (p. 180)	AWS::CloudFormation::Stack	• 除了生成基础Amazon CloudFormation资源，此无服务器资源没有其他方案。
AWS::Serverless::Function	AWS::Lambda::Function	• 指定了 AutoPublishAlias 属性 (p. 181) • 未指定角色属性 (p. 181) • 指定了 DeploymentPreference 属性 (p. 182) • 指定了 Api 事件源 (p. 182) • 指定了 HTTPAPI 事件源 (p. 182) • 指定了流媒体事件源 (p. 182) • 指定了事件桥（或事件总线）事件源 (p. 183) • 指定了 iotRule 事件源 (p. 183) • 为 Amazon SNS 事件指定了 onSuccess（或 onFailure）属性 (p. 183) • 为 Amazon SQS 事件指定了 onSuccess（或 onFailure）属性 (p. 183)
AWS::Serverless::HttpApi	AWS::ApiGatewayV2::Api	• 指定了 StageName 属性 (p. 184) • StageName 属性是不指定的 (p. 184) • 指定了 DomainName 属性 (p. 185)
AWS::Serverless::LayerVersion	AWS::Lambda::LayerVersion	• 除了生成基础Amazon CloudFormation资源，此无服务器资源没有其他方案。
AWS::Serverless::SimpleTable	AWS::DynamoDB::Table	• 除了生成基础Amazon CloudFormation资源，此无服务器资源没有其他方案。

Amazon SAM 资源	BaseAmazon CloudFormation 资源	方案
AWS::Serverless::StateMachine	AWS::StepFunctions::StateMachine	• 未指定角色属性 (p. 186) • 指定了 Api 事件源 (p. 186) • 指定了事件桥（或事件总线）事件源 (p. 186)

主题

- Amazon CloudFormation指定 Amazon# Serverless# Api 时生成的资源 (p. 179)
- Amazon CloudFormation指定 Amazon# Serverless# 应用程序时生成的资源 (p. 180)
- Amazon CloudFormation生成的资源时间AWS::Serverless::Function已指定 (p. 180)
- Amazon CloudFormation指定 Amazon# Serverless# HTTPAPI 时生成的资源 (p. 184)
- Amazon CloudFormation当时生成的资源AWS::Serverless::LayerVersion已指定 (p. 185)
- Amazon CloudFormation当时生成的资源AWS::Serverless::SimpleTable已指定 (p. 185)
- Amazon CloudFormation生成的资源AWS::Serverless::StateMachine已指定 (p. 185)

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. 180)
- 指定了 UsagePlan 属性 (p. 180)

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

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. 184)
- StageName 属性是不指定的 (p. 184)
- 指定了 DomainName 属性 (p. 185)

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

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

当你使用创作无服务器应用程序时Amazon SAM，你构造一个Amazon SAM模板来声明和配置应用程序的组件。

本节包含有关验证您的主题Amazon SAM模板并使用依赖关系构建应用程序。它还包含有关使用Amazon SAM对于某些使用案例，例如使用 Lambda 层、使用嵌套应用程序、控制对 API Gateway API 的访问、编排Amazon具有 Step Functions 的资源以及对应用程序进行代码签

主题

正在验证Amazon SAM模板文件

使用验证模板`sam validate (p. 281)`. 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，请参阅构建层 (p. 209).

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. 67)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. 64)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 部署无服务器程序 (p. 221)。

> 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. 194).

- 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. 196)。
- 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. 197)。
- 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. 198).
- 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. 198).
- 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. 199)。

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. 194)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. 199).

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。

Lambda`TOKEN`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. 41)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. 41)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. 139)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. 41)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. 41)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. 41)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. 41)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. 139)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.

编排Amazon资源Amazon Step Functions

您可以使用Amazon Step Functions编排Amazon Lambda函数和其他Amazon资源可形成复杂而强大的工作流程。

> Note
>
> 要管理Amazon SAM包含 Step Functions 状态机的模板，您必须使用Amazon SAMCLI。要检查您具有的版本，`sam --version`.

Step Functions 基于任务和状态机. 您使用基于 JSON 的定义状态机Amazon States Language. 这些区域有：Step Functions显示状态机结构的图形视图，这样您可以直观检查状态机逻辑和监控执行情况。

在中使用 Step Functions 支持Amazon Serverless Application Model(Amazon SAM ），您可以执行以下操作：

- 直接在Amazon SAM模板或在单独的文件中
- 通过创建状态机执行角色Amazon SAM策略模板、内联策略或托管策略
- 使用 API Gateway 或 Amazon EventBridge 事件，按时间表触发状态机执行Amazon SAM模板，或者直接调用 API
- 使用可用Amazon SAM策略模板对于常见的 Step Functions 开发模式。

示例

以下示例代码段Amazon SAM模板文件在定义文件中定义 Step Functions 状态机。请注意，`my_state_machine.asl.json`必须写入文件Amazon States Language.

```
AWSTemplateFormatVersion: "2010-09-09"
```

```
Transform: AWS::Serverless-2016-10-31
Description: Sample SAM template with Step Functions State Machine

Resources:
  MyStateMachine:
    Type: AWS::Serverless::StateMachine
    Properties:
      DefinitionUri: statemachine/my_state_machine.asl.json
      ...
```

下载样本Amazon SAM包含 Step Functions 状态机的应用程序，请参阅使用创建 Step Functions 状态机 Amazon SAM中的Amazon Step Functions开发人员指南.

更多信息

要了解有关 Step Functions 并将其与结合使用Amazon SAM，请参阅：

- 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规范：AWS::Serverless::StateMachine (p. 158)

为 配置代码签名Amazon SAM应用程序

您可以使用Amazon SAM以启用无服务器应用程序的代码签名，以帮助确保只部署受信任的代码。有关代码签名功能的更多信息，请参阅为 Lambda 函数配置代码签名中的Amazon Lambda开发人员指南.

为无服务器应用程序配置代码签名之前，必须使用创建签名配置文件。Amazon签名者。您可以将此签名配置文件用于以下任务：

1. 创建代码签名配置— 声明`AWS::Lambda::CodeSigningConfig`资源，以指定受信任发布者的签名配置文件以及为验证检查设置策略操作。你可以在同一个中声明这个对象Amazon SAM模板作为你的无服务器功能，在不同的Amazon SAM模板，或者在Amazon CloudFormation。模板。然后，您可以通过指定`CodeSigningConfigArn`属性具有 Amazon 资源名称 (ARN) 的函数`AWS::Lambda::CodeSigningConfig`资源。
2. 为代码签名— 使用`sam package`要么`sam deploy`命令使用`--signing-profiles`选项。

   Note

   为了成功使用`sam package`要么`sam deploy`命令，必须为与这些命令配合使用的 Amazon S3 存储桶启用版本控制。如果您使用 Amazon S3 存储桶，Amazon SAM为你创建，版本控制将自动启用。有关 Amazon S3 存储桶版本控制的更多信息以及在 Amazon S3 存储桶上启用版本控制的说明，请参阅在 Amazon S3 存储桶中使用版本控制中的Amazon Simple Storage Service 用户指南.

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，请参阅签名验证中的Amazon Lambda开发人员指南.

示例

创建签名配置文件

要创建签名配置文件，请运行以下命令：

```
aws signer put-signing-profile --platform-id "AWSLambda-SHA384-ECDSA" --profile-
name MySigningProfile
```

如果上一个命令成功，则会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"
}
```

这些区域有：`profileVersionArn`字段包含创建代码签名配置时要使用的 ARN。

为函数创建代码签名配置并启用代码签名

以下示例Amazon SAM模板声明`AWS::Lambda::CodeSigningConfig`资源并为 Lambda 函数启用代码签名。在此示例中，有一个受信任的配置文件，如果签名检查失败，则会拒绝部署。

```
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"
```

为代码签名

您可以在打包或部署应用程序时签署代码。指定`--signing-profiles`选项可以使用`sam package`要么`sam deploy`命令，如以下示例命令所示。

打包应用程序时签署函数代码：

```
sam package --signing-profiles HelloWorld=MySigningProfile --s3-bucket test-bucket --
output-template-file packaged.yaml
```

在打包应用程序时，同时签署函数代码和函数所依赖的层：

```
sam package --signing-profiles HelloWorld=MySigningProfile MyLayer=MySigningProfile --s3-
bucket test-bucket --output-template-file packaged.yaml
```

签署函数代码和图层，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

为了成功使用sam package要么sam deploy命令，必须为与这些命令配合使用的 Amazon S3 存储桶启用版本控制。如果您使用 Amazon S3 存储桶，Amazon SAM为你创建，版本控制将自动启用。有关 Amazon S3 存储桶版本控制的更多信息以及在 Amazon S3 存储桶上启用版本控制的说明，请参阅在 Amazon S3 存储桶中使用版本控制中的Amazon Simple Storage Service 用户指南.

提供签名配置文件sam deploy --guided

当你运行sam deploy --guided命令配置了代码签名的无服务器应用程序，Amazon SAM提示您提供用于代码签名的签名配置文件。有关 的更多信息sam deploy --guided提示，请参阅sam deploy (p. 257)中的Amazon SAMCLI 命令参考。

构建无服务器应用程

构建无服务器应用程序涉及将Amazon SAM模板文件、应用程序代码以及任何适用的特定于语言的文件和依赖关系，并将所有构建项目放置为适当的格式和位置，以便在工作流程中的后续步骤使用。

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 SAMCLI。这两个活动都使用应用程序的构建工件作为输入。

本节介绍如何使用sam build (p. 252)命令来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函数和层，或应用程序的单个组件，例如特定函数或层。

主题

构建应用程序

要构建您的无服务器应用程序，请使用sam build (p. 252)命令。此命令还会收集应用程序依赖项的构建构件，并将它们放置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。

您可以在清单文件中指定应用程序的依赖关系，例如requirements.txt(Python)或package.json(Node.js)，或者使用Layers函数资源的属性。这些区域有：Layers属性包含以下列表Amazon Lambda层Lambda 函数所依赖的资源。

应用程序的构建构件的格式取决于每个函数的PackageType属性。此属性的选项是：

- **Zip**— .zip 文件存档，其中包含应用程序代码及其依赖项。如果要将代码打包为 .zip 文件存档，则必须为函数打包了 Lambda 运行时。
- **Image**— 容器镜像，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 程序包类型的更多信息，请参阅Lambda 部署程序包在Amazon Lambda开发人员指南.

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，请声明PackageType: Zip用于您的无服务器功能。

Amazon SAM为... 构建应用程序建筑 (p. 68)您指定的值。如果不指定架构，Amazon SAM使用x86_64默认情况下。

如果您的 Lambda 函数依赖于具有本机编译程序的软件包，请使用--use-containerFlags。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，因此当您将它们部署到AmazonCloud。

当您使用以下应用程序时：--use-container选项，默认Amazon SAM从中提取容器镜像Amazon ECR Public. 例如，如果你想从另一个存储库提取容器镜像 DockerHub，您可以使用--build-image选项并提供备用容器镜像的 URI。以下是使用容器镜像构建应用程序的两个示例命令 DockerHub 存储库：

```
# 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
```

要获取 URI 列表，您可以使用`--build-image`，请参阅映像存储库 (p. 325)其中包含 DockerHub 许多支持的运行时的 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

要将无服务器应用程序构建为容器镜像，请声明`PackageType: Image`用于您的无服务器功能。你还必须声明`Metadata`包含以下条目的资源属性：

`Dockerfile`

与 Lambda 函数关联的 Docker File 的名称。

`DockerContext`

Dockerfile 的位置。

`DockerTag`

（可选）应用于构建镜像的标签。

`DockerBuildArgs`

为编译生成参数。

以下是示例`Metadata`资源属性部分：

```
    Metadata:
      Dockerfile: Dockerfile
      DockerContext: ./hello_world
      DockerTag: v1
```

要下载配置了的示例应用程序，请执行以下操作`Image`包裹类型，请参阅第 1 步：下载示例Amazon SAM应用程序 (p. 17)在教程：部署 Hello World 应用程序. 在询问要安装哪种软件包类型的提示下，选择`Image`.

Note

如果您在 Dockerfile 中指定了多架构基础镜像，Amazon SAM为主机的架构构建容器镜像。要针对不同的架构进行构建，请指定使用特定目标架构的基础映像。

容器环境变量文件

要为构建容器提供包含环境变量的 JSON 文件，请使用`--container-env-var-file`与之argumentsam `build`命令。您可以提供适用于所有无服务器资源的单个环境变量，也可以为每种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`Parameters`对象如下所示：

```
{
  "Parameters": {
    "GITHUB_TOKEN": "TOKEN_GLOBAL"
```

```
  }
}
```

要为每种资源提供不同的环境变量，请为每个资源指定对象，如下所示：

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

将环境变量保存为一个文件，例如，名为`env.json`. 以下命令使用此文件将您的环境变量传递给构建容器：

```
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`命令构建 .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 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，你可以使用`npm ci`而不是`npm install`安装依赖项。要使用`npm ci`，请指定`UseNpmCi: True`下`BuildProperties`在您的 Lambda 函数中`Metadata`资源属性。要使用`npm ci`，则应用程序必须具有`package-lock.json`要么`npm-shrinkwrap.json`文件存在于`CodeUri`用于您的 Lambda 函数。

以下示例使用了`npm ci`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
```

使用 esbuild 构建 Node.js Lambda 函数

编译和打包 Node.jsAmazon Lambda函数，您可以使用Amazon SAM使用 esbuild 的 CLI JavaScript 捆绑商。esbuild 打包器支持您在其中编写的 Lambda 函数 TypeScript.

要使用 esbuild 构建 Node.js Lambda 函数，请添加一个`Metadata`反对你`AWS:Serverless::Function`资源并指定`esbuild`对于`BuildMethod`. 当你运行sam build命令，Amazon SAM使用 esbuild 捆绑您的 Lambda 函数代码。

元数据属性

这些区域有：`Metadata`对象支持 esbuild 的以下属性：

BuildMethod

为您的应用程序指定打包程序。`esbuild` 是唯一受支持的值。

BuildProperties

为您的 Lambda 函数代码指定构建属性。

这些区域有：`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

指定要在编译中省略的软件包列表。

加载程序

指定用于加载给定文件类型数据的配置列表。

MainFields

指定哪个`package.json`解析程序包时要尝试导入的字段。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`。

源地图

指定打包器是否生成源映射文件。默认值为 `false`。

当设置为`true`,`NODE_OPTIONS: --enable-source-maps`将附加到 Lambda 函数的环境变量中，生成源映射并将其包含在函数中。

或者，当`NODE_OPTIONS: --enable-source-maps`包含在函数的环境变量中，`Sourcemap`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 会对所有静态环境变量进行加密Amazon Key Management Service(Amazon KMS)。使用源映射时，要成功部署，函数的执行角色必须具有执行`kms:Encrypt`操作。

目标

指定目标 ECMAScript 版本。默认值为 `es2020`。

TypeScript Lambda 函数示例

下面的示例Amazon SAM模板代码段使用 esbuild 创建一个 Node.js Lambda 函数 TypeScript 编码输入`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:
        Minify: false
        Target: "es2020"
        Sourcemap: true
        EntryPoints:
          - app.ts
```

构建层

您可以使用Amazon SAM以构建自定义层。有关层的信息，请参阅AmazonLambda 层中的Amazon Lambda开发人员指南.

要构建自定义图层，请在Amazon Serverless Application Model(Amazon SAM) 模板文件并包含`Metadata`资源属性部分带`BuildMethod`条目。的有效值`BuildMethod`是的标识符Amazon Lambda运行时间，或者`makefile`. 包含`BuildArchitecture`条目以指定您的层支持的指令集体系结构。的有效值`BuildArchitecture`是Lambda 指令集架构.

如果你指定`makefile`中，提供自定义 makefile，您可以在其中声明表单的构建目标`build-`*`layer-logical-id`*其中包含图层的构建命令。如有必要，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适当位置。makefile 的位置由`ContentUri`图层资源的属性，且必须命名`Makefile`.

> **Note**
>
> 在创建自定义层时，Amazon Lambda取决于环境变量来查找图层代码。Lambda 运行时包括`/opt`将图层代码复制到的目录。项目的构建工件文件夹结构必须与运行时的预期文件夹结构匹配，以便能够找到自定义层代码。
> 例如，对于 Python，您可以将代码放在`python/`子目录。对于 NodeJS，你可以将代码放在`nodejs/node_modules/`子目录。
> 有关更多信息，请参阅 。在层中包括库依赖项中的Amazon Lambda开发人员指南.

以下是示例：`Metadata`资源属性部分。

```
Metadata:
  BuildMethod: python3.8
  BuildArchitecture: arm64
```

> **Note**
>
> 如果您不包含`Metadata`资源属性部分，Amazon SAM不构建图层。相反，它会从中指定的位置复制构建工件`CodeUri`层资源的属性。有关更多信息，请参阅 。内容 URI (p. 153)的财产`AWS::Serverless::LayerVersion`资源类型。

当您包含`Metadata`资源属性部分，您可以使用sam build (p. 252)命令来构建图层，既可以作为独立对象，也可以作为依赖项构建图层Amazon Lambdafunction.

- 作为一个独立的对象。您可能只想构建层对象，例如，当您在本地测试对图层的代码更改而不需要构建整个应用程序时。要独立构建图层，请使用`sam build layer-logical-id`命令。
- 作为 Lambda 函数的依赖项。当您将图层的逻辑 ID 包含在`Layers`相同 Lambda 函数的属性Amazon SAM模板文件，该层是该 Lambda 函数的依赖关系。当该图层还包含`Metadata`资源属性部分带`BuildMethod`条目中，您可以通过使用`sam build`命令或者使用指定函数资源`sam build function-logical-id`命令。

示例

模板示例 1：针对 Python 3.6 运行时环境构建图层

以下示例Amazon SAM模板根据 Python 3.6 运行时环境构建图层。

```
Resources:
  MyLayer:
    Type: AWS::Serverless::LayerVersion
    Properties:
      ContentUri: my_layer
      CompatibleRuntimes:
        - python3.6
    Metadata:
      BuildMethod: python3.6              # 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`makefile`包含构建目标和将要执行的命令。请注意，`ContentUri`属性设置为`my_layer`，所以 makefile 必须位于`my_layer`子目录，文件名必须为`Makefile`. 另请注意，构建工件被复制到`python/`子目录以便Amazon Lambda将能够找到层代码。

```
build-MyLayer:
    mkdir -p "$(ARTIFACTS_DIR)/python"
    cp *.py "$(ARTIFACTS_DIR)/python"
    python -m pip install -r requirements.txt -t "$(ARTIFACTS_DIR)/python"
```

例子 sam 构建命令

以下`sam build`命令构建包含`Metadata`资源属性部分。

```
# 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sam build (p. 252)命令来构建 Lambda 函数所需的自定义运行时。您可以通过指定来声明 Lambda 函数使用自定义运行时Runtime: provided为了该函数。

要构建自定义运行时，请声明Metadata资源属性带BuildMethod: makefile条目。你提供了一个自定义 makefile，你可以在其中声明表单的构建目标build-*function-logical-id*其中包含运行时的构建命令。如有必要，makefile 负责编译自定义运行时，并将构建工件复制到工作流程后续步骤所需的适当位置。makefile 的位置由CodeUri函数资源的属性，且必须命名Makefile.

示例

示例 1：用 Rust 编写的函数的自定义运行时间

以下Amazon SAM模板声明了一个函数，该函数使用自定义运行时用于用 Rust 编写的 Lambda 函数，并指示sam build执行命令build-HelloRustFunction构建目标。

```
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 包含构建目标和将要执行的命令。请注意，CodeUri属性将设定为.，所以 makefile 必须位于项目根目录中（也就是说，与应用程序的目录相同的目录Amazon SAM模板文件)。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cargo build之前的命令makefile，请参阅Rust 运行时Amazon Lambda博客帖子。

示例 2：Python3.7 的 Makefile 构建器（替代使用捆绑构建器）

您可能想要使用捆绑构建器中未包含的库或模块。此示例显示了Amazon SAM使用 makefile 构建器的 Python3.7 运行时的模板。

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: hello_world/
      Handler: app.lambda_handler
      Runtime: python3.7
```

```
    Metadata:
      BuildMethod: makefile
```

以下 makefile 包含构建目标和将要执行的命令。请注意，`CodeUri`属性将设定为`hello_world`，所以 makefile 必须位于`hello_world`子目录，文件名必须为`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命令行接口 (CLI)，你可以在本地测试和 “逐步” 调试你的无服务器应用程序，然后再将你的应用程序上传到AmazonCloud。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。

当您在调试模式在本地调用 Lambda 函数时Amazon SAMCLI，然后你可以将调试器连接到它。使用调试器，您可以逐行浏览代码，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，layers 包将下载并缓存在本地主机上。有关更多信息，请参阅图层在本地缓存方式 (p. 189)。

主题

在本地调用 Lambda 函数

你可以调用你的Amazon Lambda使用在本地运行`sam #### (p. 265)` Amazon SAMCLI 命令并提供函数的逻辑 ID 和事件文件。或者，sam local invoke也接受`stdin`作为一项活动。有关事件的更多信息，请参阅Event在里面Amazon Lambda开发人员指南. 有关来自不同的事件消息格式的信息Amazon服务，请参阅使用Amazon Lambda将与其他服务一起使用在里面Amazon Lambda开发人员指南.

> Note
>
> 这些区域有：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。

您必须运行sam local invoke命令位于包含要调用的函数的项目目录中。

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`参数用于覆盖模板中定义的值。

声明环境变量

要声明适用于所有资源的全局环境变量，请指定`Parameters`对象如下所示：

```
{
    "Parameters": {
        "TABLE_NAME": "localtable",
        "BUCKET_NAME": "testBucket",
        "STAGE": "dev"
    }
}
```

要为每个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"
    }
}
```

在为每种资源指定对象时，可以使用以下标识符，这些标识符按从高到低的优先级顺序列出：

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，请使用`--env-vars`与之argumentinvoke要么start-api命令。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. 188).

在本地运行 API Gateway

要启动可用于测试 HTTP 请求/响应功能的 Amazon API Gateway 的本地实例，请使用`sam ## start-api (p. 267)` Amazon SAMCLI 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

Reload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Reload是指刷新整个应用程序而应用程序的状态丢失的时候。

你必须运行sam local start-api命令，该命令位于包含要调用的函数的项目目录中。

默认情况下，Amazon SAM使用Amazon Lambda代理集成，同时支持两者`HttpApi`和`Api`资源类型。有关代理集成的更多信息`HttpApi`资源类型，请参阅使用Amazon LambdaHTTP API 的代理集成在里面API Gateway 开发人员指南. 有关代理集成的更多信息`Api`资源类型，请参阅了解 API Gateway Lambda 代理集成在里面API Gateway 开发人员指南.

示例：

```
sam local start-api
```

Amazon SAM自动找到你中的任何函数Amazon SAM有以下内容的模板`HttpApi`要么`Api`事件源定义。然后，它将函数挂载到定义的 HTTP 路径上。

在下文中`Api`示例，`Ratings`功能挂载`ratings.py:handler()`在`/ratings`为了`GET`请求:

```
Ratings:
  Type: AWS::Serverless::Function
  Properties:
    Handler: ratings.handler
    Runtime: python3.6
    Events:
      Api:
        Type: Api
        Properties:
          Path: /ratings
          Method: get
```

以下是示例`Api`响应:

```
// 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`参数用于覆盖模板中定义的值。

声明环境变量

要声明适用于所有资源的全局环境变量，请指定`Parameters`对象如下所示：

```
{
    "Parameters": {
        "TABLE_NAME": "localtable",
```

```
        "BUCKET_NAME": "testBucket",
        "STAGE": "dev"
    }
}
```

要为每个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"
    }
}
```

在为每种资源指定对象时，可以使用以下标识符，这些标识符按从高到低的优先级顺序列出：

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，请使用`--env-vars`argumentinvoke要么start-api命令。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层问题的信息，请参阅使用图层 (p. 188).

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
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

IDE	运行时	Amazon Toolkit	逐步调试说明
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. 265)要么sam 本地 start-api (p. 267)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。

部署无服务器程序

Amazon SAM使用Amazon CloudFormation作为基本的部署机制。有关更多信息，请参阅 Amazon CloudFormation 用户指南的 什么是 Amazon CloudFormation？。部署无服务器应用程序的标准输入是使用sam build (p. 252)命令。有关 sam build 的更多信息，请参阅构建无服务器应用程 (p. 204)。

您可以使用手动部署应用程序。Amazon SAM命令行界面 (CLI) 命令。您还可以使用持续集成和持续部署 (CI/CD) 系统自动执行应用程序部署。您可以使用许多常用的CI/CD 系统进行部署Amazon SAM应用程序，包括Amazon CodePipeline、詹金斯、GitLab CI/CD, 和GitHub 操作.

使用 CI/CD 系统部署

Amazon SAM帮助组织为其首选 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、缩短变更的准备时间以及减少部署错误。

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。那些映像的Amazon SAM提供包括Amazon SAM用于许多受支持的 CLI 和构建工具Amazon Lambda运行时。这样，就可以更轻松地使用 Amazon SAMCLI。这些映像还减轻了团队为 CI/CD 系统创建和管理自己的映像的需要。有关 的更多信息 Amazon SAM构建容器镜像，请参阅映像存储库 (p. 325).

支持多个 CI/CD 系统Amazon SAM构建容器映像。你应该使用哪个 CI/CD 系统取决于几个因素。其中包括应用程序是使用单个运行时还是多次运行时，还是要在容器映像中构建应用程序，还是直接在主机上构建应用程序，无论是虚拟机 (VM) 还是裸机主机。

Amazon SAM还为封装的多个 CI/CD 系统提供了一组默认管道模板Amazon的部署最佳实践。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，并验证管道是否无法对基础架构进行意外更改。

您有两个主要选项可供使用。Amazon SAM部署无服务器应用程序：1) 修改现有管道配置以使用Amazon SAMCLI 命令，或 2) 生成 CI/CD 管道配置示例，您可以用作自己的应用程序的起点。

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. 222)
- 生成初学者 CI/CD 管道 (p. 225)

使用部署Amazon SAMCLI

在本地开发和测试无服务器应用程序之后，您可以使用sam deploy (p. 257)命令。

要拥有Amazon SAM通过提示指导您完成部署，请指定--guided旗子。当你指定此标志时，sam deploy命令 Zip 您的应用程序工件，将其上传到亚马逊 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，命令将您的应用程序部署到Amazon云。

示例：

```
# Deploy an application using prompts:
```

```
sam deploy --guided
```

使用对部署故障排除Amazon SAMCLI

Amazon SAMCLI 错误：“安全限制未满足”

运行时sam deploy --guided，系统会提示你问题`HelloWorldFunction may not have authorization defined, Is this okay? [y/N]`. 如果你回应此提示**N**（默认响应），您会看到以下错误：

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

该提示通知您，您即将部署的应用程序可能配置了未经授权的 Amazon API Gateway API。通过响应**N**对于这个提示，你说这不好。

要解决此问题，您可使用以下选项：

- 使用授权配置应用程序。有关配置授权的信息，请参阅控制对 API Gateway API 的访问 (p. 192).
- 用回答这个问题**Y**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

如果您想部署Amazon SAM应用程序逐渐而不是一次性，您可以指定部署配置Amazon CodeDeploy提供。有关更多信息，请参阅 。在 CodeDeploy 中使用部署配置中的Amazon CodeDeploy用户指南.

有关配置您的Amazon SAM应用程序要逐步部署，请参阅逐步部署无服务器应用程 (p. 327).

修改现有的CI/CD 管道

现有 CI/CD 管道使用部署无服务器应用程序的过程Amazon SAM略有不同，具体取决于您使用的 CI/CD 系统。

以下主题提供了将 CI/CD 系统配置为在Amazon SAM构建容器镜像：

主题

使用部署Amazon CodePipeline

配置您的Amazon CodePipeline管道来自动构建和部署Amazon SAM您的应用程序Amazon CloudFormation模板和`buildspec.yml`文件必须包含执行以下操作的行：

1. 从可用映像中使用必要的运行时间来引用构建容器映像。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命令行界面 (CLI) 命令。以下示例运行两个Amazon SAMCLI 命令：sam build和sam deploy（有必要的选择）。

此示例假定您已在中声明了所有函数和层。Amazon SAM带有模板文件`runtime: nodejs14.x`.

Amazon CloudFormation模板代码段：

```
  CodeBuildProject:
    Type: AWS::CodeBuild::Project
    Properties:
      Environment:
        ComputeType: BUILD_GENERAL1_SMALL
        Image: public.ecr.aws/sam/build-nodejs14.x
        Type: LINUX_CONTAINER
      ...
```

`buildspec.yml`代码段：

```
version: 0.2
phases:
  build:
    commands:
      - 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) 构建容器镜像的列表，请参阅映像存储库 (p. 325).

使用 Bitbucket 管道进行部署

配置您的Bitbucket 管道自动构建和部署Amazon SAM应用程序，您`bitbucket-pipelines.yml`文件必须包含执行以下操作的行：

1. 从可用映像中使用必要的运行时间来引用构建容器映像。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命令行界面 (CLI) 命令。以下示例运行两个Amazon SAMCLI 命令：sam build和sam deploy（有必要的选择）。

此示例假定您已在您中声明了所有函数和层。Amazon SAM带模板文件`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 (Amazon ECR) 构建容器镜像，请参阅映像存储库 (p. 325).

使用 Jenkins 进行部署

配置您的詹金斯管道来自动构建和部署Amazon SAM您的应用程序`Jenkinsfile`必须包含执行以下操作的行：

1. 从可用映像中使用必要的运行时间来引用构建容器映像。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命令行界面 (CLI) 命令。以下示例运行两个Amazon SAMCLI 命令：sam build和sam deploy（有必要的选择）。

此示例假定您已在您中声明了所有函数和层。Amazon SAM使用模板文件`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'
            }
        }
    }
}
```

有关适用于不同运行时的可用 Amazon Elastic Container Registry (Amazon ECR) 构建容器镜像的列表，请参阅映像存储库 (p. 325).

使用 GitLab CI/CD 进行部署

配置您的GitLab管道来自动构建和部署Amazon SAM您的应用程序`gitlab-ci.yml`文件必须包含执行以下操作的行：

1. 从可用映像中使用必要的运行时间来引用构建容器映像。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命令行界面 (CLI) 命令。以下示例运行两个。Amazon SAMCLI 命令：sam build和sam deploy（有必要的选择）。

此示例假定您已在中声明了所有函数和层。Amazon SAM带有模板文件`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) 构建容器镜像的列表，请参阅映像存储库 (p. 325).

使用部署 GitHub 操作

配置您的GitHub管道来自动构建和部署您的Amazon SAM应用程序，您必须首先安装Amazon SAM主机上的命令行界面 (CLI)。您可以使用GitHub 操作在您的 GitHub 帮助进行此设置的工作流程。

以下示例 GitHub 工作流使用一系列 GitHub 操作设置 Ubuntu 主机，然后运行Amazon SAM构建和部署 CLI 命令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
```

要了解适用于不同运行时的 Amazon Elastic Container Registry (Amazon ECR) 构建容器镜像的列表，请参阅映像存储库 (p. 325).

生成初学者 CI/CD 管道

当您准备好以自动方式部署无服务器应用程序时，您可以为您选择的 CI/CD 系统生成部署管道。Amazon SAM提供了一组入门管道模板，您可以使用这些模板在几分钟内使用sam 管道 init (p. 276)命令。

初学者管道模板使用 CI/CD 系统熟悉的 JSON/YAML 语法，并融入了最佳实践，例如跨多个账户和区域管理工件，以及使用部署应用程序所需的最低权限。目前，Amazon SAMCLI 支持生成初学者 CI/CD 管道配置Amazon CodePipeline、詹金斯、GitLab CI/CD、GitHub 操作, 和Bitbucket 管道.

以下是生成启动管道配置所需执行的高级任务：

1. 创建基础设施资— 你的管道需要一定的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 Git 存储库与 CI/CD 系统 Connect 起来— 你的 CI/CD 系统需要知道哪个 Git 存储库将触发管道运行。请注意，此步骤可能不是必要的，具体取决于您使用的 Git 存储库和 CI/CD 系统的组合。
3. 生成管道配置— 此步骤将生成包括两个部署阶段的入门管道配置。
4. 将管道配置提交到 Git 存储库— 此步骤是必要的，以确保 CI/CD 系统了解您的管道配置，并在提交更改时运行。

在您生成入门管道配置并将其提交到 Git 存储库之后，每当有人向该仓库提交代码更改时，您的管道将被触发以自动运行。

这些步骤的顺序以及每个步骤的详细信息根据您的 CI/CD 系统的不同而有所不同：

- 如果您正在使用 Amazon CodePipeline，请参阅 为生成初学者管道Amazon CodePipeline (p. 225)。
- 如果您使用的是 Jenkins、GitLab CI/CD、GitHub 操作或 Bitbucket 管道，请参阅为詹金斯、GitLab CI/CD、GitHub 操作或 Bitbucket 管道生成入门管道 (p. 227).

为生成初学者管道Amazon CodePipeline

生成初学者管道配置Amazon CodePipeline中，请按以下顺序执行以下任务：

1. 创建基础设施资

2. 生成管道配置
3. 将管道配置提交给 Git
4. 将 Git 存储库与 CI/CD 系统 Connect 起来

> **Note**
>
> 以下过程使用两个Amazon SAM CLI commands, `sam #### (p. 275)`和`sam ## init (p. 276)`. 有两个命令的原因是处理管理员（即需要权限才能设置基础架构的用户）的使用案例Amazon资源（如 IAM 用户和角色）拥有更多的权限，开发人员（即只需要权限来设置单个管道但不需要所需基础设施的用户）的权限Amazon资源)。

第 1 步：创建基础设施资

使用的管道Amazon SAM需要一定的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。管道的每个部署阶段都必须拥有一组基础架构资源。

您可以运行以下命令来帮助此设置：

```
sam pipeline bootstrap
```

> **Note**
>
> 为管道的每个部署阶段运行上一个命令。

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：将管道配置提交到 Git 存储库

此步骤是必要的，以确保 CI/CD 系统了解您的管道配置，并在提交更改时运行。

第 4 步：将 Git 存储库与 CI/CD 系统 Connect 起来

适用于Amazon CodePipeline您现在可以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>
```

如果你正在使用 GitHub 或 Bitbucket，在运行sam deploy命令之前，请按照下面的步骤完成连接完成连接在更新挂起的连接中的主题控制台用户指南. 此外，还可以存储`CodeStarConnectionArn`从输出中sam deploy命令，因为如果您想使用的话将需要它Amazon CodePipeline还有另一个分支机构`main`.

配置其他分支

默认情况下，Amazon CodePipeline使用`main`使用分支Amazon SAM. 如果您想使用以外的分支`main`，您必须运行sam deploy再次命令。请注意，根据您使用的 Git 存储库，您可能还可能需要提供`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>"
```

为詹金斯、GitLab CI/CD、GitHub 操作或 Bitbucket 管道生成入门管道

要为 Jenkins、GitLab CI/CD、GitHub 操作或 Bitbucket 管道生成启动管道配置，请按以下顺序执行以下任务：

1. 创建基础设施资
2. 将 Git 存储库与 CI/CD 系统 Connect 起来
3. 创建凭证对象
4. 生成管道配置
5. 将管道配置提交到 Git 存储库

> **Note**
>
> 以下过程使用两个Amazon SAMCLI 命令，sam #### (p. 275)和sam ## init (p. 276). 有两个命令的原因是处理管理员（即需要权限才能设置基础架构的用户）的使用案例Amazon资源（如 IAM 用户和角色）拥有更多的权限，开发人员（也就是说，只需要权限来设置单个管道的用户，但不需要所需的基础设施的用户Amazon资源)。

第 1 步：创建基础设施资

使用的管道Amazon SAM需要一定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。管道的每个部署阶段都必须拥有一组基础架构资源。

您可以运行以下命令来帮助此设置：

```
sam pipeline bootstrap
```

> **Note**
>
> 为管道的每个部署阶段运行上一个命令。

你必须捕获Amazon管道的每个部署阶段的管道用户的凭据（密钥 ID 和密钥），因为后续步骤都需要这些证书（密钥 ID 和密钥）。

第 2 步：将 Git 存储库与 CI/CD 系统 Connect 起来

将 Git 存储库连接到 CI/CD 系统是必要的，这样 CI/CD 系统才能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，因为连接是自动为您完成的：
>
> 1. GitHub 仓库的 GitHub 操作

2. 带有 GitLab 存储库的 GitLab CI/CD
3. 带有 Bitbucket 存储库的 Bitbucket 管道

要将 Git 存储库与 CI/CD 系统连接，请执行以下操作之一：

- 如果您使用的是 Jenkins，请参阅Jenkins 文档对于 “添加分支源”。
- 如果你使用的是 GitLab CI/CD 和 GitLab 之外的 Git 存储库，请参阅GitLab 文档用于 “连接外部存储库”。

第 3 步：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 Git 存储库所需的凭据。

要创建必要的凭证对象，请执行以下操作之一：

- 如果你使用的是 Jenkins，请创建一个存储密钥 ID 和秘密密钥的单个 “凭据”。按照中的说明进行操作使用构建詹金斯管道Amazon SAM博客，在配置 Jenkins部分。您在下一步中需要 “凭证 ID”。
- 如果您使用的是 GitLab CI/CD，请创建两个 “受保护的变量”，每个用于密钥 ID 和密钥。按照中的说明进行操作GitLab 文档-您在下一步中需要两个 “可变键”。
- 如果您使用的是 GitHub Actions，请创建两个 “加密密钥”，每个密钥和密钥一个。按照中的说明进行操作GitHub 文档-您在下一步中需要两个 “密钥名称”。
- 如果您使用的是 Bitbucket Pipeline，请创建两个 “安全变量”，每个用于密钥 ID 和密钥。按照中的说明进行操作变量和密钥-您在下一步中需要两个 “密钥名称”。

第 4 步：生成管道配置

要生成管道配置，请运行以下命令。您需要输入上一步中创建的凭证对象：

```
sam pipeline init
```

第 5 步：将管道配置提交到 Git 存储库

此步骤是必要的，以确保 CI/CD 系统了解您的管道配置，并在提交更改时运行。

自定义初学者管道

作为 CI/CD 管理员，您可能需要自定义初学者管道模板和相关的指导提示，组织中的开发人员可以使用这些模板创建管道配置。

这些区域有：Amazon SAMCLI 在创建初学者模板时使用 Cookiecutter 模板。有关 cookie 切割器模板的详细信息，Cookiecutter.

您还可以自定义提示Amazon SAM使用创建管道配置时，CLI 向用户显示`sam pipeline init`命令。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。这是相同的目录`cookiecutter.json`文件。查看的架构`questions.json`文件，请参阅问题.json.schema. 要查看示例`questions.json`文件，请参阅问题.json.
2. 将问题键与 cookie 切割名称映射— 中的每个对象`questions.json`文件需要一个与 cookiecutter 模板中的名称匹配的密钥。这个关键匹配就是Amazon SAMCLI 将用户提示响应映射到 cookie 切割器模板。要查看此密钥匹配的示例，请参阅示例文件 (p. 229)在本主题后面的一节。

3. 创建**metadata.json**文件— 宣布管道将在metadata.json文件。指示阶段的数量sam pipeline init命令提示有多少阶段提示信息，或者在--bootstrap选项，为多少个阶段创建基础架构资源。要查看示例metadata.json声明具有两个阶段的管道的文件，请参阅metadata.json.

示例项目

以下是示例项目，每个项目都包括一个 Cookiecutter 模板，questions.json文件，和metadata.jsonfile:

- Jenkins 示例：两阶段詹金斯管道模板
- CodePipeline 示例：两阶段 CodePipeline 管道模板

示例文件

以下一组文件显示了中的问题如何questions.json文件与 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。要查看完整文件的示例，请参阅示例项目 (p. 229)在本主题前面的一节。

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}}'
    ...
```

无服务器应用程序

将无服务器应用程序部署到Amazon云端，您需要验证它是否持续运行正常。

主题

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. 272). 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。

使用发布无服务器应用程序Amazon SAMCLI

为了让您Amazon SAM可供他人查找和部署的应用程序，您可以使用Amazon SAMCLI 将其发布到Amazon Serverless Application Repository. 要使用发布应用程序Amazon SAMCLI，您必须使用Amazon SAM。模板。你还必须在本地或在Amazon云。

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。（你做什么取决于应用程序是否已经存在于Amazon Serverless Application Repository，以及是否有任何应用程序元数据在更改。）有关应用元数据的更多信息，请参阅Amazon SAM元数据部分属性 (p. 234).

先决条件

在将应用程序发布到Amazon Serverless Application Repository使用Amazon SAMCLI，您必须满足以下条件：

- 这些区域有：Amazon SAM已安装 CLI。有关更多信息，请参阅 安装 Amazon SAM CLI (p. 3)。确定是否Amazon SAM安装 CLI，请运行以下命令：

```
sam --version
```

- 一个有效Amazon SAM。模板。
- 您的应用程序代码和依赖项Amazon SAM模板引用。
- 一个语义版本，仅需要公开共享您的应用程序。该值可以像 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，以及如何在自行部署应用程序之前对其进行配置Amazon账户。
- 一个`LICENSE.txt`文件，仅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.
- 一个有效的 Amazon Simple Storage Service (Amazon S3) 存储桶策略，它为在您打包应用程序时上传到 Amazon S3 的项目授予服务读取权限。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. 选择您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. 在存储库的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。在策略声明中，请确保在`Resource`和你的Amazon中的账户 ID`Condition`元素。中的表达式`Condition`元素可以确保Amazon Serverless Application Repository只能访问来自指定的应用程序Amazonaccount. 有关策略声明的更多信息，请参阅IAM JSON 策略元素参考中的IAM 用户指南.

```
{
    "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

第 1 步：添加Metadata到部分Amazon SAM模板

首先，添加Metadata部分到你的Amazon SAM。模板。提供要发布的应用程序信息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
        ...
```

有关的更多信息Metadata的 部分Amazon SAM模板，请参阅Amazon SAM元数据部分属性 (p. 234).

第 2 步：Package 应用程序

运行以下命令Amazon SAMCLI 命令，该命令将应用程序的工件上传到 Amazon S3 并输出名为packaged.yaml：

```
sam package --output-template-file packaged.yaml --s3-bucket <your-bucket-name>
```

使用packaged.yaml将应用程序发布到Amazon Serverless Application Repository.（此文件类似于原始模板文件template.yaml），但它有一个关键的区别-CodeUri、LicenseUrl, 和ReadmeUrl属性指向 Amazon S3 存储桶和包含各自工件的对象。

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/fbd77a3647a4f47a352fcObjectGUID

...
```

第 3 步：发布应用程序

要发布私有版本的Amazon SAM应用到Amazon Serverless Application Repository，运行以下命令Amazon SAMCLI 命令：

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

的输出sam publish命令包括指向应用程序的链接Amazon Serverless Application Repository. 您也可以直接转到Amazon Serverless Application Repository登录页面然后搜索您的应用程序。

第 4 步：共享应用程序（可选）

默认情况下，您的应用程序设置为私有，因此其他应用程序不可见Amazon账户。要与其他人共享您的应用程序，您必须将其公开或授予特定列表的权限Amazon账户。

有关使用共享应用程序的信息Amazon CLI，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. 有关使用共享应用程序的信息Amazon Web Services Management Console，请参阅共享应用程序中的Amazon Serverless Application Repository开发人员指南.

发布现有应用程序的新版本

在将应用程序发布到Amazon Serverless Application Repository，您可能想要发布它的新版本。例如，您可能已经更改了 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，请使用之前详细介绍的相同流程再次发布该应用程序。在Metadata的部分Amazon SAM模板文件，请提供与您最初发布该文件相同的应用程序名称，但包含一个新的SemanticVersion值。

例如，考虑使用名称发布的应用程序SampleApp和SemanticVersion的1.0.0. 要更新该应用程序，Amazon SAM模板必须有应用程序名称SampleApp和SemanticVersion的1.0.1（或者除了其他任何东西1.0.0）。

其他主题

- Amazon SAM元数据部分属性 (p. 234)

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
```

示例无服务器程序

以下示例展示了如何下载、测试和部署一些其他无服务器应用程序，包括如何配置事件源和Amazon资源的费用。

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知识进行构建，然后安装另一个示例应用程序。此应用程序由一个由 DynamoDB 表事件源调用的 Lambda 函数组成。Lambda 函数非常简单 — 它会记录通过事件源消息传入的数据。

本练习向您展示如何模拟调用 Lambda 函数时传递给 Lambda 函数的事件源消息。

开始前的准备工作

请确保您已完成中的所需设置安装 Amazon SAM CLI (p. 3).

第 1 步：初始化应用程序

在本节中，您下载应用程序包，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 gh:aws-samples/cookiecutter-aws-sam-dynamodb-python \
--no-input
```

   请注意`gh:`在上面的命令中被扩展到 GitHub URL`https://github.com/`.
2. 查看命令创建的目录的内容 (`dynamodb_event_reader/`)：

   - `template.yaml`— 定义两个Amazon读取 DynamoDB 应用程序需要的资源：Lambda 函数和 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。

第 2 步：在本地测试应用程序

对于本地测试，请使用Amazon SAM用于生成示例 DynamoDB 事件并调用 Lambda 函数的 CLI：

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

这些区域有：`generate-event`命令创建测试事件源消息，如在将所有组件部署到Amazon云。此事件源消息通过管道传送到 Lambda 函数 readdynamoDBEvent。

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.

第 3 步：打包应用程序

在本地测试应用程序后，您可以使用Amazon SAM用于创建部署程序包的 CLI，您可以使用该软件包将应用程序部署到Amazon云。

创建 Lambda 部署包

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. 通过运行以下命令创建部署程序包package在命令提示符下显示 CLI 命令。

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

您可以指定新的模板文件，packaged.yaml，当您在下一步骤中部署应用程序时。

第 4 步：部署应用程序

现在您已创建部署程序包，您可以使用它将应用程序部署到Amazon云。然后测试应用程序。

将无服务器应用程序部署到Amazon云

- 在Amazon SAM使用 CLI，请使用deploy用于部署模板中定义的所有资源的 CLI 命令。

```
sam deploy \
    --template-file packaged.yaml \
    --stack-name sam-app \
    --capabilities CAPABILITY_IAM \
    --region us-east-1
```

在命令中，--capabilities参数允许Amazon创建 IAM 角色的 CloudFormation。

Amazon CloudFormation创建Amazon在模板中定义的资源。您可以在Amazon CloudFormation控制台。

要在Amazon云

1. 打开 DynamoDB 控制台。
2. 将记录插入您刚创建的表中。
3. 转至指标表格的选项卡，然后选择查看全部 CloudWatch 指标. 在 CloudWatch 控制台中，选择日志以便能够查看日志输出。

后续步骤

这些区域有：Amazon SAMGitHub 存储库包含其他示例应用程序供您下载和试验。要访问此存储库，请参阅Amazon SAM示例程序.

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基础上进行构建，并安装更复杂的应用程序。此应用程序由 Amazon S3 数据元上传事件源调用的 Lambda 函数组成。此练习向您演示如何访问Amazon资源和制造Amazon通过 Lambda 函数进行服务调用。

此示例无服务器应用程序处理 Amazon S3 中的对象创建事件。对于上传到存储桶的每个映像，Amazon S3 都会检测对象创建的事件并调用 Lambda 函数。Lambda 函数调用 Amazon Rekognition 来检测图像中的文本。然后，它会将 Amazon Rekognition 返回的结果存储在 DynamoDB 表中。

> Note
>
> 使用此示例应用程序，您执行步骤的顺序与之前的示例略有不同。这样做的原因是这个例子要求Amazon已创建资源并配置了 IAM 权限以前您可以在本地测试 Lambda 函数。我们将利用Amazon CloudFormation以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。
> 由于此示例比较复杂，所以在执行此示例应用程序之前，请务必熟悉安装前面的示例应用程序。

开始前的准备工作

请确保您已完成中的所需设置安装 Amazon SAM CLI (p. 3).

第 1 步：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`— 定义三个AmazonAmazon S3 应用程序所需的资源：Lambda 函数、Amazon S3 存储桶和 DynamoDB 表。此模板还定义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`— 用于本地测试的示例事件源。

第 2 步：打包应用程序

在本地测试此应用程序之前，必须使用Amazon SAM用于创建部署程序包的 CLI，您可以使用该软件包将应用程序部署到Amazon云。此部署创建了必要的Amazon本地测试应用程序所需的资源和权限。

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. 通过运行以下命令创建部署程序包package命令提示符下的 CLI 命令。

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

您可以指定新的模板文件，packaged.yaml，当您在下一步骤中部署应用程序时。

第 3 步：部署应用程序

既然您已创建部署包，就可以使用它将应用程序部署到Amazon云。然后，您可以通过在Amazon云。

将无服务器应用程序部署到Amazon云

- 在Amazon SAMCLI，请使用deploy命令以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参数允许Amazon CloudFormation创建 IAM 角色。

Amazon CloudFormation创建Amazon在模板中定义的资源。您可以在Amazon CloudFormation控制台。

要在Amazon云

1. 将图像上传到您为此示例应用程序创建的 Amazon S3 存储桶。
2. 打开 DynamoDB 控制台然后找到创建的表。有关 Amazon Rekognition 返回的结果，请参阅表格。
3. 验证 DynamoDB 表包含包含 Amazon Rekognition 在上传图片中找到的文本的新记录。

第 4 步：本地测试应用程序

在本地测试应用程序之前，必须先检索Amazon创建的资源Amazon CloudFormation.

- 从中检索 Amazon S3 密钥名称和存储桶名称Amazon CloudFormation. 修改SampleEvent.json文件的方法是替换对象密钥、存储桶名称和存储桶 ARN 的值。
- 检索 DynamoDB 表名。此名称用于以下内容sam local invoke命令。

使用Amazon SAMCLI 以生成示例 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
```

这些区域有：TABLE_NAME=部分用于设置 DynamoDB 表名。这些区域有：--event参数指定包含要传递到 Lambda 函数的测试事件消息的文件。

现在，您可以根据 Amazon Rekognition 返回的结果来验证预期的 DynamoDB 记录是否已创建。

后续步骤

这些区域有：Amazon SAMGitHub 存储库包含其他示例应用程序供您下载和试验。要访问此存储库，请参阅Amazon SAM例应用程序.

Amazon Cloud Development Kit (Amazon CDK)

您可以使用Amazon SAMCLI 用于在本地测试和构建使用Amazon Cloud Development Kit (Amazon CDK). 由于Amazon SAMCLI 在Amazon CDK项目结构，您仍然可以使用Amazon CDK工具包用于创建、修改和部署Amazon CDK应用程序。

有关安装和配置的信息Amazon CDK，请参阅开始使用Amazon CDK中的Amazon Cloud Development Kit (Amazon CDK)开发人员指南.

> Note
>
> 这些区域有：Amazon SAMCLI 支持Amazon CDKv1 从 1.135.0 版本开始和Amazon CDKv2 从 2.0.0 版本 2.0.0 开始。

主题

开始使用Amazon SAM和Amazon CDK

本主题介绍您需要使用Amazon SAMCLI 与Amazon CDK应用程序，并提供了构建和本地测试简单的说明Amazon CDK应用程序.

先决条件

使用Amazon SAMCLI 与Amazon CDK，您必须安装Amazon CDK，以及Amazon SAMCLI。

- 有关安装的信息Amazon CDK，请参阅开始使用Amazon CDK中的Amazon Cloud Development Kit (Amazon CDK)开发人员指南.
- 有关安装的信息Amazon SAMCLI，请参阅安装 Amazon SAM CLI (p. 3).

创建和本地测试Amazon CDK应用程序

在本地测试Amazon CDK应用程序使用Amazon SAMCLI，您必须已执行Amazon CDK包含 Lambda 函数的应用程序。请按照以下步骤创建基本版本Amazon CDK具有 Lambda 函数的应用程序。有关更多信息，请参阅 。使用创建无服务器应用程序Amazon CDK中的Amazon Cloud Development Kit (Amazon CDK)开发人员指南.

> Note
>
> 这些区域有：Amazon SAMCLI 支持Amazon CDKv1 从 1.135.0 版本开始和Amazon CDKv2 从版本 2.0.0 开始。

第 1 步：创建Amazon CDK应用程序

在本教程中，请初始化Amazon CDK使用 TypeScript 的应用程序。

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
```

第 2 步：将 Lambda 函数添加到应用程序

在中替换代码`lib/cdk-sam-example-stack.ts`有以下权限：

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'),
    });
  }
}
```

第 3 步：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!"
```

第 4 步：测试 Lambda 函数

您可以使用Amazon SAMCLI 来本地调用您在Amazon CDK应用程序. 要执行此操作，您需要函数构造标识符和合成路径Amazon CloudFormation模板。

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
```

有关可测试选项的更多信息Amazon CDK应用程序使用Amazon SAMCLI，请参阅本地测试Amazon CDK应用程序 (p. 245).

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. 265)
- sam 本地 start-api (p. 267)
- sam 本地 start-lambda (p. 269)

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sam build (p. 252).

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加速

您可以使用Amazon SAM加快更新和监控中的无服务器应用程序Amazon Web Services 云在开发过程中。

Amazon SAM加快从开发环境到Amazon Web Services 云使用Amazon服务 API 而不是Amazon CloudFormation以部署代码更新。Amazon SAM加速还支持自动部署到Amazon Web Services 云更改应用程序时。

部署到Amazon Web Services 云在开发过程中，您可以识别应用程序在本地环境中难以检测到的问题。例如，在Amazon Web Services 云可以帮助您识别 IAM 角色或 API 授权方面的问题。

这些区域有：??? (p. 277)命令将您的本地更改部署到Amazon Web Services 云.

您可以使用??? (p. 272)和??? (p. 280)监控无服务器应用程序的命令。

有关加速的更多信息，请参阅无服务器土地.

主题

- 开始使用Amazon SAM加速 (p. 248)

开始使用Amazon SAM加速

本主题介绍您需要使用的内容。Amazon SAM加速，并提供有关构建和部署简单应用程序的说明。

先决条件

使用Amazon SAM加速，你必须安装 1.53.0 或更高版本Amazon SAMCLI。有关安装说明，请参阅安装 Amazon SAM CLI (p. 3)。

入门

在本指南中，您将使用来下载、构建和部署示例 Hello World 应用程序。Amazon SAM. 然后你对代码进行修改Amazon SAM在中加速自动部署和测试应用程序Amazon Web Services 云.

此应用程序实现了基本的 API 后端。

先决条件

本教程假定您熟悉Amazon SAM. 有关更详细的教程，请参阅the section called “教程：Hello World 申请” (p. 15).

第 1 步：下载示例Amazon SAM应用程序

要运行的命令：

```
sam init --app-template hello-world --name sam-tutorial --package-type Zip --runtime
 python3.9
```

在本教程中，我们使用 “你好世界” Python 应用程序和`Zip`程序包类型。

第 2 步：开始 sam 同步 —Watch

首先，更改为sam-tutorial目录，其中template.yaml已找到示例应用程序的文件。然后，运行以下命令以启动监视您的无服务器应用程序的更改。响应Y当系统提示您确认是否要使用预览功能时。

```
sam sync --watch --stack-name sam-app
```

你第一次运行sync --watch命令，Amazon SAM启动Amazon CloudFormation部署。部署之后，Amazon SAM监视您的无服务器应用程序的更改。对于代码资源（例如 Lambda 函数）的后续更改，Amazon SAM使用服务 API 自动部署更改。对于基础设施的更改，例如 IAM 角色，Amazon SAM自动启动Amazon CloudFormation部署。

第 3 步：对你的应用程序进行更改

使用sync --watch进程正在运行，更新本地 Lambda 函数代码。Amazon SAM自动构建 Lambda 函数，并将更新部署到Amazon Web Services 云.Amazon SAM调用 Lambda API 来更新函数的代码，而不是部署Amazon CloudFormation堆栈。部署该更改需要几秒钟时间。

将函数代码更改为以下内容以编写Invoking the updated function到您的 Lambda 函数日志：

```
import json
def lambda_handler(event, context):
    print("Invoking the updated function")
    return {
        "statusCode": 200,
        "body": json.dumps({
            "message": "hello world",
        }),
    }
```

第 4 步：测试你的应用程序并检查日志

使用调用你的 APIcurl，然后检查 Lambda 函数的日志。

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

使用logs (p. 272)命令从应用程序中获取日志。

```
sam logs --tail
```

如果你明白Invoking the updated function在日志中，您已成功将 Lambda 函数更新部署到Amazon Web Services 云.

> Note
>
> 由于加速使用 Lambda API 来更新您的函数代码，Amazon CloudFormation堆栈不反映应用程序的更新。更新您的Amazon CloudFormation堆叠最新的更改，运行sam sync.

清除

如果您不再需要Amazon通过运行本教程创建的资源，则可以通过删除Amazon CloudFormation你部署的堆栈。

删除Amazon CloudFormation堆栈，使用sam --delete命令：

```
sam delete --stack-name sam-app
```

结论

在本教程中，您完成了以下操作：

1. 创建、构建和部署了无服务器应用程序Amazon使用Amazon SAM.
2. 使用`sam sync --watch`将更改自动部署到Amazon Web Services 云.
3. 在Amazon Web Services 云.
4. 已删除Amazon您不再需要的资源。

Amazon SAM 引用

Amazon SAM规格

这些区域有：Amazon SAM规范是 Apache 2.0 许可证下的开源规范。的当前版本Amazon SAM规格可在Amazon Serverless Application Model(Amazon SAM) 规范 (p. 26).

Amazon SAM模板是的扩展Amazon CloudFormation模板。有关以下内容的完整参考Amazon CloudFormation模板，请参阅Amazon CloudFormation模板参考.

Amazon SAMCLI 命令参考

这些区域有：Amazon SAMCLI是一个命令行工具，可在Amazon SAM模板和应用程序代码。使用Amazon SAMCLI，您可以在本地调用 Lambda 函数，为无服务器应用程序创建部署程序包，将无服务器应用程序部署到Amazon云等。

您可以使用Amazon SAMClit for Visual Studio for Visual Studio VisualAmazon云。以下是以下是一些示例 Amazon SAMCLI 命令：

- `sam init`— 如果你是新的Amazon SAMCLI 用户，你可以运行`sam init`不带任何参数的命令来创建 Hello World 应用程序。该命令生成预配置的Amazon SAM模板和所选语言的示例应用程序代码。
- `sam local invoke`和`sam local start-api`— 使用这些命令在本地测试您的应用程序代码，然后再将其部署到Amazon云。
- `sam logs`— 使用此命令提取您的 Lambda 函数生成的日志。这可以帮助你在将应用程序部署到应用程序后对其进行测试和调试Amazon云。
- `sam package`— 使用此命令将您的应用程序代码和依赖项捆绑到 “部署包” 中。需要部署包才能将您的应用程序上传到Amazon云。
- `sam deploy`— 使用此命令将您的无服务器应用程序部署到Amazon云。它创建Amazon资源和设置权限和其他配置，这些权限和配置在中定义Amazon SAM模板。

有关安装的说明Amazon SAMCLI，请参阅安装 Amazon SAM CLI (p. 3).

Amazon SAM策略模板

Amazon SAM允许您从策略模板列表中进行选择，将 Lambda 函数的权限范围限定为应用程序使用的资源。

主题

Amazon SAMCLI 命令参考

本节是参考Amazon SAMCLI 命令。有关安装Amazon SAMCLI，请参阅安装 Amazon SAM CLI (p. 3).

主题

sam build

构建无服务器应用程序并为工作流程中的后续步骤做好准备，例如在本地测试应用程序或将其部署到Amazon云。如果你提供`RESOURCE_LOGICAL_ID`，那么Amazon SAM只构建该资源。要构建嵌套应用程序或堆栈的资源，可以使用格式提供应用程序或堆栈逻辑 ID 以及资源逻辑 ID`StackLogicalId/ResourceLogicalId`.

这些区域有：`sam build`命令处理你的Amazon SAM模板文件、应用程序代码以及任何特定于语言的适用文件和依赖关系。该命令还以工作流程后续步骤预期的格式和位置复制构建工件。您可以在应用程序中包含的清单文件中指定依赖关系，例如`requirements.txt`对于 Python 函数，或者`package.json`对于 Node.js 函数。

应用程序的构建工件的格式取决于其软件包类型。你指定你的Amazon Lambda函数的包类型与`PackageType`财产。选项包括：

- **`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，除应用程序代码及其依赖项外，还包括基本操作系统、运行时和扩展。

有关 Lambda 包类型的更多信息，请参阅Lambda 部署程序包中的Amazon Lambda开发人员指南.

如果资源包含Metadata资源属性带BuildMethod进入，sam build根据BuildMethod入口。的有效值BuildMethod是 1) Lambda 运行时的标识符之一，或 2)makefile标识符。

- Lambda 运行时标识符— 根据 Lambda 运行时构建资源。有关支持的运行时标识符的列表，请参阅Lambda 运行时中的Amazon Lambda开发人员指南.
- **makefile**标识符 — 运行资源的构建目标的命令。在这种情况下，你的 makefile 必须命名Makefile并包括一个名为的构建目标build-*resource-logical-id*.

要构建层和自定义运行时，您还可以使用Metadata资源属性带BuildMethod入口。有关构建图层的信息，请参阅构建层 (p. 209). 有关构建自定义运行时的信息，请参阅构建自定义运行时 (p. 211).

对于具有Image包装类型，请使用Metadata资源属性，用于配置构建容器映像所需的 Docker 映像设置。有关构建容器映像的更多信息，请参阅构建容器镜像 (p. 205).

有关使用此命令的完整示例，包括在本地测试和部署到Amazon云，请参阅教程：部署 Hello World 应用程序 (p. 15). 这些区域有：sam build命令是的一部分第 2 步：构建你的应用 (p. 17).

使用方法：

```
sam build [OPTIONS] [RESOURCE_LOGICAL_ID]
```

示例：

```
To use these commands, update your SAM template to specify the path
to your function's source code in the resource's Code or CodeUri property.

To build on your workstation, run this command in the directory containing your
SAM template. Built artifacts are written to the .aws-sam/build directory.
$ sam build

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

To build with environment variables passed to the build container from the command line
$ sam build --use-container --container-env-var Function1.GITHUB_TOKEN=<token1> --
container-env-var GLOBAL_ENV_VAR=<global-token>

To build with environment variables passed to the build container from a file
$ sam build --use-container --container-env-var-file <env-file.json>

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

To build and run your functions locally
$ sam build && sam local invoke

To build and package for deployment
$ sam build && sam package --s3-bucket <bucketname>

To build the 'MyFunction' resource
$ sam build MyFunction

To build the 'MyFunction' resource of the 'MyNestedStack' nested stack
$ sam build MyNestedStack/MyFunction
```

参数：

参数	描述
RESOURCE_LOGICAL_ID	可选。指示Amazon SAM构建中声明的单个资源Amazon SAM模板。指定资源的构建工件将是工作流中后续命令的唯一可用的构建工件，即sam package和sam deploy.

选项：

选项	描述
-b, --build-dir DIRECTORY	到存储构建工件的目录的路径。使用此选项将删除此目录及其所有内容。
-s, --base-dir DIRECTORY	解析与此目录相关的函数或图层源代码的相对路径。如果要更改与源代码文件夹的相对路径的解析方式，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。 除了要构建的根应用程序或堆栈中的资源外，此选项还应用嵌套的应用程序或堆栈。 此选项适用于以下资源类型和属性： • 资源类型：AWS::Serverless::Function财产：CodeUri • 资源类型：AWS::Serverless::Function资源属性：Metadata条目DockerContext • 资源类型：AWS::Serverless::LayerVersion财产：ContentUri • 资源类型：AWS::Lambda::Function财产：Code • 资源类型：AWS::Lambda::LayerVersion财产：Content
-u, --use-container	如果你的函数依赖于具有本地编译依赖关系的软件包，请使用此选项在类Lambda Docker 容器中构建函数。
-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，其中键是资源变量和环境变量，值是环境变量的值。例如：--container-env-var Function1.GITHUB_TOKEN=TOKEN1 --container-env-var Function2.GITHUB_TOKEN=TOKEN2。 此选项仅适用于--use-container选项被指定，否则将导致出现错误。
-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. 205). 此选项仅适用于--use-container选项被指定，否则将导致出现错误。
--build-image TEXT	您要为构建而提取的容器映像的 URI。默认情况下，Amazon SAM从Amazon ECR Public 中提取容器镜像。使用此选项从另一个位置拉取图像。 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。如果指定键值对，则键为资源名称，值是要用于该资源的容器镜像的 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。

选项	描述
	此选项仅适用于`--use-container`选项被指定，否则将导致出现错误。
`-m, --manifest PATH`	要使用的自定义依赖关系清单文件（例如 package.json）的路径，而不是默认值。
`-t, --template-file, --template PATH`	的路径和文件名Amazon SAM模板文件`[default: template.[yaml|yml]]`.
`--parameter-overrides`	（可选）包含的字符串Amazon CloudFormation参数覆盖编码为键值对。使用的格式与Amazon Command Line Interface(Amazon CLI)。例如：`'ParameterKey=KeyPairName、ParameterValue=MyKey ParameterKey=InstanceType、ParameterValue=t1.micro'`。
`--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。
`--parallel`	启用 parallel 构建。使用此选项来构建Amazon SAM模板的功能和层parallel。默认情况下，函数和图层是按顺序构建的。
`--cached | --no-cached`	启用或禁用缓存的构建。使用此选项可以重复使用与之前的构建版本没有更改的构建工件。Amazon SAM评估您是否更改项目目录中的任何文件。默认情况下，不会缓存构建。如果`--no-cached`选项被调用，它会覆盖`cached = true`在 samcofig.toml 中设置。注意： Amazon SAM 不会评估您是否更改了项目所依赖的第三方模块，而您尚未提供特定版本。例如，如果你的 Python 函数包含`requirements.txt`带条目的文件`requests=1.x`，最新的请求模块版本更改自`1.1`到`1.2`，那么Amazon SAM在运行非缓存的构建之前，才会提取最新版本。
`--cache-dir`	当时存储缓存工件的目录`--cached`已指定。默认缓存目录为`.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`Function1、Function2`, 和`Function3`然后你跑`sam build --exclude Function2`仅限`Function1`和`Function3`将建成。
`--profile TEXT`	从您的凭证文件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`	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”`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. 281)。
`--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。有关配置文件的详细信息，请参阅 Amazon SAM CLI 配置文件 (p. 281)。
`--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。

示例

使用 Lambda 运行时标识符构建资源

示例如下：Amazon SAM模板显示了如何使用 Lambda 运行时标识符构建资源：

```
Resources:
  MyLayer:
    Type: AWS::Serverless::LayerVersion
    Properties:
      ContentUri: my_layer
      CompatibleRuntimes:
        - python3.6
    Metadata:
      BuildMethod: python3.6
```

使用此模板，以下命令将构建`MyLayer`针对 Python 3.6 运行时环境的资源：

```
sam build MyLayer
```

使用`makefile`标识符

示例如下：Amazon SAM展示如何使用`makefile`标识符：

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

这是关联 makefile 的示例。该文件必须命名为`Makefile`，并将构建目标和您要运行的命令包括在内：

```
build-MyLayer:
    mkdir -p "$(ARTIFACTS_DIR)/python"
    cp *.py "$(ARTIFACTS_DIR)/python"
    python -m pip install -r requirements.txt -t "$(ARTIFACTS_DIR)/python"
```

使用此模板和 makefile，以下命令将执行`build-MyLayer`目标：

```
sam build MyLayer
```

将环境变量传递到构建容器

下面是一个示例，展示如何使用文件将环境变量传递给构建容器。

首先，创建名为的文件`env.json`包含以下内容：

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

```
}
```

然后，运行以下命令：

```
sam build --use-container --container-env-var-file env.json
```

更多有关容器环境变量文件信息，请参阅。容器环境变量文件 (p. 205).

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. 281)。
`--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. 281)。
`--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

部署Amazon SAM应用程序。

默认情况下，当你使用这个命令时，Amazon SAMCLI 假定您当前的工作目录是项目的根目录。这些区域有：Amazon SAMCLI 首先尝试找到使用创建的模板文件sam build (p. 252)命令，位于`.aws-sam`子文件夹，并命名为`template.yaml`. 接下来，则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。

要覆盖Amazon SAMCLI 的默认行为，请指定`--template`选项。指定此选项时，则Amazon SAMCLI 只部署那个Amazon SAM模板及其指向的本地资源。

要打开引导式交互模式，请指定`--guided`选项。此模式显示部署所需的参数，提供默认选项，并可选择将这些选项保存在项目目录的配置文件中。当您使用以下方法执行应用程序的后续部署时sam deploy，Amazon SAMCLI 从配置文件中检索所需的参数。

在中声明的对象`Parameters`部分Amazon SAM模板文件显示为其他交互模式提示。这些区域有：Amazon SAMCLI 提示您为每个参数提供值。有关这些对象和相应提示的示例，请参阅示例本主题后面的部分。

使用代码签名配置的无服务器应用程序会生成更多交互模式提示。这些区域有：Amazon SAMCLI 询问您是否要对代码进行签名，如果是，则提示您输入签名配置文件名称和所有者。有关这些提示的示例，请参阅示例本主题后面的部分。

有关设置的更多信息Amazon SAMCLI 可选择在您指定`--guided`选项，请参阅Amazon SAM CLI 配置文件 (p. 281).

要部署Amazon Lambda函数通过Amazon CloudFormation，则必须包含 Lambda 部署包的 Amazon Simple Storage Service (Amazon S3) 存储桶。这些区域有：Amazon SAMCLI 会为您创建和管理这个 Amazon S3 存储桶。Amazon SAM启用对 Amazon S3 中存储的所有文件进行加密。

如果您的应用程序包含任何用声明的函数或层资源`PackageType: Image`，那么你可以指示Amazon SAMCLI 可自动创建所需的Amazon Elastic Container Registry (Amazon ECR) 存储库 要有Amazon SAMCLI 创建这些存储库，使用`--resolve-image-repos`选项或`--guided`选项，然后使用以下命令回复后续提示**`Y`**.

用法

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

选项：

选项	描述
`-g, --guided`	指定此选项可获得Amazon SAMCLI 使用提示来指导您完成部署。
`-t, --template-file, --template PATH`	您的路径和文件名Amazon SAM模板位于。 注意：如果指定此选项，则Amazon SAM仅部署模板及其指向的本地资源。
`--stack-name TEXT`	（必需）的名称Amazon CloudFormation您要部署到的堆栈。如果指定现有堆栈，则该命令将更新堆栈。如果指定了新堆栈，则该命令将创建它。
`--s3-bucket TEXT`	Amazon S3 存储桶的名称，此命令将您的Amazon CloudFormation 模板。如果你的模板大于 51,200 字节，那么要么`--s3-bucket`选项或`--resolve-s3`选项是必需的。如果指定了两者`--s3-bucket`和`--resolve-s3`选项，则出错。
`--s3-prefix TEXT`	前缀添加到上传到 Amazon S3 存储桶的构件名称中。前缀名称是 Amazon S3 存储桶的路径名（文件夹名称）。
`--image-repository TEXT`	此命令用于上传函数映像的 Amazon ECR 存储库的名称。此选项对于使用声明的函数是必需的`Image`程序包类型。
`--signing-profiles LIST`	用于对部署包进行签名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，可选配置文件所有者分隔为`:`. 例如，`FunctionNameToSign=SigningProfileName1 LayerNameToSign=SigningProfileName2:SigningProfileOwner`。

选项	描述
`--capabilities LIST`	必须指定才能允许的功能列表Amazon CloudFormation创建特定的堆栈。一些堆栈模板中可能包含的资源会影响您的中的权限Amazon Web Services 账户，例如，通过创建新的Amazon Identity and Access Management(IAM) 用户。对于这些堆栈，您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，则操作将返回`InsufficientCapabilities`错误消息。
`--region TEXT`	这些区域有：Amazon Web Services 区域要部署到。例如，us-east-1。
`--profile TEXT`	凭证文件中的特定配置文件将获取Amazon凭证。
`--kms-key-id TEXT`	的身份证Amazon Key Management Service(Amazon KMS) 密钥用于加密 Amazon S3 存储桶中的静态构件。如果不指定此选项，则Amazon SAM使用 Amazon S3 托管加密密钥。
`--force-upload`	指定此选项可上传构件，即使它们与 Amazon S3 存储桶中的现有项目相匹配。匹配的构件将被覆盖。
`--no-execute-changeset`	指示是否应用变更集。如果您想在应用变更集之前查看堆栈更改，请指定此选项。此命令会创建一个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，请在不使用此选项的情况下运行相同的命令。
`--role-arn TEXT`	IAM 角色的 Amazon Resource Name (ARN)，则Amazon CloudFormation在应用变更集时假设。
`--fail-on-empty-changeset | --no-fail-on-empty-changeset`	指定在未对堆栈进行任何更改时是否返回非零退出代码。默认行为是返回非零退出代码。
`--confirm-changeset | --no-confirm-changeset`	提示确认是否Amazon SAMCLI 部署计算出的变更集。
`--use-json`	的输出 JSONAmazon CloudFormation模板。默认输出为 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`--guided`选项，然后是Amazon SAMCLI 忽略`--resolve-s3`. 如果指定了两者`--s3-bucket`和`--resolve-s3`选项，则出错。
`--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以下功能的函数和层`PackageType: Image`指定的。如果您指定`--guided`选项，然后是Amazon SAMCLI 忽略`--resolve-image-repos`. 注意：如果Amazon SAM使用此选项自动为函数或层创建任何 Amazon ECR 存储库，您稍后将这些函数或层从您的Amazon SAM模板，然后自动删除相应的 Amazon ECR 存储库。
`--metadata`	要附加到模板中引用的所有构件的元数据映射。
`--notification-arns LIST`	Amazon Simple Notification Service (Amazon SNS) 主题 ARN 列表Amazon CloudFormation与堆栈关联。
`--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。

选项	描述
`--parameter-overrides`	包含以下内容的字符串Amazon CloudFormation参数覆盖编码为键值对。使用与Amazon Command Line Interface(Amazon CLI)。例如，`ParameterKey=ParameterValue InstanceType=t1.micro`。
`--disable-rollback | --no-disable-rollback`	指定是否回滚您的Amazon CloudFormation如果在部署期间出现错误，则堆栈。默认情况下，如果在部署期间出现错误，您的Amazon CloudFormation堆栈回滚到最后一个稳定状态。如果您指定`--disable-rollback`并且在部署期间出现错误，则不会回滚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`— 将堆栈回滚到以前的已知良好状态。 • `DELETE`— 将堆栈回滚到以前的已知良好状态（如果存在）。否则将删除堆栈。 • `DO_NOTHING`— 既不回滚也不会删除堆栈。效果与的效果相同`--disable-rollback`. 默认行为是 `ROLLBACK`。 注意：您可指定以下任一值`--disable-rollback`选项或`--on-failure`选项，但不能同时兼而有之。
`--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`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. 281)。
`--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 `default`。有关配置文件的详细信息，请参阅 Amazon SAM CLI 配置文件 (p. 281)。
`--no-progressbar`	将构件上传到 Amazon S3 时不要显示进度条。
`--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。

环境变量：

环境变量	描述
`SAM_CLI_POLL_DELAY`	指定延迟，以秒为单位`DescribeStack`API 调用。

示例

参数

以下是声明的示例对象`Parameters`部分，以及使用时出现的相应提示sam deploy --guided.

Amazon SAM模板

```
Parameters:
  MyPar:
```

```
    Type: String
    Default: MyParVal
```

相应的**sam deploy --guided**提示

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

代码签名

以下是使用代码签名配置的示例函数。

Amazon SAM模板

```
Resources:
  HelloWorld:
    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
```

相应的**sam deploy --guided**提示：

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

SAM_CLI_POLL_DELAY

以下是示例sam deploy使用`SAM_CLI_POLL_DELAY`用于设置 5 秒延迟的变量`DescribeStack`API 调用。

```
SAM_CLI_POLL_DELAY=5 sam deploy
```

sam init

使用初始化无服务器应用程序Amazon SAM。模板。该模板为您提供了文件夹结构Amazon Lambda并连接到事件源，例如 API、Amazon Simple Storage Service (Amazon S3) 桶或 Amazon DynamoDB 表等事件源。此应用程序包括入门并最终将其扩展到生产规模应用程序所需的所有内容。

对于某些示例应用程序，您可以选择应用程序的软件包类型，`Zip`要么`Image`. 有关 Lambda 包类型的更多信息，请参阅Lambda 部署程序包中的Amazon Lambda开发人员指南.

使用方法：

```
sam init [OPTIONS]
```

> **Note**
>
> 与Amazon SAM版本 0.30.0 或更高版本，您可以使用以下两种模式之一初始化应用程序：1) 交互式工作流程，或 2) 提供所有必需参数。

- 交互式工作流：通过交互式初始化工作流程，您可以输入 1) 项目名称、首选运行时和模板文件，或 2) 自定义模板的位置。
- 提供参数：提供所有必需参数。

如果您提供了所需参数的子集，系统将提示您输入其他必需信息。

示例：

```
Initializes a new SAM project with required parameters passed as parameters

sam init --runtime python3.7 --dependency-manager pip --app-template hello-world --name
 sam-app

Initializes a new SAM project using custom template in a Git/Mercurial repository

# gh being expanded to github url
sam init --location gh:aws-samples/cookiecutter-aws-sam-python

sam init --location git+ssh://git@github.com/aws-samples/cookiecutter-aws-sam-python.git

sam init --location hg+ssh://hg@bitbucket.org/repo/template-name

# Initializes a new SAM project using custom template in a Zipfile

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

sam init --location https://example.com/path/to/template.zip

# Initializes a new SAM project using cookiecutter template in a local path

sam init --location /path/to/template/folder
```

选项：

选项	描述
`-a, --architecture [x86_64 | arm64]`	应用程序 Lambda 函数的指令集架构。指定其中之一`x86_64`要么`arm64`.
`--app-template TEXT`	要使用的托管应用程序模板的标识符。如果您不能确定，请致电`sam init`没有交互式工作流程的选项。 如果需要此参数：`--no-interactive`已指定并`--location`未提供。 此参数仅在中提供Amazon SAMCLI 版本 0.30.0 及更高版本。使用较早版本指定此参数会导致错误。
`--base-image [amazon/nodejs16.x-base | amazon/nodejs14.x-base | amazon/nodejs12.x-base | amazon/python3.9-base | amazon/python3.8-base | amazon/python3.7-base | amazon/python3.6-base | amazon/ruby2.7-base | amazon/go1.x-base |`	应用程序的基本映像。此选项仅适用于包装类型为`Image`. 如果需要此参数：`--no-interactive`已指定，`--image-type`被指定为`Image`, 和`--location`未指定。

选项	描述
`amazon/java11-base | amazon/java8.al2-base | amazon/java8-base | amazon/dotnet6-base | amazon/dotnet5.0-base | amazon/dotnetcore3.1-base ]`	
`--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. 281)。
`--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. 281)。
`-d, --dependency-manager [gradle | mod | maven | bundler | npm | cli-package | pip]`	Lambda 运行时的依赖关系管理器。
`--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。
`--extra-content`	覆盖模板中的任何自定义参数`cookiecutter.json`例如，配置，`{"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）。 如果需要此参数：`--no-interactive`已指定并`--runtime`、`--name`，和`--app-template`未提供。 对于 Git 仓库，您必须使用仓库根目录的位置。 对于本地路径，模板必须位于 .zip 文件或Cookiecutter格式的日期和时间。
`-n, --name TEXT`	要作为目录生成的项目的名称。 如果需要此参数：`--no-interactive`已指定并`--location`未提供。
`--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，如果缺少任何必需值，则失败。
`--no-tracing`	不要追加`Tracing: Active`转到的全局部分Amazon SAM模板并禁用关于跟踪的交互式提示。有关模板全局部分的更多信息，请参阅的 “全局变量” 部分Amazon SAM模板 (p. 28)
`-o, --output-dir PATH`	输出初始化应用程序的位置。
`--package-type [Zip | Image]`	示例应用程序的软件包类型。`Zip`创建 .zip 文件存档，并`Image`创建容器映像。

选项	描述
`-r, --runtime [ruby2.7 | java8 | java8.al2 | java11 | nodejs12.x | nodejs14.x | nodejs16.x | dotnet6 | dotnet5.0 | dotnetcore3.1 | python3.9 | python3.8 | python3.7 | python3.6 | go1.x]`	应用程序的 Lambda 运行时。此选项仅适用于包装类型为`Zip`. 如果需要此参数：`--no-interactive`已指定，`--image-type`被指定为`Zip`, 和`--location`未指定。
`--tracing`	附加`Tracing: Active`转到的全局部分Amazon SAM。模板。有关模板全局部分的更多信息，请参阅的 “全局变量” 部分Amazon SAM模板 (p. 28)

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. 281)。
`--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. 281)。

选项	描述
`--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

调用本地Amazon Lambda函数一次，然后在调用完成后退出。

默认情况下，当你使用此命令时，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sam build (p. 252)命令，位于`.aws-sam`子文件夹，并命名`template.yaml`要么`template.yml`. 下一步：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，并且只会加载该行为Amazon SAM模板以及它指向的本地资源。

要调用嵌套应用程序或堆栈的函数，可以使用格式提供应用程序或堆栈逻辑 ID 以及函数逻辑 ID`StackLogicalId/FunctionLogicalId`.

这些区域有：`sam local invoke`命令对于开发处理异步事件（如 Amazon Simple Storage Service (Amazon S3) 或 Amazon Kinesis 事件）的无服务器函数非常有用。如果你想编写测试用例的脚本，它也很

有用。你可以使用--event参数。有关事件的更多信息，请参阅 。Event中的Amazon Lambda开发人员指南. 有关不同的事件消息格式的详细信息Amazon服务，请参阅使用其他服务中的Amazon Lambda开发人员指南.

运行时输出（例如，日志）输出到stderr，然后 Lambda 函数结果将输出到stdout.

> Note
>
> 如果在您的中定义了多个函数。Amazon SAM模板，您必须提供FUNCTION_LOGICAL_ID函数您希望调用的函数。

使用方法：

```
sam local invoke [OPTIONS] [FUNCTION_LOGICAL_ID]
```

选项：

选项	描述
-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，将不假定任何事件。要从中输入 JSONstdin，您必须传入值 “-”。有关不同的事件消息格式的详细信息Amazon服务，请参阅使用其他服务中的Amazon Lambda开发人员指南.
--no-event	使用空事件调用函数。
-t, --template PATH	这些区域有：Amazon SAM模板文件。 注意：如果指定此选项，Amazon SAM仅加载模板及其指向的本地资源。
-n, --env-vars PATH	JSON 文件包含 Lambda 函数环境变量的值。有关环境变量文件的更多信息，请参阅环境变量文件 (p. 213).
--parameter-overrides	（可选）包含此项的字符串Amazon CloudFormation参数覆盖编码为键值对。使用的格式与Amazon Command Line Interface(Amazon CLI)。例如：'ParameterKey=KeyPairName、ParameterValue=MyKey ParameterKey=InstanceType、ParameterValue=t1.micro'。
-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	挂载到 Lambda 容器中的调试器的主机路径。
--debug-args TEXT	要传递到调试器的额外参数。
-v, --docker-volume-basedir TEXT	基目录的位置Amazon SAM文件存在。如果 Docker 在远程计算机上运行，则必须将路径挂载到Amazon SAM文件存在于 Docker 计算机上，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 Lambda 函数映像容器。
-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。

选项	描述
`--skip-pull-image`	指定是否Amazon SAMCLI 应跳过拉取 Lambda 运行时的最新 Docker 镜像。
`--force-image-build`	指定是否Amazon SAMCLI 应重建用于调用带图层的 Lambda 函数的映像。
`--invoke-image TEXT`	要用于本地函数调用的容器镜像的 URI。默认情况下，Amazon SAM从 Amazon ECR Public 中提取容器镜像。使用此选项可以从另一个位置拉出图像。 例如：`sam local invoke MyFunction --invoke-image amazon/aws-sam-cli-emulation-image-python3.8`。
`--profile TEXT`	从您的凭证文件获取的具体配置文件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"`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. 281)。
`--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. 281)。
`--shutdown`	在调用完成后模拟关闭事件，以测试关闭行为的扩展处理。
`--container-host TEXT`	本地模拟 Lambda 容器的主机。默认值为 `localhost`。如果你想跑Amazon SAM在 macOS 上 Docker 容器中的 CLI，你可以指定`host.docker.internal`. 如果你想在不同的主机上运行容器Amazon SAMCLI，您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`。使用`0.0.0.0`绑定到所有接口。
`--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。

sam 本地 start-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。当你在包含无服务器函数和你的Amazon SAM模板，它会创建本地 HTTP 服务器，用于托管您的所有函数。

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sam build (p. 252)命令，位于`.aws-sam`子文件夹，并命名`template.yaml`要么`template.yml`. 下一步：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，并且只会加载该行为Amazon SAM模板以及它指向的本地资源。

当它被访问（通过浏览器、CLI 等）时，它会在本地启动 Docker 容器以调用该函数。它读取了的 CodeUri 属性`AWS::Serverless::Function`资源，在您的文件系统中找到包含 Lambda 函数代码的路径。这可以是 Node.js 和 Python 等解释性语言的项目根目录，也可以是存储编译构件或 Java 归档文件的构建目录。

如果使用解释性语言，本地更改在每次调用时都会立即在 Docker 容器中使用。对于更多编译语言或需要复杂包装支持的项目，建议运行您自己的建筑解决方案，Amazon SAM转到包含构建工件的目录或文件。

要查看使用此命令的端到示例，请参阅教程：部署 Hello World 应用程序 (p. 15). 这些区域有：`sam local start-api`命令是的一部分第 4 步：（可选）在本地测试应用程序 (p. 20).

使用方法：

```
sam local start-api [OPTIONS]
```

选项：

选项	描述
`--host TEXT`	要绑定到的本地主机名或 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源（例如，CSS/javasCRIPT/HTML）文件都显示在/。
`-t, --template PATH`	这些区域有：Amazon SAM模板文件。 注意：如果指定此选项，Amazon SAM仅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。
`--parameter-overrides`	可选。包含的字符串Amazon CloudFormation参数覆盖编码为键值对的参数。使用的格式与Amazon CLI— 例如，“参数键 = 密钥对名称，参数值 = myKey 参数键 = 实例类型，参数值 =T1.micro”。
`-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`	将挂载到 Lambda 容器中的调试器的主机路径。
`--debug-args TEXT`	传递到调试器的其他参数。
`--warm-containers [EAGER | LAZY]`	可选。指定如何Amazon SAMCLI 为每个函数管理容器。 有两个选项： `EAGER`：所有函数的容器都在启动时加载，并在调用之间保留。 `LAZY`：只有在首次调用每个函数时才加载容器。这些容器会持续进行额外的调用。
`--debug-function`	可选。指定要将调试选项应用于何时的 Lambda 函数`--warm-containers`已指定。此参数适用于`--debug-port`、`--debugger-path`, 和`--debug-args`.
`-v, --docker-volume-basedir TEXT`	基目录的位置Amazon SAM文件存在。如果 Docker 在远程计算机上运行，则必须将路径挂载到Amazon SAM文件存在于 Docker 计算机上，然后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，以及默认桥接网络。如果未指定此操作，Lambda 容器将仅连接到默认桥接 Docker 网络。
`--container-env-vars`	可选。在本地调试时将环境变量传递给映像容器。
`-l, --log-file TEXT`	要向其发送运行时日志的日志文件。
`--layer-cache-basedir DIRECTORY`	指定您的模板使用的图层下载到的基准位置。

选项	描述
`--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 是否应重建用于使用层调用函数的映像。
`--invoke-image TEXT`	您要用于 Lambda 函数的容器映像的 URI。默认情况下，Amazon SAM从 Amazon ECR Public 中提取容器镜像。使用此选项可以从另一个位置拉出图像。 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。如果指定字符串，则它是用于应用程序中所有函数的容器镜像的 URI。例如：`sam local start-api --invoke-image public.ecr.aws/sam/emu-python3.8`。如果指定键值对，则键为资源名称，值是要用于该资源的容器镜像的 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`	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`	配置文件的路径和文件名，该文件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. 281)。
`--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. 281)。
`--shutdown`	在调用完成后模拟关闭事件，以测试关闭行为的扩展处理。
`--container-host TEXT`	本地模拟 Lambda 容器的主机。默认值为 `localhost`。如果你想跑Amazon SAM在 macOS 上 Docker 容器中的 CLI，你可以指定`host.docker.internal`. 如果你想在不同的主机上运行容器Amazon SAMCLI，您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`。使用`0.0.0.0`绑定到所有接口。
`--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 本地 start-lambda

使您能够通过使用Amazon CLI或开发工具包。此命令启动模拟的本地终端节点Amazon Lambda.

默认情况下，当你使用此命令时，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sam build (p. 252)命令，位于`.aws-sam`子文件夹，并命名`template.yaml`要么`template.yml`. 下一步：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，并且只会加载该行为Amazon SAM模板以及它指向的本地资源。

您可以针对此本地 Lambda 终端节点运行自动测试。当您使用向此终端节点发送调用时Amazon CLI或者开发工具包，它会在本地执行请求中指定的 Lambda 函数。

使用方法：

```
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
`--host TEXT`	要绑定到的本地主机名或 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`	这些区域有：Amazon SAM模板文件。 注意：如果指定此选项，Amazon SAM仅加载模板及其指向的本地资源。
`-n, --env-vars PATH`	JSON 文件，该文件包含 Lambda 函数环境变量的值。
`--parameter-overrides`	可选。包含的字符串Amazon CloudFormation参数覆盖编码为键值对的。使用的格式与Amazon CLI— 例如，“参数键 = 密钥对名称，参数值 = myKey 参数键 = 实例类型，参数值 =T1.micro”。
`-d, --debug-port TEXT`	指定后，以调试模式启动 Lambda 函数容器，然后在本地主机上公开此端口。
`--debugger-path TEXT`	要挂载到 Lambda 容器中的调试器的主机路径。
`--debug-args TEXT`	传递到调试器的其他参数。
`--warm-containers [EAGER | LAZY]`	可选。指定如何Amazon SAMCLI 为每个函数管理容器。 有两个选项可供选择：

选项	描述
	`EAGER`：所有函数的容器都在启动时加载，并在调用之间保留。 `LAZY`：只有在首次调用每个函数时才加载容器。这些容器会持续存在以进行额外的调用。
`--debug-function`	可选。指定要将调试选项应用于何时的 Lambda 函数`--warm-containers`已指定。此参数适用于`--debug-port`、`--debugger-path`, 和`--debug-args`.
`-v, --docker-volume-basedir TEXT`	基目录的位置Amazon SAM文件存在。如果 Docker 在远程计算机上运行，则必须将路径挂载到Amazon SAM文件存在于 Docker 计算机上，然后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，以及默认桥接网络。如果指定此项，Lambda 容器将仅连接到默认桥接 Docker 网络。
`--container-env-vars`	可选。在本地调试时将环境变量传递给映像容器。
`-l, --log-file TEXT`	要向其发送运行时日志的日志文件。
`--layer-cache-basedir DIRECTORY`	指定您的模板使用的图层下载到的 base ir 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，Amazon SAM从 Amazon ECR Public 中提取容器镜像。使用此选项可以从另一个位置拉出图像。 例如：`sam local start-lambda MyFunction --invoke-image amazon/aws-sam-cli-emulation-image-python3.8`。
`--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图层的函数的映像。
`--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`	配置文件的路径和文件名，以及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. 281)。
`--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. 281)。
`--shutdown`	在调用完成后模拟关闭事件，以测试关闭行为的扩展处理。
`--container-host TEXT`	本地模拟 Lambda 容器的主机。默认值为 `localhost`。如果你想跑Amazon SAM在 macOS 上 Docker 容器中的 CLI，你可以指定`host.docker.internal`. 如果你想在不同的主机上运行容器Amazon SAMCLI，您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`。使用`0.0.0.0`绑定到所有接口。
`--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 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. 281)。
`--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. 281)。
`--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sam build (p. 252)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. 257)现在隐式执行的功能`sam package`. 您可以使用sam deploy (p. 257)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. 281)。
`--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. 281)。
`--no-progressbar`	将工件上传到 Amazon S3 时，请勿显示进度条。

选项	描述
`--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，请参阅使用发布无服务器应用程序Amazon SAMCLI (p. 232).

sam 管道引导

此命令会生成必需的Amazon用于连接 CI/CD 系统的基础设施资源。必须先为管道中的每个部署阶段运行此步骤，然后才能运行sam pipeline init命令。

此命令将设置以下内容：Amazon基础设施资源：

- 具有要与 CI/CD 系统共享的访问密钥 ID 和私有密钥访问凭证的管道 IAM 用户。
- 管道用户担任的管道执行 IAM 角色，以获取对Amazonaccount.
- 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 CloudFormation担任的执行 IAM 角色Amazon CloudFormation部署Amazon SAM应用程序.
- 一个 Amazon S3 存储桶，用于存储Amazon SAM构件。
- （可选）一个用于存放容器映像 Lambda 部署包的 Amazon ECR 映像存储库（如果您有包类型的资源）`Image`。

使用方法：

```
sam pipeline bootstrap [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 “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. 281)。
`--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. 281)。
`--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。对于这个命令`--stage`是唯一必需的参数。
`--stage TEXT`	相应部署阶段的名称。它被用作创建的后缀Amazon基础设施资源。
`--pipeline-user TEXT`	IAM 用户的 Amazon 资源名称 (ARN)，其访问密钥 ID 和私有访问密钥与 CI/CD 系统共享。它用于授予此 IAM 用户访问相应Amazonaccount. 如果未提供此命令，将创建访问密钥 ID 和秘密访问密钥凭证。
`--pipeline-execution-role TEXT`	此阶段将由管道用户担任的 IAM 角色的 ARN。只有在你想使用自己的角色时才提供它，否则此命令将创建一个角色。

选项	描述
`--cloudformation-execution-role TEXT`	由该项目担任的 IAM 角色的 ARNAmazon CloudFormation服务，同时部署应用程序的堆栈。仅当你想使用自己的角色时才提供，否则该命令将创建一个角色。
`--bucket TEXT`	ARN 于 Amazon S3Amazon SAM构件。
`--create-image-repository / --no-create-image-repository`	指定是否创建 Amazon ECR 映像存储库（如果未提供）。Amazon ECR 存储库保存 Lambda 函数或层的容器映像，其包类型为`Image`. 默认为 `--no-create-image-repository`。
`--image-repository TEXT`	亚马逊 ECR 映像存储库的 ARN，用于存放包类型为的 Lambda 函数或图层的容器映像`Image`. 如果提供，则`--create-image-repository`忽略选项。如果没有提供和`--create-image-repository`被指定，该命令将创建一个。
`--confirm-changeset / --no-confirm-changeset`	提示确认是否要部署资源。
`--profile TEXT`	获取凭证文件中的特定配置文件Amazon凭证。
`--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。
`--region TEXT`	这些区域有：Amazon部署到的区域。例如：`us-east-1`。
`-h, --help`	显示此消息并退出。

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. 281)。
`--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. 281)。
`--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，请参阅使用发布无服务器应用程序Amazon SAMCLI (p. 232).

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. 281)。
`--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. 281)。
`--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。

sam 同步

这些区域有：Amazon SAMCLIsync命令将您的本地更改部署到Amazon Web Services 云. 使用`sync`在迭代应用程序时构建、打包和部署对开发环境的更改。

用法：

```
sam sync [OPTIONS]
```

选项：

选项	描述
`-t, --template-file, --template PATH`	您的路径和文件名Amazon SAM模板位于。 注意：如果指定此选项，则Amazon SAM仅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： • `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，Amazon SAM使用Amazon直接提供服务 API，而不是通过部署Amazon CloudFormation. 更新您的Amazon CloudFormation堆栈，运行sam sync --watch要么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，Amazon SAM更新应用程序中的所有资源时对其进行同步。有了该选项，Amazon SAM执行初始操作Amazon CloudFormation部署。然后，Amazon SAM使用Amazon用于更新代码资源的服务 API。Amazon SAM使用Amazon CloudFormation在更新您的基础架构资源时更新基础架构资源Amazon SAM模板。
`--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。此选项受支持`--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。此选项受支持`--code`选项。该值必须是下面列出的资源之一`--code`. 例如，`--resource AWS::Serverless::Function --resouce AWS::Serverless::LayerVersion`。
`--dependency-layer | --no-dependency-layer`	指定是否将各个函数的依赖关系分离到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本机编译依赖项的包，请使用此选项在Amazon Lambda像 Docker 容器一样。

选项	描述
	注意：目前，此选项与不兼容`--dependency-layer`. 如果您将`--use-container`和`--dependency-layer`，Amazon SAMCLI 会通知您并继续`--no-dependency-layer`.
`--stack-name TEXT`	（必需）的名称Amazon CloudFormation为您的应用程序堆栈。
`--s3-bucket TEXT`	Amazon Simple Storage Service (Amazon S3) 存储桶的名称Amazon CloudFormation模板。如果你的模板大于 51,200 字节，那么要么`--s3-bucket`或者`--resolve-s3`选项是必需的。如果指定了两者`--s3-bucket`和`--resolve-s3`选项，则将出错。
`--s3-prefix TEXT`	前缀添加到您上传到 Amazon S3 存储桶的构件名称中。前缀名称是 Amazon S3 存储桶的路径名（文件夹名称）。这仅适用于使用声明的函数`Zip`The 程序包。
`--capabilities LIST`	您指定允许的功能列表Amazon CloudFormation创建特定的堆栈。一些堆栈模板中可能包含的资源会影响您的中的权限Amazon Web Services 账户.例如，通过创建新的Amazon Identity and Access Management(IAM) 用户。默认功能是`CAPABILITY_NAMED_IAM`和`CAPABILITY_AUTO_EXPAND`. 指定此选项以覆盖默认值。有效值包括： • CAPABITY_IAM • CAPABILITY_NAMED_IAM • 能力_资源_政策 • CAPABILITY_AUTO_EXPAND
`-s, --base-dir DIRECTORY`	解析函数或层相对于此目录的源代码的相对路径。使用此选项更改源代码文件夹相对路径的解析方式。默认情况下，相对路径是相对路径解析的 Amazon SAM模板的位置。 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`	包含以下内容的字符串Amazon CloudFormation参数覆盖。编码为键值对的参数。使用与Amazon Command Line Interface(Amazon CLI)。例如，`ParameterKey=ParameterValue InstanceType=t1.micro`。
`--image-repository TEXT`	Amazon EElastic Container Registry (Amazon ECR) 存储库的名称。此命令将上载函数镜像。使用声明的函数是必需的`Image`The 程序包。
`--kms-key-id TEXT`	的身份证Amazon Key Management Service(Amazon KMS) 密钥用于加密 Amazon S3 存储桶中的静态构件。如果不指定此选项，则Amazon SAM使用 Amaon S3 托管加密密钥。
`--role-arn TEXT`	IAM 角色的 Amazon 资源名称 (ARN)，Amazon CloudFormation在应用变更集时假设。
`--notification-arns LIST`	Amazon Simple Notification Service (Amazon S3) 主题 ARN 列表Amazon CloudFormation与堆栈关联。

选项	描述
`--tags LIST`	要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该标签的资源。
`--metadata`	元数据映射，用于附加到您在模板中引用的所有工件。

示例

运行以下命令以启动一个进程，该进程将本地环境中的更改自动部署到开发环境中Amazon Web Services云.

```
sam sync --stack-name sam-app --watch
```

运行以下命令将代码更改部署到特定的 Lambda 函数和 Lambda 层。Amazon SAM使用 Lambda API 在中更新您的代码Amazon Web Services 云.

```
sam sync --stack-name sam-app --code --resource-id HelloWorldFunction --resource-id
 HelloWorldLayer
```

运行以下命令以将最新的本地更改部署到应用程序的Amazon CloudFormation堆栈。

```
sam sync --stack-name sam-app
```

嵌套堆栈示例

> Note
>
> 我们目前支持使用以下两种方法引用堆栈：`!Ref LayerName`要么`!GetAttr Stack.Output.LayerName`.

Amazon SAM加速现在支持嵌套堆栈。要同步子堆栈中的资源，请继续为其提供根堆栈`--stack-name`参数。对于`--resource-id`参数，按以下格式指定子堆栈和资源*`ChildStack/resourceId`*. 您可以定义其他子堆栈，方法是随后使用以下方式描述堆栈名称`/`. 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. 190).

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

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

验证是否Amazon SAM模板文件是否有效。

使用方法：

```
sam validate [OPTIONS]
```

选项：

选项	描述
`-t, --template, --template-file PATH`	这些区域有：Amazon SAM模板文件 [默认值：模板。 [yaml|yml]]。
`--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`	配置文件对应的路径和文件名，其中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. 281)。
`--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. 281)。
`--debug`	打开调试日志记录以打印由Amazon SAMCLI 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. 283)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. 252)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 CLI`sam 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。

Amazon SAM策略模板

Amazon SAM允许您从策略模板列表中选择来将您的 Lambda 函数的权限范围限定为您的应用程序使用的资源。

Amazon SAM中的应用程序Amazon Serverless Application Repository使用策略模板不需要任何特殊的客户确认，即可从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 policy_templates.json 源文件提交拉取请求`develop`的分支Amazon SAM GitHub 项目。源文件在策略模板.json在 GitHub 网站。
2. 在中提交问题Amazon SAM GitHub 项目，其中包括拉取请求的理由和至请求的链接。使用此链接提交新问题：Amazon Serverless Application Model：问题.

语法

对于您在中指定的每个策略模板Amazon SAM模板文件时，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. 289) 策略模板期待 `QueueName` 作为资源。这些区域有：Amazon SAM 模板检索" 的名称`MyQueue`“Amazon SQS 队列，该队列可以在同一应用程序中创建或作为应用程序的参数请求。

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

示例 2：策略模板不具有占位符值

以下示例包含 CloudWatchPutMetric策略 (p. 290) 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，否则将导致错误。

```
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. 312)	授予从中读取证书的权限Amazon Certificate Manager.		
AMIDescribePolicy (p. 313)	授予描述Amazon 系统映像 (AMI) 的权限。		
AthenaQuery策略 (p. 318)	授予执行 Athena 查询的权限。		
AWSSecretsManagerGetSecretValuePolicy (p. 319)	授予获取指定的 secret 值的权限Amazon Secrets Manager密钥。		
AWSSecretsManagerRotationPolicy (p. 320)	授予在中轮换密钥的权限Amazon Secrets Manager.		

策略模板	描述		
CloudFormationDescribeStacksPolicy 策略 (p. 297)	授予描述权限Amazon CloudFormation堆栈。		
CloudWatchDashboardPolicy (p. 287)	授予对指标进行操作的权限 CloudWatch控制面板。		
CloudWatchDescribeAlarmHistoryPolicy (p. 289)	授予描述权限 CloudWatch 告警历史记录。		
CloudWatchPutMetricPolicy 策略 (p. 290)	授予权限以将指标发送到 CloudWatch.		
CodeCommitCrudPolicy (p. 291)	授予创建/读取/更新/删除特定对象的权限 CodeCommit 存储库。		
CodeCommitReadPolicy (p. 293)	授予读取特定对象的权限 CodeCommit存储库。		
CodePipelineLambdaExecutionPolicy 策略 (p. 305)	授予由调用的 Lambda 函数的权限 CodePipeline 以报告任务的状态。		
CodePipelineReadOnlyPolicy 策略 (p. 307)	授予对获取有关详细信息的读取权限 CodePipeline 管道。		
ComprehendBasicAccessPolicy (p. 312)	授予检测实体、关键短语、语言和情绪的权限。		
CostExplorerReadOnlyPolicy 策略 (p. 313)	为账单历史记录的只读Cost Explorer API 授予只读权限。		
DynamoDBBackupFullAccessPolicy (p. 201)	授予对表进行 DynamoDB 按需备份的读写权限。		
DynamoDBCrudPolicy (p. 201)	授予对 Amazon DynamoDB 表的创建、读取、更新和删除权限。		
DynamoDBReadPolicy (p. 202)	授予对 DynamoDB 表的只读权限。		
DynamoDBReconfigurePolicy (p. 202)	授予重新配置 DynamoDB 表的权限。		
DynamoDBRestoreFromBackupPolicy (p. 203)	授予从备份还原 DynamoDB 表的权限。		
DynamoDBStreamReadPolicy 策略 (p. 300)	授予描述和读取 DynamoDB 流和记录的权限。		
DynamoDBWritePolicy (p. 203)	授予对 DynamoDB 表的只写权限。		
EC2CopyImage 策略 (p. 306)	授予复制 Amazon EC2 映像的权限。		
EC2DescribePolicy (p. 324)	授予描述Amazon EElastic Compute Cloud (Amazon EC2)		
EcsRunTaskPolicy (p. 324)	授予权限以启动任务定义的新任务。		
EFSWriteAccess 策略 (p. 324)	授予挂载具有写入权限的 Amazon EFS 文件系统的权限。		
EKSDescribePolicy (p. 325)	授予描述或列出 Amazon EKS 集群的权限。		

策略模板	描述		
ElasticMapReduceAddJobFlowSteps策略 (p. 320)	授予权限以将新步骤添加到运行的集群中。		
ElasticMapReduceCancelSteps策略 (p. 320)	授予权限以取消运行的集群中的一个或多个待处理步骤。		
ElasticMapReduceModifyInstanceFleet策略 (p. 320)	授予列出集群中实例队列详细信息和修改容量的权限。		
ElasticMapReduceModifyInstanceGroups策略 (p. 321)	授予列出集群中实例组的详细信息和修改设置的权限。		
ElasticMapReduceSetTerminationProtection策略 (p. 321)	授予权限以设置集群的终止保护。		
ElasticMapReduceTerminateJobFlows策略 (p. 322)	授予以下权限以关闭集群的权限。		
ElasticsearchHttpPostPolicy (p. 294)	向亚马逊授予 POST 权限 OpenSearch 服务 。		
EventBridgePutEventsPolicy策略 (p. 320)	授予将事件发送到的权限 EventBridge.		
FilterLogEventsPolicy (p. 294)	授予筛选权限 CloudWatch 记录指定日志组的事件。		
FirehoseCrud策略 (p. 312)	授予创建、写入、更新和删除 Kinesis Data Firehose 传输流的权限。		
FirehoseWrite策略 (p. 312)	授予写入到 Kinesis Data Firehose 传输流的权限。		
KinesisCrud策略 (p. 302)	授予创建、发布和删除 Amazon Kinesis 直播的权限。		
KinesisStreamReadOnlyPolicy (p. 303)	授予列出并阅读 Amazon Kinesis 流的权限。		
自杀DecryptPolicy (p. 303)	授予使用解密的权限Amazon Key Management Service(Amazon KMS) 键。		
自杀EncryptPolicy (p. 303)	授予使用加密的权限Amazon Key Management Service(Amazon KMS) 键。		
LambdaInvoke策略 (p. 290)	授予调用Amazon Lambda函数、别名或版本。		
MobileAnalyticsWriteOnlyAccessPolicy (p. 295)	授予对所有应用程序资源放置事件数据的只写权限。		
OrganizationsListAccountsPolicy (p. 295)	授予列出子账户名称和 ID 的只读权限。		
PinpointEndpointAccessPolicy (p. 314)	授予为 Amazon Pinpoint 应用程序获取并更新终端节点的权限。		
PollyFullAccessPolicy (p. 304)	授予对 Amazon Polly 词典资源的完全访问权限。		
RekognitionDetectOnlyPolicy (p. 295)	授予检测面部、标签和文本的权限。		
RekognitionFacesManagementPolicy (p. 308)	授予在 Amazon Rekognition 系列中添加、删除和搜索人脸的权限。		

策略模板	描述		
RekognitionFaces策略 (p. 309)	授予比较并检测面部和标签的权限。		
RekognitionLabels策略 (p. 309)	授予检测对象和审核标签的权限。		
RekognitionNoDataAccess策略 (p. 297)	授予比较并检测面部和标签的权限。		
RekognitionRead策略 (p. 298)	授予列出和搜索人脸的权限。		
RekognitionWriteOnlyAccess策略 (p. 298)	授予创建集合和索引人脸的权限。		
Route53ChangeResourceRecordSets策略 (p. 325)	授予更改 Route 53 中的资源记录集的权限。		
S3CrudPolicy (p. 299)	授予创建、读取、更新和删除权限，以便对 Amazon S3 存储桶中的数据元执行操作。		
S3FullAccess策略 (p. 304)	授予对 Amazon S3 存储桶中的对象执行操作的完全访问权限。		
S3ReadPolicy (p. 299)	授予对 Amazon Simple Storage Service (Amazon S3) 存储桶中的只读权限。		
S3WritePolicy (p. 299)	授予写入到 Amazon S3 存储桶的写权限。		
SageMakerCreateEndpointConfigPolicy (p. 323)	授予权限以在 SageMaker.		
SageMakerCreateEndpointPolicy策略 (p. 323)	授予权限以在 SageMaker.		
ServerlessRepoReadWriteAccessPolicy (p. 304)	授予在 Amazon Serverless Application Repository服务。		
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。		
SESCrudPolicy (p. 314)	授予发送电子邮件和验证身份的权限。		
SESEmailTemplateCrudPolicy (p. 314)	授予创建、获取、列出、更新和删除 Amazon SES 电子邮件模板的权限。		
SESSendBounce策略 (p. 294)	给 SendBounce 授予对Amazon SSimple Email Service (Amazon SES)		
SNSCrudPolicy (p. 299)	授予创建、发布和订阅 Amazon SNS 主题的权限。		
SNSPublishMessagePolicy策略 (p. 299)	授予权限以发布消息到 Amazon Simple Notification Service (Amazon SNS) 主题。		
SQSPollerPolicy (p. 299)	授予轮询Amazon Simple Queueue Service (Amazon SQS		
SQSSendMessage策略 (p. 299)	授予将消息发送到 Amazon SQS 队列的权限。		

策略模板	描述		
SSMParameterReadPolicy 策略 (p. 315)	授予访问来自 Amazon EC2 Systems Manager (SSM) 参数存储的参数的权限，以便在此账户中加载密钥。		
StepFunctionsExecutionPolicy (p. 315)	授予开始执行 Step Functions 状态机的权限。		
TextractDetectAnalyzePolicy (p. 316)	允许使用 Amazon Textract 检测和分析文档。		
TextractGetResultPolicy (p. 322)	允许从 Amazon Textract 获取检测到和分析的文档。		
TextractPolicy (p. 323)	授予对Amazon Textract ect 的完全访问权限		
VPCAccessPolicy (p. 324)	授予创建、删除、描述和分离弹性网络接口的权限。		

问题排查

SAM CLI 错误：“必须为策略模板指定有效的参数值 '<policy-template-name>'”

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，请声明该策略，如以下示例所示CloudWatchPutMetric策略 (p. 290).

```
MyFunction:
  Policies:
    - CloudWatchPutMetricPolicy: {}
```

Policy templates 列表

以下是可用的策略模板以及应用于每个策略模板的权限。Amazon Serverless Application Model(Amazon SAM) 将自动填充占位符项目 (例如Amazon包含适当的信息的区域和账户 ID。

SQSPollerPolicy

授予对 Amazon Simple Queue Service (Amazon SQS) 队列进行轮询的权限。

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

LambdaInvoke策略

授予调用Amazon Lambda函数、别名或版本。

```
      "Statement": [
        {
          "Effect": "Allow",
          "Action": [
            "lambda:InvokeFunction"
          ],
          "Resource": {
            "Fn::Sub": [
              "arn:${AWS::Partition}:lambda:${AWS::Region}:${AWS::AccountId}:function:
${functionName}*",
              {
                "functionName": {
                  "Ref": "FunctionName"
                }
              }
            ]
          }
        }
      ]
```

CloudWatchDescribeAlarmHistoryPolicy

授予描述 Amazon 的权限CloudWatch警报历史记录。

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

CloudWatchPutMetric策略

授予将指标发送到CloudWatch.

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

EC2DescribePolicy

授予描述 Amazon Elastic Compute Cloud (Amazon EC2) 实例的权限。

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

DynamoDBCrudPolicy

授予对 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

授予对 DynamoDB 表的只读权限。

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

DynamoDBWritePolicy

授予对 DynamoDB 表的只写权限。

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

DynamoDBReconfigurePolicy

授予重新配置 DynamoDB 表的权限。

```
        "Statement": [
          {
            "Effect": "Allow",
            "Action": [
              "dynamodb:UpdateTable"
            ],
            "Resource": {
              "Fn::Sub": [
                "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
${tableName}",
                {
                  "tableName": {
                    "Ref": "TableName"
                  }
                }
              ]
            }
          }
        ]
```

SESSendBounce策略

赋予SendBounce授予对 Amazon Simple Email Service (Amazon SES) 身份的权限。

```
        "Statement": [
          {
            "Effect": "Allow",
            "Action": [
              "ses:SendBounce"
            ],
            "Resource": {
              "Fn::Sub": [
                "arn:${AWS::Partition}:ses:${AWS::Region}:${AWS::AccountId}:identity/
${identityName}",
                {
                  "identityName": {
                    "Ref": "IdentityName"
                  }
                }
              ]
            }
          }
        ]
```

ElasticsearchHttpPostPolicy

向亚马逊授予 POST 和 PUT 权限OpenSearch服务 。

```
        "Statement": [
          {
            "Effect": "Allow",
            "Action": [
              "es:ESHttpPost",
              "es:ESHttpPut"
            ],
            "Resource": {
              "Fn::Sub": [
                "arn:${AWS::Partition}:es:${AWS::Region}:${AWS::AccountId}:domain/
${domainName}/*",
                {
                  "domainName": {
                    "Ref": "DomainName"
                  }
                }
              ]
            }
          }
        ]
```

S3ReadPolicy

授予读取 Amazon Simple Storage Service (Amazon S3) 存储桶中的对象的只读权限。

```
        "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

授予写入 Amazon S3 存储桶中的对象写入 Amazon S3 存储桶的权限。

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

```
                "Ref": "BucketName"
              }
            }
          ]
        }
      ]
    }
  ]
```

S3CrudPolicy

授予创建、读取、更新和删除权限，以便对 Amazon S3 存储桶中的对象执行操作。

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

AMIDescribePolicy

授予描述 Amazon 系统映像 (AMI) 的权限。

```
"Statement": [
```

```
          {
            "Effect": "Allow",
            "Action": [
              "ec2:DescribeImages"
            ],
            "Resource": "*"
          }
        ]
```

CloudFormationDescribeStacks策略

允许描述Amazon CloudFormation堆栈。

```
        "Statement": [
          {
            "Effect": "Allow",
            "Action": [
              "cloudformation:DescribeStacks"
            ],
            "Resource": {
              "Fn::Sub": "arn:${AWS::Partition}:cloudformation:${AWS::Region}:
${AWS::AccountId}:stack/*"
            }
          }
        ]
```

RekognitionDetectOnlyPolicy

授予检测面部、标签和文本的权限。

```
        "Statement": [
          {
            "Effect": "Allow",
            "Action": [
              "rekognition:DetectFaces",
              "rekognition:DetectLabels",
              "rekognition:DetectModerationLabels",
              "rekognition:DetectText"
            ],
            "Resource": "*"
          }
        ]
```

RekognitionNoDataAccess策略

授予比较并检测面部和标签的权限。

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

RekognitionRead策略

允许列出和搜索人脸。

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

RekognitionWriteOnlyAccess策略

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"
                  }
                }
              ]
            }
          }
        ]
```

SQSSendMessage策略

授予将消息发送到 Amazon SQS 队列的权限。

```
        "Statement": [
          {
            "Effect": "Allow",
            "Action": [
              "sqs:SendMessage*"
            ],
            "Resource": {
              "Fn::Sub": [
                "arn:${AWS::Partition}:sqs:${AWS::Region}:${AWS::AccountId}:${queueName}",
                {
                  "queueName": {
                    "Ref": "QueueName"
                  }
                }
              ]
            }
          }
        ]
```

SNSPublishMessage策略

授予将消息发布到 Amazon Simple Notification Service (Amazon SNS) 主题的权限。

```
        "Statement": [
          {
            "Effect": "Allow",
            "Action": [
              "sns:Publish"
            ],
            "Resource": {
              "Fn::Sub": [
                "arn:${AWS::Partition}:sns:${AWS::Region}:${AWS::AccountId}:${topicName}",
                {
                  "topicName": {
                    "Ref": "TopicName"
                  }
                }
              ]
            }
          }
        ]
```

VPCAccessPolicy

授予创建、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": "*"
          }
        ]
```

DynamoDBStreamRead策略

允许描述和读取 DynamoDB 流和记录。

```
        "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"
                  }
```

```
              }
            ]
          }
        }
      ]
```

KinesisStreamReadPolicy

授予发布并阅读 Amazon Kinesis 流的权限。

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

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"
                  }
                }
              ]
            }
          }
        ]
```

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"
                  }
                }
              ]
            }
          }
        ]
```

KinesisCrud策略

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"
                  }
                }
              ]
            }
          }
        ]
```

自杀DecryptPolicy

授予使用解密的权限Amazon Key Management Service(Amazon KMS) 键。请注意`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"
                  }
                }
              ]
            }
          }
        ]
```

自杀EncryptPolicy

授予使用加密的权限Amazon KMS键。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"
                  }
                }
```

```
            ]
          }
        }
      ]
```

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/*"
            }
          ]
        }
      ]
```

S3FullAccess策略

授予对 Amazon S3 存储桶中的对象采取行动的完全访问权限。

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

CodePipelineLambdaExecution策略

授予对由调用的 Lambda 函数的权限Amazon CodePipeline以报告任务的状态。

```
      "Statement": [
        {
          "Effect": "Allow",
          "Action": [
            "codepipeline:PutJobSuccessResult",
            "codepipeline:PutJobFailureResult"
          ],
          "Resource": "*"
        }
      ]
```

ServerlessRepoReadWriteAccessPolicy

授予在中创建并列出应用程序的权限Amazon Serverless Application Repository(Amazon SAM) 服务。

```
        "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/*"
              }
            ]
          }
        ]
```

EC2CopyImage策略

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"
                  }
                }
              ]
            }
          }
        ]
```

AWSSecretsManagerRotationPolicy

允许在中轮换秘密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": "*"
          }
        ]
```

AWSSecretsManagerGetSecretValuePolicy

授予获取指定的密钥值的权限Amazon Secrets Manager密钥。

```
        "Statement": [
          {
            "Effect": "Allow",
            "Action": [
              "secretsmanager:GetSecretValue"
            ],
            "Resource": {
              "Fn::Sub": [
                "${secretArn}",
                {
                  "secretArn": {
                    "Ref": "SecretArn"
                  }
                }
              ]
            }
          }
        ]
```

CodePipelineReadOnly策略

授予读取权限以获取有关CodePipeline管道。

```
        "Statement": [
          {
            "Effect": "Allow",
            "Action": [
              "codepipeline:ListPipelineExecutions"
            ],
            "Resource": {
              "Fn::Sub": [
                "arn:${AWS::Partition}:codepipeline:${AWS::Region}:${AWS::AccountId}:
${pipelinename}",
                {
                  "pipelinename": {
                    "Ref": "PipelineName"
                  }
                }
              ]
            }
          }
        ]
```

CloudWatchDashboardPolicy

授予使用指标进行操作的权限CloudWatch控制面板。

```
        "Statement": [
          {
            "Effect": "Allow",
            "Action": [
              "cloudwatch:GetDashboard",
              "cloudwatch:ListDashboards",
              "cloudwatch:PutDashboard",
              "cloudwatch:ListMetrics"
            ],
            "Resource": "*"
          }
        ]
```

RekognitionFacesManagementPolicy

授予在 Amazon Rekognition 收藏中添加、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"
              }
            }
          ]
        }
      }]
```

RekognitionFaces策略

授予比较并检测面部和标签的权限。

```
      "Statement": [{
        "Effect": "Allow",
        "Action": [
          "rekognition:CompareFaces",
          "rekognition:DetectFaces"
        ],
        "Resource": "*"
        }
      ]
```

RekognitionLabels策略

授予检测对象和审核标签的权限。

```
      "Statement": [{
        "Effect": "Allow",
        "Action": [
          "rekognition:DetectLabels",
          "rekognition:DetectModerationLabels"
        ],
        "Resource": "*"
       }
      ]
```

DynamoDBBackupFullAccessPolicy

授予对表的 DynamoDB 按需备份的读写权限。

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

DynamoDBRestoreFromBackupPolicy

授予从备份中还原 DynamoDB 表的权限。

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

ComprehendBasicAccessPolicy

授予检测实体、关键短语、语言和情绪的权限。

```
"Statement": [{
  "Effect": "Allow",
  "Action": [
    "comprehend:BatchDetectKeyPhrases",
    "comprehend:DetectDominantLanguage",
    "comprehend:DetectEntities",
    "comprehend:BatchDetectEntities",
    "comprehend:DetectKeyPhrases",
    "comprehend:DetectSentiment",
    "comprehend:BatchDetectDominantLanguage",
    "comprehend:BatchDetectSentiment"
  ],
  "Resource": "*"
  }
]
```

MobileAnalyticsWriteOnlyAccessPolicy

授予对所有应用程序资源放置事件数据的只写权限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "mobileanalytics:PutEvents"
    ],
    "Resource": "*"
  }
]
```

PinpointEndpointAccessPolicy

授予获取并更新 Amazon Pinpoint 应用程序的终端节点的权限。

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

FirehoseWrite策略

授予写入 Kinesis Data Firehose 传输流的权限。

```
        "Statement": [
          {
            "Effect": "Allow",
            "Action": [
              "firehose:PutRecord",
              "firehose:PutRecordBatch"
            ],
            "Resource": {
              "Fn::Sub": [
                "arn:${AWS::Partition}:firehose:${AWS::Region}:
${AWS::AccountId}:deliverystream/${deliveryStreamName}",
                {
                  "deliveryStreamName": {
                    "Ref": "DeliveryStreamName"
                  }
                }
              ]
            }
          }
        ]
```

FirehoseCrud策略

授予创建、写入、更新和删除 Kinesis Data Firehose 传输流的权限。

```
        "Statement": [
          {
            "Effect": "Allow",
            "Action": [
              "firehose:CreateDeliveryStream",
              "firehose:DeleteDeliveryStream",
              "firehose:DescribeDeliveryStream",
              "firehose:PutRecord",
              "firehose:PutRecordBatch",
              "firehose:UpdateDestination"
            ],
            "Resource": {
              "Fn::Sub": [
```

```
                "arn:${AWS::Partition}:firehose:${AWS::Region}:
${AWS::AccountId}:deliverystream/${deliveryStreamName}",
                {
                  "deliveryStreamName": {
                    "Ref": "DeliveryStreamName"
                  }
                }
              ]
            }
          }
        ]
```

EKSDescribePolicy

授予描述或列出 Amazon Elastic Kubernetes Service (Amazon EKS) 集群的权限。

```
        "Statement": [
          {
            "Effect": "Allow",
            "Action": [
              "eks:DescribeCluster",
              "eks:ListClusters"
            ],
            "Resource": "*"
          }
        ]
```

CostExplorerReadOnly策略

授予只读权限Amazon Cost Explorer（Cost Explorer）账单历史记录的 API。

```
        "Statement": [{
          "Effect": "Allow",
          "Action": [
            "ce:GetCostAndUsage",
            "ce:GetDimensionValues",
            "ce:GetReservationCoverage",
            "ce:GetReservationPurchaseRecommendation",
            "ce:GetReservationUtilization",
            "ce:GetTags"
          ],
          "Resource": "*"
        }]
```

OrganizationsListAccountsPolicy

授予列出子账户名称和 ID 的只读权限。

```
        "Statement": [{
          "Effect": "Allow",
          "Action": [
            "organizations:ListAccounts"
          ],
```

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

SESBulkTemplatedCrudPolicy

允许发送 Amazon SES 电子邮件、模板电子邮件和模板批量电子邮件以及验证身份。

```
        "Statement": [
          {
            "Effect": "Allow",
            "Action": [
              "ses:GetIdentityVerificationAttributes",
              "ses:SendEmail",
              "ses:SendRawEmail",
              "ses:SendTemplatedEmail",
              "ses:SendBulkTemplatedEmail",
              "ses:VerifyEmailIdentity"
            ],
            "Resource": {
              "Fn::Sub": [
                "arn:${AWS::Partition}:ses:${AWS::Region}:${AWS::AccountId}:identity/
${identityName}",
                {
                  "identityName": {
                    "Ref": "IdentityName"
                  }
                }
              ]
            }
          }
        ]
```

SESEmailTemplateCrudPolicy

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": "*"
        }]
```

FilterLogEventsPolicy

授予过滤权限CloudWatch记录来自指定日志组的事件。

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

SSMParameterRead策略

授予从 Amazon EC2 Systems Manager (SSM) 参数存储访问参数的权限，以便在此账户中加载密码。

Note

如果你没有使用默认密钥，你还需要`KMSDecryptPolicy`政策。

```
        "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

授予启动 Step Functions 状态机执行的权限。

```
        "Statement": [
          {
            "Effect": "Allow",
            "Action": [
              "states:StartExecution"
            ],
            "Resource": {
              "Fn::Sub": [
                "arn:${AWS::Partition}:states:${AWS::Region}:
${AWS::AccountId}:stateMachine:${stateMachineName}",
                {
                  "stateMachineName": {
                    "Ref": "StateMachineName"
                  }
                }
              ]
            }
          }
        ]
```

CodeCommitCrudPolicy

授予创建、读取、更新和删除特定对象的权限CodeCommit存储库。

```
        "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

授予读取特定对象的权限CodeCommit存储库。

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

AthenaQuery策略

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"
                  }
                }
              ]
            }
          }
        ]
```

TextractPolicy

授予对 Amazon Textract 的完全访问权限。

```
        "Statement": [
         {
            "Effect": "Allow",
            "Action": [
              "textract:*"
            ],
            "Resource": "*"
          }
        ]
```

TextractDetectAnalyzePolicy

允许使用 Amazon 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 Amazon Textract 获取检测到和分析的文档。

```
        "Statement": [
          {
            "Effect": "Allow",
            "Action": [
              "textract:GetDocumentTextDetection",
              "textract:GetDocumentAnalysis"
            ],
            "Resource": "*"
          }
        ]
```

EventBridgePutEvents策略

授予将事件发送到 Amazon 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"
                  }
                }
              ]
            }
          }
        ]
```

ElasticMapReduceModifyInstanceFleet策略

授予列出集群内实例队列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"
        }
      ]
```

ElasticMapReduceSetTerminationProtectionPolicy

授予为集群设置终止保护的权限。

```
      "Statement": [
        {
          "Action": "elasticmapreduce:SetTerminationProtection",
          "Resource": {
            "Fn::Sub": [
              "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:
${AWS::AccountId}:cluster/${clusterId}",
              {
                "clusterId": {
                  "Ref": "ClusterId"
                }
              }
            ]
          },
          "Effect": "Allow"
        }
      ]
```

ElasticMapReduceModifyInstanceGroups策略

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"
        }
      ]
```

ElasticMapReduceCancelStepsPolicy

授予取消运行的集群中的一个或多个待处理步骤的权限。

```
        "Statement": [
          {
            "Action": "elasticmapreduce:CancelSteps",
            "Resource": {
              "Fn::Sub": [
                "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:
${AWS::AccountId}:cluster/${clusterId}",
                {
                  "clusterId": {
                    "Ref": "ClusterId"
                  }
                }
              ]
            },
            "Effect": "Allow"
          }
        ]
```

ElasticMapReduceTerminateJobFlows策略

授予关闭集群的权限。

```
        "Statement": [
          {
            "Action": "elasticmapreduce:TerminateJobFlows",
            "Resource": {
              "Fn::Sub": [
                "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:
${AWS::AccountId}:cluster/${clusterId}",
                {
                  "clusterId": {
                    "Ref": "ClusterId"
                  }
                }
              ]
            },
            "Effect": "Allow"
          }
        ]
```

ElasticMapReduceAddJobFlowStepsPolicy

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"
        }
      ]
```

SageMakerCreateEndpoint策略

授予在中创建终端节点的权限SageMaker.

```
      "Statement": [
        {
          "Action": [
            "sagemaker:CreateEndpoint"
          ],
          "Resource": {
            "Fn::Sub": [
              "arn:${AWS::Partition}:sagemaker:${AWS::Region}:${AWS::AccountId}:endpoint/
${endpointName}",
              {
                "endpointName": {
                  "Ref": "EndpointName"
                }
              }
            ]
          },
          "Effect": "Allow"
        }
      ]
```

SageMakerCreateEndpointConfigPolicy

授予在中创建终端节点配置的权限SageMaker.

```
      "Statement": [
        {
          "Action": [
            "sagemaker:CreateEndpointConfig"
          ],
          "Resource": {
            "Fn::Sub": [
              "arn:${AWS::Partition}:sagemaker:${AWS::Region}:${AWS::AccountId}:endpoint-
config/${endpointConfigName}",
              {
                "endpointConfigName": {
                  "Ref": "EndpointConfigName"
                }
              }
            ]
          },
          "Effect": "Allow"
        }
      ]
```

EcsRunTaskPolicy

授予为任务定义启动新任务的权限。

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

EFSWriteAccess策略

授予挂载具有写入权限的 Amazon EFS 文件系统的权限。

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

```
            }
          }
        }
      ]
```

Route53ChangeResourceRecordSets策略

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"
                }
              }
            ]
          }
        }
      ]
```

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"
                }
              }
            ]
          }
        }
      ]
```

映像存储库

Amazon SAM借助构建容器映像，简化了无服务器应用程序的持续集成和持续部署 (CI/CD) 任务。那些映像Amazon SAM提供包括Amazon SAM用于许多受支持的命令行接口 (CLI) 和构建工具Amazon Lambda运行

时。这使得使用Amazon SAMCLI。您可以将这些映像与 CI/CD 系统一起使用，以自动构建和部署Amazon SAM应用程序。有关示例，请参阅 使用 CI/CD 系统部署 (p. 221)。

Amazon SAM构建容器映像 URI 使用Amazon SAM该图片中包含 CLI。如果指定未标记 URI，则将使用最新版本。例如，`public.ecr.aws/sam/build-nodejs14.x`使用最新的图片。但是，`public.ecr.aws/sam/build-nodejs14.x:1.24.1`使用包含的图片Amazon SAMCLI 版本 1.24.1。

从版本 1.33.0 开始Amazon SAMCLI、`x86_64`和`arm64`容器映像可用于支持的运行时。有关更多信息，请参阅 。Lambda 运行时中的Amazon Lambda开发人员指南.

> Note
>
> 在版本 1.22.0 之前Amazon SAMCLI、 DockerHub 是默认存储库Amazon SAMCLI 从拉取容器映像。从 1.22.0 版开始，默认存储库更改为 Amazon Elastic Container Registry Public (Amazon ECR Public)。要从当前默认值以外的存储库中提取容器映像，可以使用sam build (p. 252)命令使用--build-image选项。本主题末尾的示例展示了如何使用构建应用程序 DockerHub 存储库映像。

映像存储库 URI

下表列出了的 URIAmazon ECR Public构建容器映像，您可以使用这些映像构建和打包无服务器应用程序Amazon SAM.

运行时	Amazon ECR Public （默认从版本 1.22.0 开始）	Dockerhub （默认值为 1.22.0 版本 1.22.0）
Node.js 16	public .ecr.aws/build-nodejs16.x	不支持
Node.js 14	public .ecr.aws/build-nodejs14.x	亚马逊/aws-sam-cli-build-image-nodejs14.x
Node.js 12	public .ecr.aws/build-nodejs12.x	亚马逊/aws-sam-cli-build-image-nodejs12.x
Node.js 10	public .ecr.aws/build-nodejs10.x	亚马逊/aws-sam-cli-build-image-nodejs10.x
Python 3.9	public .ecr.aws/build-python3.9	不支持
Python 3.8	public .ecr.aws/build-python3.8	亚马逊/aws-sam-cli-build-图像-python3.8
Python 3.7	public .ecr.aws/build-python3.7	亚马逊/aws-sam-cli-build-图像-python3.7
Python 3.6	public .ecr.aws/build-python3.6	亚马逊/aws-sam-cli-build-图像-python3.6
Python 2.7	public .ecr.aws/build-python2.7	亚马逊/aws-sam-cli-build-图像-python2.7
Ruby 2.7	public .ecr.aws/build-ruby2.7	亚马逊/aws-sam-cli-build-image-ruby2.7
Ruby 2.5	public .ecr.aws/build-ruby2.5	亚马逊/aws-sam-cli-build-image-ruby2.5
Java 11	映像 .ecr.aws/build-java11	亚马逊/aws-sam-cli-build-image-java11

运行时	Amazon ECR Public （默认从版本 1.22.0 开始）	Dockerhub （默认值为 1.22.0 版本 1.22.0）
Java 8 (AL2)	public .ecr.aws/build-java8.al2	亚马逊/aws-sam-cli-build-image-java8.al2
Java 8	public .ecr.aws/build-java8	亚马逊/aws-sam-cli-build-image-java8
Go 1.x	映像 .ecr.aws/build-go1.x	亚马逊/aws-sam-cli-build-image-go1.x
自定义运行时 (AL2)	映像 .ecr.aws/build 提供 Dal2	亚马逊/aws-sam-cli-build-图片提供。al2
自定义运行时	Aupl.ecr.aws/Build 提供	亚马逊/aws-sam-cli-build 图片提供

示例

以下两个示例命令使用来自 DockerHub 存储库：

```
# 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
```

逐步部署无服务器应用程

如果您使用Amazon SAM为了创建无服务器应用程序，它内置CodeDeploy以提供渐进的 Lambda 部署。只需几行配置，Amazon SAM为您执行以下操作：

- 部署您的 Lambda 函数的新版本，并自动创建指向新版本的别名。
- 逐步将客户流量转移到新版本，直到您确认它按预期方式运行或者回滚更新。
- 定义流量前和转移流量后的测试函数，以验证新部署的代码是否已正确配置并且您的应用程序是否按预期方式
- 如果触发了 CloudWatch 警报，则回滚部署。

> **Note**
>
> 如果你通过你的Amazon SAM模板，系统会自动为您创建一个 CodeDeploy 资源。您可 CodeDeploy 接通过Amazon Web Services Management Console.

示例

下面的示例演示了一个使用 CodeDeploy 将客户流量逐步转移到您新部署的版本的简单版本：

```
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
```

这些修订版本Amazon SAM模板执行以下操作：

- AutoPublishAlias：通过添加此属性并指定别名，Amazon SAM：
  - 根据对 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 (除非已存在别名) 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.
- 部署首选项类型：在上一个示例中，10% 的客户流量立即转移到新版本。在 10 分钟之后，所有流量均转移到新版本。但是，如果你的预挂钩/后挂钩如果触发了 CloudWatch 警报，CodeDeploy 将回滚您的部署。下表概述了除以前使用的选项之外其他可用的流量转移选项。请注意以下几点：
  - 加纳利：流量在两次增量中转移。您可以从预定义的金丝雀部署选项中 此选项指定在第一次增量中转移到更新后的 Lambda 函数版本的流量百分比以及以分钟为单位的间隔；然后指定在第二次增量中转移剩余的流量。
  - 线性：流量使用相等的增量转移，在每次递增之间间隔的分钟数相同。您可以从预定义的线性选项中进行选择，这些选项指定在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。
  - 一次性：所有流量均从原始 Lambda 函数一次性地转移到更新后的 Lambda 函数版本。

部署首选项类型
Canary10Percent30Minutes
Canary10Percent5Minutes
Canary10Percent10Minutes
Canary10Percent15Minutes
Linear10PercentEvery10Minutes
Linear10PercentEvery1Minute
Linear10PercentEvery2Minutes
Linear10PercentEvery3Minutes
AllAtOnce

- Alarms：这些警报是由部署中出现的任何错误触发的 CloudWatch 警报。他们会自动回滚您的部署。例如，您正在部署的更新代码是否在应用程序中造成错误。另一个示例是否有Amazon Lambda或者您指定的自定义 CloudWatch 指标已超过警报阈值。

- hooks：这些函数是转移流量前和转移流量后的测试函数，这些函数在开始将流量转移到新版本之前以及完成将流量转移到新版本之后
  - 预交通：在开始转移流量之前，CodeDeploy 将调用转移流量前挂钩 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，它将中止并将故障报告回Amazon CloudFormation. 如果该函数成功，CodeDeploy 将继续进行流量转移。
  - 邮政流量：转移流量后，CodeDeploy 将调用转移流量后挂钩 Lambda 函数。与转移流量前挂钩类似，该函数必须回调 CodeDeploy 来报告成功或失败。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。

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. 330).

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

权限

控制对的访问Amazon资源，Amazon SAM使用与Amazon CloudFormation. 有关更多信息，请参阅 。使用控制访问Amazon Identity and Access Management中的Amazon CloudFormation用户指南.

授予用户管理无服务器应用程序的权限的主要选项有三种。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Amazon托管策略。
- 特定于授予Amazon Identity and Access Management(IAM) 权限。

根据您选择的选项，用户只能管理包含Amazon他们有权访问的资源。

以下各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，他们可以管理包含以下任意组合的无服务器应用程序Amazon资源的费用。这是最简单的选择，但它也授予用户最广泛的权限集，因此使他们能够执行影响最大的操作。

有关向用户授予管理员权限的更多信息，请参阅创建您的第一个 IAM 管理员用户和组中的IAM 用户指南.

附加必要Amazon托管策略

您可以使用以下方法授予用户一部分权限Amazon托管策略，而不是授予完全管理员权限。如果使用此选项，请确保Amazon托管策略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，以下内容Amazon托管策略足以部署示例 Hello World (p. 15)：

- AWSCloudFormationFullAccess
- IAMFullAccess
- AWSLambda_FullAccess
- 亚马逊 APIGatewayAdministrator
- 亚马逊 S3FullAccess
- AmazonEC2ContainerRegistryFullAccess

有关将策略附到 IAM 用户上的信息，请参阅更改 IAM 用户的权限中的IAM 用户指南.

授予特定的 IAM 权限

对于最精细的访问控制级别，您可以使用以下方法向用户授予特定 IAM 权限策略声明. 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，这样他们就无法授予自己升级的权限。因此，作为管理员，你必须首先创建Lambda 执行角色这将在用户将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 (p. 15)这AWSLambdaBasicExecutionRole足以运行该应用程序。创建 Lambda 执行角色后，请修改Amazon SAM示例 Hello World 应用程序的模板文件，用于将以下属性添加到`AWS::Serverless::Function`资源：

```
  Role: lambda-execution-role-arn
```

修改后的 Hello World 应用程序就位后，以下策略声明为用户授予部署、更新和删除应用程序的足够权限：

```
{
    "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

本节中的示例策略声明授予了足够的权限，以便您部署、更新和删除示例 Hello World (p. 15). 如果向应用程序添加其他资源类型，则需要更新策略声明以包含以下内容：

1. 允许您的应用程序调用该服务的操作。
2. 服务委托人（如果服务的操作需要）。

例如，如果添加 Step Functions 工作流程，则可能需要为列出的操作添加权限这里，以及`states.amazonaws.com`服务委托人。

有关 IAM 策略的更多信息，请参阅管理 IAM 策略中的IAM 用户指南.

重要提示

本节包含的重要说明和已知问题Amazon Serverless Application Model.

安装Amazon SAM32 位 Windows 上的 CLI

对该项的支持Amazon SAM32 位 Windows 上的 CLI 很快就会被弃用。如果您在 32 位系统上运行，我们建议您升级到 64 位系统，然后按照中的说明进行操作安装Amazon SAM在 Windows 上执行 CLI (p. 9).

如果无法升级到 64 位系统，则可以使用旧 Docker 工具箱和Amazon SAMCLI 在 32 位系统上。但是，这将导致您遇到某些限制Amazon SAMCLI。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 位本地编译容器，则无法在 32 位系统上本地测试它。

安装Amazon SAM在 32 位系统上执行 CLI，执行以下命令：

```
pip install aws-sam-cli
```

Important

虽然`pip install aws-sam-cli`命令也适用于 64 位 Windows，我们建议您使用64 位 MSI要安装Amazon SAM64 位系统上的 CLI。

Amazon SAM 的文档历史记录

下表描述了每次发布中所做的重要更改Amazon Serverless Application Model开发人员指南. 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：2021 年 10 月 1 日

变更	说明	日期
Lambda 指令集架构 (p. 335)	使用Amazon SAM用于构建 Lambda 函数和 Lambda 层的 CLI`x86_64`要么`arm64`指令集架构。有关更多信息，请参阅 。架构的属性`AWS::Serverless::Function`资源类型和CompatibleArchitectures的属性`AWS::Serverless::LayerVersion`资源类型。	2021 年 10 月 1 日
生成示例流水线配置 (p. 335)	使用Amazon SAMCLI 使用新的 CLI 为多个 CI/CD 系统生成示例管道`sam pipeline bootstrap`和`sam pipeline init`命令。有关更多信息，请参阅 。生成示例 CI/CD 流水线.	2021 年 7 月 21 日
Amazon SAMCLIAmazon CDK集成（预览版，第 2 阶段）(p. 335)	在公共预览版的第 2 阶段中，您现在可以使用Amazon SAM用于打包和部署的 CLIAmazon CDK应用程序。你也可以下载样本Amazon CDK应用程序直接使用Amazon SAMCLI。有关更多信息，请参阅 。Amazon Cloud Development Kit (Amazon CDK)（预览版）.	2021 年 7 月 13 日
RabbitMQ 作为函数的事件源 (p. 335)	添加了无服务器函数事件源。添加了针对 RabbitMQ 的事件源。有关更多信息，请参阅 。`SourceAccessConfigurations`的属性`MQ`的事件源`AWS::Serverless::Function`资源类型。	2021 年 7 月 7 日
使用 Amazon ECR 部署无服务器应用程序构建容器镜像 (p. 335)	使用 Amazon ECR 构建容器镜像使用常见 CI/CD 系统部署无服务器应用程序，例如Amazon CodePipeline，詹金斯， GitLab CI/CD，以及 GitHub 操作。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
调试Amazon SAM带有的应用程序Amazon工具包 (p. 335)	AmazonToolkit 现在支持分步调试，集成开发环境 (IDE) 和运行时	2021 年 5 月 20 日

	的更多组合。有关更多信息，请参阅 。使用Amazon工具包.	
Amazon SAMCLIAmazon CDK集成（预览）(p. 335)	现在，可以将使用Amazon SAM用于本地测试和构建的CLIAmazon 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)（预览版）.	2021 年 4 月 29 日
默认容器镜像存储库更改为Amazon ECR Public (p. 335)	默认容器镜像存储库更改为DockerHub 到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 SAMCLI 版本 (p. 335)	您现在可以安装预发行版本的Amazon SAMCLI，它是每晚构建的。有关更多信息，请参阅 。每晚构建数你选择的操作系统副主题部分安装Amazon SAMCLI.	2021 年 3 月 25 日
构建容器环境变量支持 (p. 335)	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 。`--container-env-var`和`--container-env-var-file`选项`sam build`.	2021 年 3 月 4 日
新 Linux 安装流程 (p. 335)	现在，可以安装Amazon SAM使用本机 Linux 安装程序的 CLI。有关更多信息，请参阅 。安装Amazon SAM在 Linux 上的.	2021 年 2 月 10 日
Support 死信队列EventBridge (p. 335)	添加了对死信队列的支持EventBridge 和`Schedule`无服务器函数和状态机的事件源。有关更多信息，请参阅 。`DeadLetterConfig`的属性`EventBridgeRule`和`Schedule`事件源，两者都是`AWS::Serverless::Function`和`AWS::Serverless::StateMachine`资源类型。	2021 年 1 月 29 日
Support 自定义检查点 (p. 335)	增加了对 DynamoDB 和无服务器函数的 Kinesis 事件源的自定义检查点的支持。有关更多信息，请参阅 。`FunctionResponseTypes`的属性`Kinesis`和`DynamoDB`的数据类型`AWS::Serverless::Function`资源类型。	2021 年 1 月 29 日

Support 翻滚窗户 (p. 335)	添加了对 DynamoDB 翻滚窗口的支持，为无服务器函数添加了 Kinesis 事件源的支持。有关更多信息，请参阅 。`TumblingWindowInSeconds`的属性`Kinesis`和`DynamoDB`的数据类型`AWS::Serverless::Function`资源类型。	2020 年 12 月 17 日
Support 温水容器 (p. 335)	在使用本地测试时添加了对温容器的支持Amazon SAMCLI 命令`sam local start-api`和`sam local start-lambda`. 有关更多信息，请参阅 。`--warm-containers`这些命令的选项。	2020 年 12 月 16 日
Support Lambda 容器镜像 (p. 335)	添加对 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. 335)	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 配置代码签名Amazon SAM应用程序.	2020 年 11 月 23 日
Support parallel 构建和缓存构建 (p. 335)	通过添加两个选项来提高无服务器应用程序版本的性能`sam build`命令：`--parallel`，它 parallel 构建函数和层，而不是按顺序构建，以及`--cached`，当没有进行需要重建的更改时，它会使用先前版本中的构建构件。	2020 年 11 月 10 日
Support Amazon MQ 和双向 TLS 身份验证 (p. 335)	添加了针对 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 。`EventSource`和`MQ`的数据类型`AWS::Serverless::Function`资源类型。还增加了对对 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 。`DomainConfiguration`的数据类型`AWS::Serverless::Api`资源类型，或`HttpApiDomainConfiguration`的数据类型`AWS::Serverless::HttpApi`资源类型。	2020 年 11 月 5 日
使用Support 于 HTTP API 的 Lambda 授权方 (p. 335)	添加了针对 Lambda 授权方的支持`AWS::Serverless::HttpApi`资源类型。有关更多信息，请参阅 。Lambda 授权方示例 (`AWS::Serverless::HttpApi`).	2020 年 10 月 27 日

Support 多个配置文件和环境 (p. 335)	增加了对存储默认参数值的多个配置文件和环境的支持Amazon SAMCLI 命令。有关更多信息，请参阅 。Amazon SAMCLI 配置文件.	2020 年 9 月 24 日
Support 带Step Functions 的 X-Ray，在控制 API 访问权限时使用引用 (p. 335)	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 。`Tracing`的属性`AWS::Serverless::StateMachine`资源类型。在控制 API 访问权限时，还添加了对引用的支持。有关更多信息，请参阅 。`ResourcePolicyStatement`数据类型。	2020 年 9 月 17 日
Support Amazon MSK (p. 335)	添加了无服务器功能事件源。添加了针对 Amazon MSK 作为事件源的支持。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 。`EventSource`和`MSK`的数据类型`AWS::Serverless::Function`资源类型。	2020 年 8 月 13 日
Support Amazon EFS (p. 335)	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 函数代码访问和修改共享资源。有关更多信息，请参阅 。`FileSystemConfigs`的属性`AWS::Serverless::Function`资源类型。	2020 年 6 月 16 日
编排编排无服务器应用程序 (p. 335)	增加了对通过使用创建 Step Functions 状态机来编排应用程序的支持Amazon SAM. 有关更多信息，请参阅 。编排Amazon资源与Amazon Step Functions还有`AWS::Serverless::StateMachine`资源类型。	2020 年 5 月 27 日
构建自定义运行时 (p. 335)	添加了用于构建自定义运行时。有关更多信息，请参阅 。构建自定义运行时.	2020 年 5 月 21 日
建筑层 (p. 335)	添加了用于构建个性化功能`LayerVersion`资源。有关更多信息，请参阅 。建筑层.	2020 年 5 月 19 日
生成Amazon CloudFormation资源 (p. 335)	提供了有关详细信息Amazon CloudFormation资源Amazon SAM生成以及如何引用它们。有关更多信息，请参阅 。生成Amazon CloudFormation资源.	2020 年 4 月 8 日

设置Amazon证书 (p. 335)	添加了设置说明Amazon证书，以防你还没有将它们设置为与其他人一起使用Amazon工具，例如其中一个AmazonSDK 或Amazon CLI. 有关更多信息，请参阅 。设置Amazon证书.	2020 年 1 月 17 日
Amazon SAM规范Amazon SAM命令行界面更新 (p. 335)	迁移Amazon SAM规格表 GitHub. 有关更多信息，请参阅 。Amazon SAM规格. 还更新了部署工作流程，更改了`sam deploy`命令。	2019 年 11 月 25 日
控制对 API Gateway API 和策略模板更新的访问权限的新选项 (p. 335)	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 的. 还更新了两个策略模板： RekognitionFacesPolicy 和 ElasticsearchHttpPostPolicy. 有关更多信息，请参阅 。Amazon SAMPolicy.	2019 年 8 月 29 日
入门更新 (p. 335)	更新了入门章节，改进了的安装说明Amazon SAMCLI 和 Hello World 教程。有关更多信息，请参阅 Amazon SAM 入门。	2019 年 7 月 25 日
控制对 API Gateway API 的 (p. 335)	增加了对控制 API Gateway API 访问权限的支持。有关更多信息，请参阅 。控制对 API Gateway API 的.	2019 年 3 月 21 日
增加了`sam publish`到Amazon SAMCLI (p. 335)	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 。使用发布无服务器应用程序Amazon SAMCLI.	2018 年 12 月 21 日
嵌套应用程序和层支持 (p. 335)	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层.	2018 年 11 月 29 日
增加了`sam build`到Amazon SAMCLI (p. 335)	新的`sam build`命令在Amazon SAMCLI 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
添加了新的安装选项Amazon SAMCLI (p. 335)	添加了 Linuxbrew (Linux)、MSI (Windows) 和 Homebrew (macOS) 安装选项Amazon SAMCLI。有关更多信息，请参阅安装 Amazon SAM CLI。	2018 年 11 月 7 日

新指南 (p. 335)	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。