aws

開發人員指南

AWS Serverless Application Model

AWS Serverless Application Model: 開發人員指南

Table of Contents

什麼是 AWS Serverless Application Model (AWS SAM)？

AWS Serverless Application Model (AWS SAM) 是使用基礎架構即程式碼 (IaC) 建置無伺服器應用程式的開放原始碼架構。使用簡寫語法，開發人員宣告AWS CloudFormation資源和專門 AWS SAM的無伺服器資源，這些資源會在部署期間轉換為基礎結構。該框架包括兩個主要組成部分： AWS SAMCLI 和 AWS SAM 項目。 AWS SAM 專案是執行時建立的應用程式專案目錄sam init。該 AWS SAM 項目包括類似 AWS SAM 模板的文件，其中包括模板規範（用於聲明資源的速記語法）。

主要功能

AWS SAM 提供各種好處，可改善開發人員體驗，讓您：

使用更少的程式碼，快速定義應用程式基礎架構

撰寫 AWS SAM 範本以定義無伺服器應用程式基礎結構程式碼。直接部署範本 AWS CloudFormation 以佈建您的資源。

在整個開發生命週期中管理無伺服器應用程式

透過開發生命週期的編寫、建置、部署、測試和監控階段，使用 AWS SAMCLI來管理您的無伺服器應用程式。如需詳細資訊，請參閱 該 AWS SAMCLI。

使用 AWS SAM 連接器在資源之間快速佈建權

在 AWS SAM 範本中使用 AWS SAM 連接器來定義 AWS 資源之間的權限。 AWS SAM 將您的程式碼轉換為促進您意圖所需的 IAM 許可。如需詳細資訊，請參閱 使用 AWS SAM 連接器管理資源權限。

在開發過程中持續將本機變更同步至雲端

使用此指 AWS SAMCLIsam sync令自動將本機變更同步至雲端，加速開發和雲端測試工作流程。如需詳細資訊，請參閱 使用同步sam sync到簡介 AWS 雲端。

管理Terraform無伺服器應用程式

使用可 AWS SAMCLI對 Lambda 函數和層執行本機偵錯和測試。如需詳細資訊，請參閱 AWS SAMCLITerraform支持。

相關資訊

- 如需有關如何 AWS SAM 運作的資訊，請參閱如何 AWS SAM 工作。

- 若要開始使用 AWS SAM，請參閱開始使用 AWS SAM。
- 如需如何使用建立無伺服器應 AWS SAM 用程式的概觀，請參閱如何使用 AWS SAM。

如何 AWS SAM 工作

AWS SAM 由您用來建立無伺服應用程式的兩個主要元件組成：

1. 該 AWS SAM 項目— 執行sam init命令時建立的資料夾和檔案。此目錄包含AWS SAM 範本，這是定義 AWS 資源的重要檔案。此範本包含範AWS SAM 本規格 — 開放原始碼架構，其中包含簡化的簡短語法，可用來定義無伺服器應用程式的函數、事件、API、組態和權限。
2. 該 AWS SAMCLI— 命令列工具，可用於 AWS SAM 專案和受支援的第三方整合，以建置和執行無伺服器應用程式。 AWS SAMCLI) 是您用來在 AWS SAM 專案上執行命令，並最終將其轉換為無伺服器應用程式的工具。

若要表示定義無伺服器應用程式的資源、事件來源對應及其他屬性，您可以在 AWS SAM 範本和 AWS SAM 專案中的其他檔案中定義資源並開發應用程式。您可以使 AWS SAMCLI用在 AWS SAM 專案上執行命令，也就是初始化、建置、測試和部署無伺服器應用程式的方式。

> 無伺服器的新手？
>
> 我們建議您檢查一下無伺服器概念。

什麼是模 AWS SAM 板規範？

範 AWS SAM 本規格是開放原始碼架構，可用來定義和管理無伺服器應用程式基礎結構程式碼。模 AWS SAM 板規範是：

- 建立在 AWS CloudFormation-您可以直接在 AWS SAM 模板中使用 AWS CloudFormation 語法，利用其對資源和屬性配置的廣泛支持。如果您已經熟悉 AWS CloudFormation，就不需要學習管理應用程式基礎結構程式碼的新服務。
- 的延伸 AWS CloudFormation — AWS SAM 提供了自己獨特的語法，特別著重於加速無伺服器開發。您可以在同一個範本中使用 AWS CloudFormation 和 AWS SAM 語法。
- 抽象的簡短語法 — 使用 AWS SAM 語法，您可以用更少的代碼行快速定義基礎結構，並降低發生錯誤的機會。其語法經過特別精心策劃，以抽象化定義無伺服器應用程式基礎結構的複雜性。

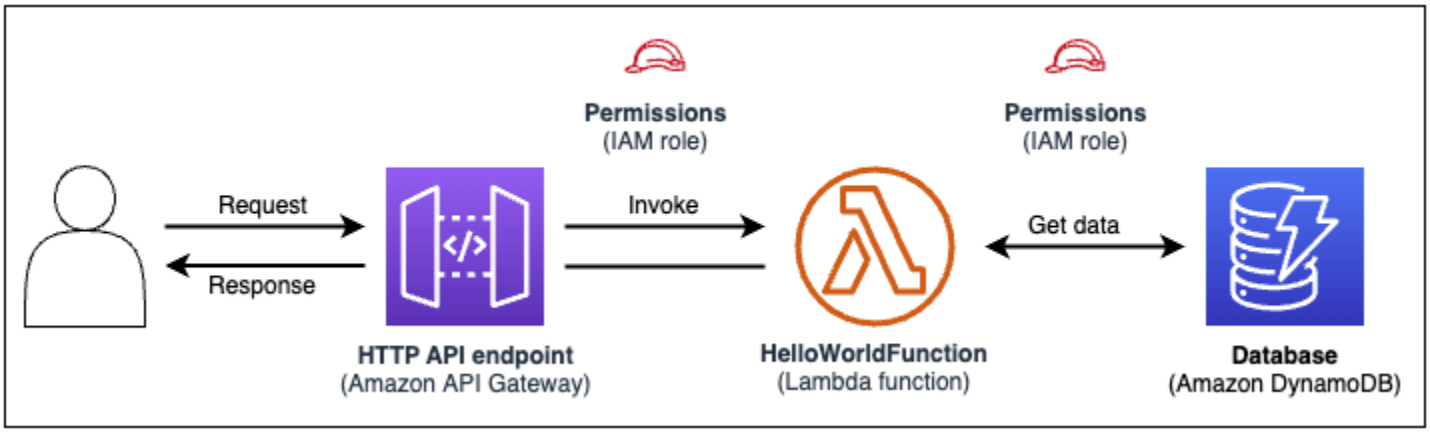
- 轉型 — AWS SAM 將範本轉換為透過佈建基礎結構所需的程式碼進行複雜的工作。 AWS CloudFormation

什麼是項 AWS SAM 目和 AWS SAM 模板？

該 AWS SAM 項目包括包含 AWS SAM 模板規範的 AWS SAM 模板。此規格是您用來定義無伺服器應用程式基礎結構的開放原始碼架構 AWS，以及一些可讓它們更容易使用的其他元件。從這個意義上說， AWS SAM 模板是 AWS CloudFormation 模板的擴展。

以下是基本無伺服器應用程式的範例。此應用程序處理請求，以通過 HTTP 請求從數據庫中獲取所有項目。它由以下部分組成：

1. 包含處理要求之邏輯的函數。
2. 用來做為用戶端 (要求程式) 與應用程式之間的通訊的 HTTP API。
3. 儲存項目的資料庫。
4. 應用程式安全執行的權限。



此應用程式的基礎結構程式碼可以在下列 AWS SAM 範本中定義：

```
AWSTemplateFormatVersion: 2010-09-09
Transform: AWS::Serverless-2016-10-31
Resources:
  getAllItemsFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: src/get-all-items.getAllItemsHandler
      Runtime: nodejs12.x
      Events:
```

```
        Api:
          Type: HttpApi
          Properties:
            Path: /
            Method: GET
    Connectors:
      MyConn:
        Properties:
          Destination:
            Id: SampleTable
          Permissions:
            - Read
  SampleTable:
    Type: AWS::Serverless::SimpleTable
```

在 23 行代碼中，定義了以下基礎結構：

- 使用該 AWS Lambda 服務的功能。
- 使用 Amazon API Gateway 服務的 HTTP API。
- 使用亞馬遜動態 B 服務的資料庫。
- 這些服務彼此互動所需的 AWS Identity and Access Management (IAM) 許可。

若要佈建此基礎結構，範本會部署至 AWS CloudFormation。在部署期間， AWS SAM 將 23 行程式碼轉換為在中產生這些資源所需的 AWS CloudFormation 語法 AWS。轉換後的 AWS CloudFormation 模板包含超過 200 行代碼！

轉換的 AWS CloudFormation 範本

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Resources": {
    "getAllItemsFunction": {
      "Type": "AWS::Lambda::Function",
      "Metadata": {
        "SamResourceId": "getAllItemsFunction"
      },
      "Properties": {
        "Code": {
          "S3Bucket": "aws-sam-cli-managed-default-samclisourcebucket-1a4x26zbcdkqr",
          "S3Key": "what-is-app/a6f856abf1b2c4f7488c09b367540b5b"
        },
```

```
        "Handler": "src/get-all-items.getAllItemsHandler",
        "Role": {
          "Fn::GetAtt": [
            "getAllItemsFunctionRole",
            "Arn"
          ]
        },
        "Runtime": "nodejs12.x",
        "Tags": [
          {
            "Key": "lambda:createdBy",
            "Value": "SAM"
          }
        ]
      }
    },
    "getAllItemsFunctionRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Action": [
                "sts:AssumeRole"
              ],
              "Effect": "Allow",
              "Principal": {
                "Service": [
                  "lambda.amazonaws.com"
                ]
              }
            }
          ]
        },
        "ManagedPolicyArns": [
          "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
        ],
        "Tags": [
          {
            "Key": "lambda:createdBy",
            "Value": "SAM"
          }
        ]
```

```
      }
    },
    "getAllItemsFunctionApiPermission": {
      "Type": "AWS::Lambda::Permission",
      "Properties": {
        "Action": "lambda:InvokeFunction",
        "FunctionName": {
          "Ref": "getAllItemsFunction"
        },
        "Principal": "apigateway.amazonaws.com",
        "SourceArn": {
          "Fn::Sub": [
            "arn:${AWS::Partition}:execute-api:${AWS::Region}:${AWS::AccountId}:
${__ApiId__}/${__Stage__}/GET/",
            {
              "__ApiId__": {
                "Ref": "ServerlessHttpApi"
              },
              "__Stage__": "*"
            }
          ]
        }
      }
    },
    "ServerlessHttpApi": {
      "Type": "AWS::ApiGatewayV2::Api",
      "Properties": {
        "Body": {
          "info": {
            "version": "1.0",
            "title": {
              "Ref": "AWS::StackName"
            }
          },
          "paths": {
            "/": {
              "get": {
                "x-amazon-apigateway-integration": {
                  "httpMethod": "POST",
                  "type": "aws_proxy",
                  "uri": {
                    "Fn::Sub": "arn:${AWS::Partition}:apigateway:
${AWS::Region}:lambda:path/2015-03-31/functions/${getAllItemsFunction.Arn}/invocations"
                  },
```

```
                "payloadFormatVersion": "2.0"
              },
              "responses": {}
            }
          }
        },
        "openapi": "3.0.1",
        "tags": [
          {
            "name": "httpapi:createdBy",
            "x-amazon-apigateway-tag-value": "SAM"
          }
        ]
      }
    }
  },
  "ServerlessHttpApiApiGatewayDefaultStage": {
    "Type": "AWS::ApiGatewayV2::Stage",
    "Properties": {
      "ApiId": {
        "Ref": "ServerlessHttpApi"
      },
      "StageName": "$default",
      "Tags": {
        "httpapi:createdBy": "SAM"
      },
      "AutoDeploy": true
    }
  },
  "SampleTable": {
    "Type": "AWS::DynamoDB::Table",
    "Metadata": {
      "SamResourceId": "SampleTable"
    },
    "Properties": {
      "AttributeDefinitions": [
        {
          "AttributeName": "id",
          "AttributeType": "S"
        }
      ],
      "KeySchema": [
        {
          "AttributeName": "id",
```

```
            "KeyType": "HASH"
          }
        ],
        "BillingMode": "PAY_PER_REQUEST"
      }
    },
    "getAllItemsFunctionMyConnPolicy": {
      "Type": "AWS::IAM::ManagedPolicy",
      "Metadata": {
        "aws:sam:connectors": {
          "getAllItemsFunctionMyConn": {
            "Source": {
              "Type": "AWS::Serverless::Function"
            },
            "Destination": {
              "Type": "AWS::Serverless::SimpleTable"
            }
          }
        }
      },
      "Properties": {
        "PolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Effect": "Allow",
              "Action": [
                "dynamodb:GetItem",
                "dynamodb:Query",
                "dynamodb:Scan",
                "dynamodb:BatchGetItem",
                "dynamodb:ConditionCheckItem",
                "dynamodb:PartiQLSelect"
              ],
              "Resource": [
                {
                  "Fn::GetAtt": [
                    "SampleTable",
                    "Arn"
                  ]
                },
                {
                  "Fn::Sub": [
                    "${DestinationArn}/index/*",
```

```
                    {
                      "DestinationArn": {
                        "Fn::GetAtt": [
                          "SampleTable",
                          "Arn"
                        ]
                      }
                    }
                  ]
                }
              ]
            }
          ]
        },
        "Roles": [
          {
            "Ref": "getAllItemsFunctionRole"
          }
        ]
      }
    }
  }
}
```

透過使用 AWS SAM，您可以定義 23 行基礎結構程式碼。 AWS SAM 將您的代碼轉換為佈建應用程序所需的 200 多行 AWS CloudFormation 代碼。

什麼是 AWS SAMCLI？

命令列工具可搭配 AWS SAM 範本和支援的第三方整合使用，以建置和執行無伺服器應用程式。 AWS SAMCLI使用 AWS SAMCLI來：

- 快速初始化新的應用程式專案。
- 建置用於部署的應用程式。
- 執行本地調試和測試。
- 部署您的應用程式。
- 設定 CI/CD 部署管線。
- 監控雲端中的應用程式並進行疑難排解。
- 在開發過程中將本機變更同步至雲端。

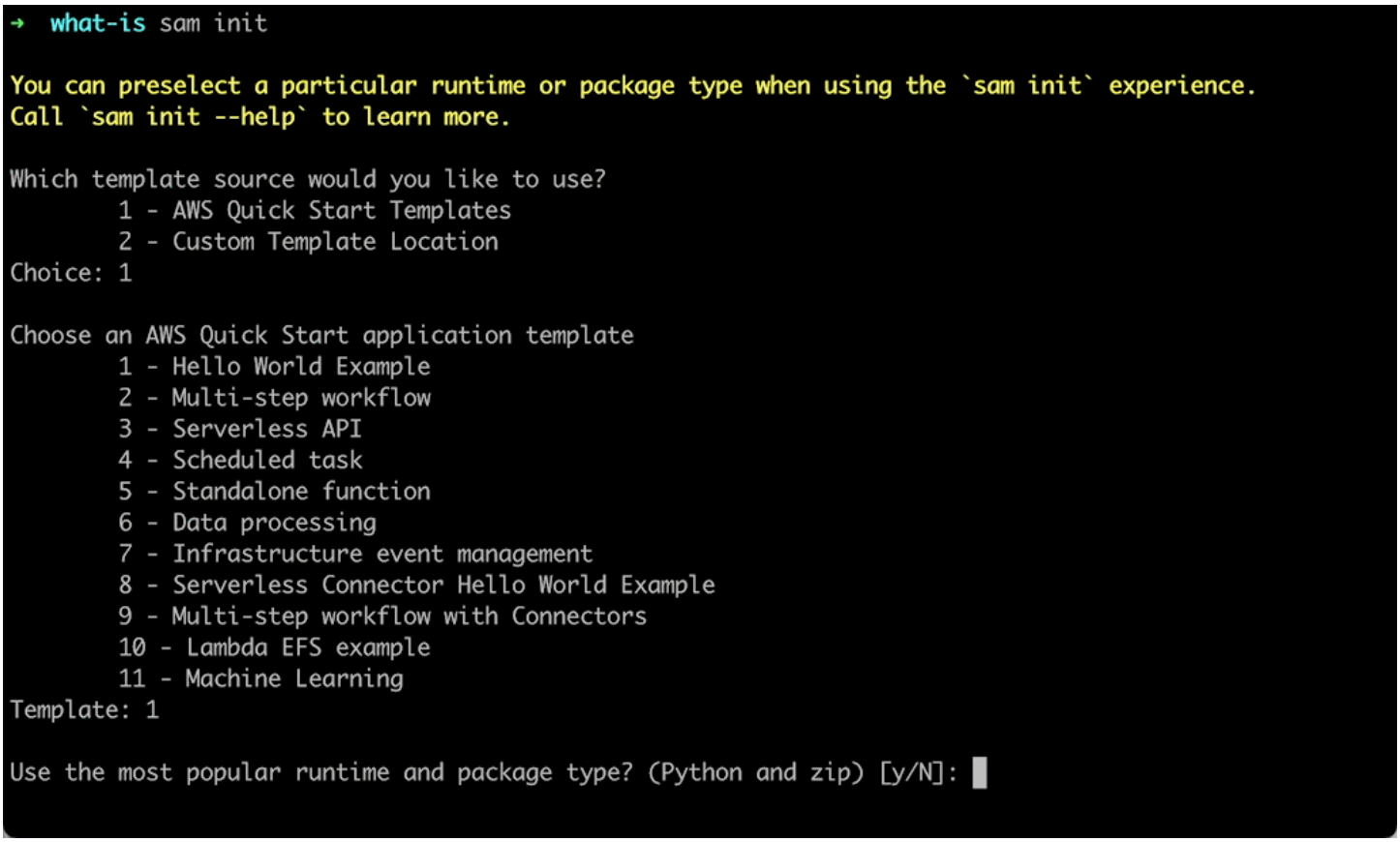
- 以及更多！

與 AWS SAM 和 AWS CloudFormation 範本搭配使用時， AWS SAMCLI最好使用。它還適用於第三方產品，例如Terraform.

初始化新專案

從初學者模板中選擇或選擇自定義模板位置以開始新項目。

在這裡，我們使用命sam init令來初始化一個新的應用程序項目。我們選擇你好世界範例項目開始。下AWS SAMCLI載入門範本並建立我們的專案資料夾目錄結構。



如需詳細資訊，請參閱使用sam init指令建立您的應用程式。

建置用於部署的應用程式

打 Package 您的函數依賴關係，並組織您的項目代碼和文件夾結構以準備部署。

在這裡，我們使用命sam build令來準備我們的應用程序進行部署。 AWS SAMCLI創建一個.aws-sam目錄並組織我們的應用程序依賴關係和文件在那裡進行部署。

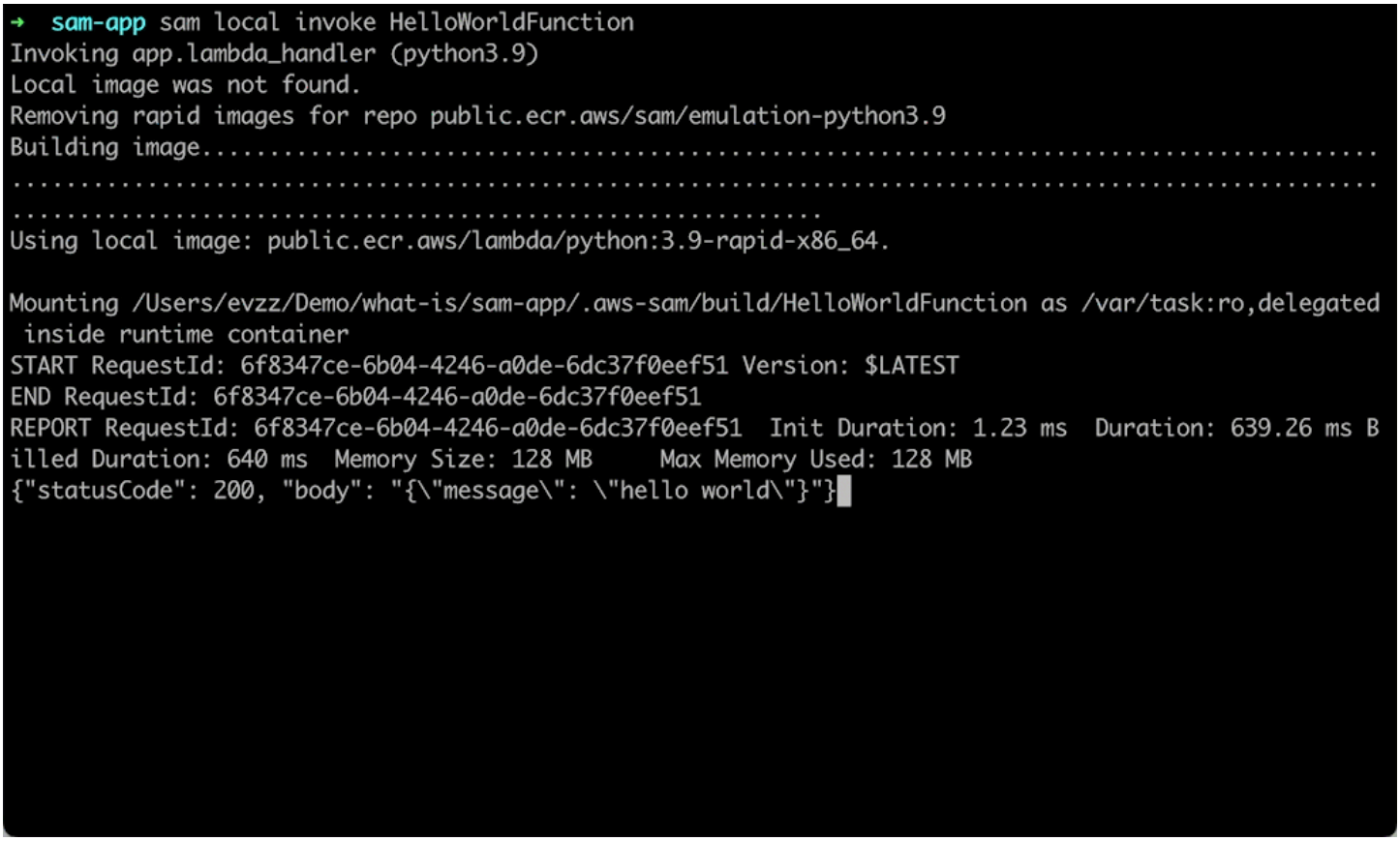
```
→  sam-app sam build
Building codeuri: /Users/evzz/Demo/what-is/sam-app/hello_world runtime: python3.9 metadata: {} archit
ecture: x86_64 functions: HelloWorldFunction
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource

Build Succeeded

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

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

如需詳細資訊，請參閱建置應用程式。

執行本地調試和測試

在您的本機電腦上，模擬事件、測試 API、叫用函數等，以偵錯和測試應用程式。

在這裡，我們使用命sam local invoke令來調用我們的`HelloWorldFunction`本地。為了做到這一點， AWS SAMCLI創建一個本地容器，構建我們的函數，調用它，並輸出結果。您可以使用 Docker 之類的應用程序在計算機上運行容器。



如需詳細資訊，請參閱測試您的應用和偵錯應用程式。

部署您的應用程式

設定應用程式的部署設定，並部署至 AWS 雲端以佈建資源。

在這裡，我們使用命sam deploy --guided令通過交互式流程部署我們的應用程序。引 AWS SAMCLI導我們完成配置應用程序的部署設置 AWS CloudFormation，將模板轉換為並部署 AWS CloudFormation 以創建我們的資源。

```
➜  sam-app sam deploy --guided

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

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

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

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

如需詳細資訊，請參閱部署您的應用程式和資源。

設定 CI/CD 部署管線

使用支援的 CI/CD 系統，建立安全的持續整合與傳遞 (CI/CD) 管線。

在這裡，我們使用命sam pipeline init --bootstrap令為我們的應用程序配置 CI/CD 部署管道。這會AWS SAMCLI引導我們完成我們的選項，並生成與我們的 CI/CD 系統一起使用的 AWS 資源和配置文件。

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

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

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

如需詳細資訊，請參閱使用 CI/CD 系統和管線進行部署。

監控雲端中的應用程式並進行疑難排解

檢視已部署資源的相關重要資訊、收集記錄，以及使用內建的監視工具，例如 AWS X-Ray.

在這裡，我們使用命sam list令來查看我們部署的資源。我們得到我們的 API 端點並調用它，這會觸發我們的功能。然後，我們用sam logs來查看函數的日誌。

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

如需詳細資訊，請參閱監控您的應用。

在開發過程中將本機變更同步至雲端

當您在本機電腦上進行開發時，會自動將變更同步到雲端。快速查看您的變更，並在雲端執行測試和驗證。

在這裡，我們使用sam sync --watch命令來 AWS SAMCLI監視本地更改。我們會修改`HelloWorldFunction`程式碼，並 AWS SAMCLI自動偵測變更，並將更新部署到雲端。

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

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

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

Stack update succeeded. Sync infra completed.

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

測試雲端中支援的資源

呼叫事件並將其傳遞至雲端中支援的資源。

在這裡，我們使用sam remote invoke命令來測試雲端中部署的 Lambda 函數。我們叫用 Lambda 函數並接收其記錄和回應。我們的 Lambda 函數設定為 AWS SAMCLI串流回應後，即時串流回應。

進一步了解

若要繼續學習 AWS SAM，請參閱下列資源：

- 完整的 AWS SAM 工作坊 — 旨在教您 AWS SAM 提供許多主要功能的工作坊。
- 與 SAM 的會話 — 由我們的 AWS 無伺服器開發人員倡導團隊建立的影片系列，內容包括 AWS SAM
- 無伺服器 Land — 將無 AWS 伺服器的最新資訊、部落格、影片、程式碼和學習資源整合在一起的網站。

後續步驟

如果這是您第一次使用 AWS SAM，請參閱開始使用 AWS SAM。

無伺服器概念

在使用 AWS Serverless Application Model (AWS SAM) 之前，請先瞭解基本的無伺服器概念。

無伺服器概念

事件驅動架構

無伺服器應用程式由個別 AWS 服務組成，例如用 AWS Lambda 於運算和用於資料庫管理的 Amazon DynamoDB，每個服務都會執行特殊角色。然後，這些服務會透過事件驅動架構彼此鬆散整合。若要深入了解事件驅動架構，請參閱什麼是事件驅動架構？ 。

基礎架構即程式碼 (IaC)

基礎結構即程式碼 (IaC) 是以與開發人員處理程式碼相同的方式來處理基礎結構的一種方式，將應用程式程式碼開發的嚴謹性套用至基礎結構佈建。您可以在範本檔案中定義基礎結構、將其部署到 AWS，然後為您 AWS 建立資源。使用 IaC，您可以在代碼中定義 AWS 要提供的內容。如需詳細資訊，請參閱 AWSAWS 白皮書簡介中的基礎架構即 DevOps 程式碼。

無伺服器技術

使用 AWS 無伺服器技術，您可以建置和執行應用程式，而不必管理自己的伺服器。所有伺服器管理都是透過完成的 AWS，提供許多好處，例如自動擴充和內建高可用性，讓您可以快速將您的想法投入生產環境。使用無伺服器技術，您可以專注於產品的核心，而不必擔心伺服器的管理和操作問題。若要進一步了解無伺服器，請參閱下列內容：

- 開啟無伺服器 AWS
- 無伺服器開發人員指南 — 提供雲端中無伺服器開發的概念性概觀。 AWS

如需核心無伺服器服務的基本簡介，請參閱 AWS 無伺服器 101：瞭解無伺服器 Land 上的無伺服器服務。

開始使用 AWS SAM

請檢閱並完成本節中的主題開始使用 AWS SAM 。 AWS SAM 前提提供設定 AWS 帳戶、建立 IAM 使用者、建立金鑰存取權，以及安裝和設定 AWS SAMCLI. 完成預測站點後，您就可以準備好了安裝 AWS SAMCLI，您可以在 Linux，視窗和 macOS 操作系統上執行此操作。安裝完成後，您可以選擇性地逐步完成 AWS SAM Hello World 教程。遵循本教學課程將引導您完成使 AWS SAM用建立基本無伺服器應用程式的程序。完成教學課程後，您就可以檢閱中詳細介紹的概念了如何使用 AWS Serverless Application Model （ AWS SAM ）。

主題

- AWS SAM 前提
- 安裝 AWS SAMCLI
- 教學課程：部署 Hello World 應用程式

AWS SAM 前提

在安裝和使用 AWS Serverless Application Model 指令行介面 (AWS SAMCLI) 之前，請先完成下列先決條件。

若要使用 AWS SAMCLI，您需要下列項目：

- AWS 帳戶、 AWS Identity and Access Management (IAM) 登入資料和 IAM 存取 key pair。
- 用於配置 AWS 認證的 AWS Command Line Interface （ AWS CLI ）。

主題

- 步驟 1：註冊一個 AWS 帳戶
- 步驟 2：建立 IAM 使用者帳戶
- 步驟 3：建立存取金鑰 ID 和私密存取金鑰
- 步驟 4：安裝 AWS CLI
- 步驟 5：使用設 AWS CLI 定 AWS 認證
- 後續步驟

步驟 1：註冊一個 AWS 帳戶

如果您沒有 AWS 帳戶，請完成以下步驟來建立一個。

若要註冊成為 AWS 帳戶

1. 開啟 https://portal.aws.amazon.com/billing/signup。
2. 請遵循線上指示進行。

   部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

   當您註冊一個時 AWS 帳戶，將創建AWS 帳戶根使用者一個。根使用者有權存取該帳戶中的所有 AWS 服務 和資源。安全性最佳做法是將管理存取權指派給使用者，並僅使用 root 使用者來執行需要 root 使用者存取權的工作。

步驟 2：建立 IAM 使用者帳戶

若要建立管理員使用者，請選擇下列其中一個選項。

選擇一種管理管理員的方式	到	By	您也可以
在 IAM Identity Center (建議)	使用短期憑證存取 AWS。 這與安全性最佳實務一致。有關最佳實務的資訊，請參閱 IAM 使用者指南中的 IAM 安全最佳實務。	請遵循 AWS IAM Identity Center 使用者指南的入門中的說明。	AWS IAM Identity Center 在《使用AWS Command Line Interface 者指南》中設定 AWS CLI 要使用的，以設定程式設計方式存取。
在 IAM 中 (不建議使用)	使用長期憑證存取 AWS。	請遵循 IAM 使用者指南中建立您的第一個 IAM 管理員使用者和使用者群組的說明。	請參閱 IAM 使用者指南中的管理 IAM 使用者的存取金鑰，設定程式設計存取。

步驟 3：建立存取金鑰 ID 和私密存取金鑰

對於 CLI 存取，您需要存取金鑰 ID 和私密存取金鑰。盡可能使用臨時憑證，而不是長期存取金鑰。臨時憑證包含存取金鑰 ID、私密存取金鑰，以及指出憑證何時到期的安全符記。如需詳細資訊，請參閱 IAM 使用者指南中的將臨時登入資料與 AWS 資源搭配使用。

如果使用者想要與 AWS 之外互動，則需要程式設計存取 AWS Management Console。授與程式設計存 AWS取權的方式取決於正在存取的使用者類型。

若要授與使用者程式設計存取權，請選擇下列其中一個選項。

哪個使用者需要程式設計存取權？	到	By
人力身分 (IAM Identity Center 中管理的使用者)	使用臨時登入資料來簽署對 AWS CLI、 AWS SDK 或 AWS API 的程式設計要求。	請依照您要使用的介面所提供的指示操作。 • 如需詳細資訊 AWS CLI，請參閱《使 AWS CLI 用AWS Command Line Interface者指南》 AWS IAM Identity Center中的〈配置使用〉。 • 如需 AWS SDK、工具和 AWS API，請參閱 AWS SDK 和工具參考指南中的 IAM 身分中心身分驗證。
IAM	使用臨時登入資料來簽署對 AWS CLI、 AWS SDK 或 AWS API 的程式設計要求。	遵循《IAM 使用者指南》中的〈將臨時登入資料搭配 AWS 資源使用〉中的指示
IAM	(不建議使用) 使用長期認證來簽署對 AWS CLI、 AWS SDK 或 AWS API 的程式設計要求。	請依照您要使用的介面所提供的指示操作。 • 如需相關資訊 AWS CLI，請參閱使用指南中的使用 IAM 使用者登入資料進行驗證。AWS Command Line Interface

哪個使用者需要程式設計存取權？	到	By
		• 對於 AWS SDK 和工具，請參閱 AWS SDK 和工具參考指南中的使用長期憑據進行身份驗證。 • 如需 AWS API，請參閱 IAM 使用者指南中的管理 IAM 使用者的存取金鑰。

步驟 4：安裝 AWS CLI

這 AWS CLI 是一個開放原始碼工具，可讓您 AWS 服務 使用命令列殼層中的命令進行互動。 AWS CLI 對於活動 (例如設定認證) AWS SAMCLI 需要。若要深入了解 AWS CLI，請參閱「什麼是 AWS Command Line Interface?」 在《AWS Command Line Interface 使用者指南》中。

若要安裝 AWS CLI，請參閱《使用指南》 AWS CLI中的〈安裝或更新最新版本的AWS Command Line Interface〉。

步驟 5：使用設 AWS CLI 定 AWS 認證

若要設定認證 AWS CLI

1. 從命`aws configure`令列執行命令。
2. 設定下列項目。選擇每個鏈接以了解更多信息：

    a. 存取金鑰識別碼
    b. 秘密存取金鑰
    c. AWS 區域
    d. 輸出格式

    下列範例顯示範本值。

```
$ aws configure
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

```
Default region name [None]: us-west-2
Default output format [None]: json
```

會將此資訊 AWS CLI 儲存在`credentials`和`config`檔案中命名的紀要 (設定集合) `default` 中。這些檔案位於主目錄中的`.aws`檔案中。根據預設，當您執行未明確指定要使用的設定檔的 AWS CLI 命令時，會使用此設定檔中的資訊。如需有關`credentials`檔案的詳細資訊，請參閱AWS Command Line Interface 使用指南中的組態和認證檔案設定。

如需有關設定身分證明的詳細資訊，例如使用現有的組態和認證檔案，請參閱《使AWS Command Line Interface 用指南》中的「快速設定」。

後續步驟

您現在可以安裝 AWS SAMCLI並開始使用 AWS SAM。若要安裝 AWS SAMCLI，請參閱安裝 AWS SAMCLI。

安裝 AWS SAMCLI

在支援的作業系統上安裝最新版本的 AWS Serverless Application Model 命令列介面 (AWS SAMCLI)。

如需管理目前安裝的版本的資訊 AWS SAMCLI，包括如何升級、解除安裝或管理夜間組建，請參閱管理 AWS SAMCLI版本。

> 這是您第一次安裝AWS SAM CLI嗎？
>
> 請先完成上一節的所有先決條件，然後再繼續進行。其中包含：
>
> 1. 註冊一個 AWS 帳戶。
> 2. 建立管理 IAM 使用者。
> 3. 創建訪問密鑰 ID 和秘密訪問密鑰。
> 4. 安裝 AWS CLI.
> 5. 設定 AWS 認證。

主題

- 安裝 AWS SAMCLI

安裝 AWS SAMCLI

> **Note**
>
> 從 2023 年 9 月開始， AWS 將不再維護 AWS SAMCLI (`aws/tap/aws-sam-cli`) 的 AWS 受管理Homebrew安裝程式。如果您使用Homebrew來安裝和管理 AWS SAMCLI，請參閱下列選項：
>
> - 若要繼續使用Homebrew，您可以使用社群管理的安裝程式。如需詳細資訊，請參閱 管理 AWS SAMCLI與 Homebrew。
> - 我們建議您使用本頁所述的其中一種第一方安裝方法。使用其中一種方法之前，請參閱從切換 Homebrew。

若要安裝 AWS SAMCLI，請遵循您作業系統的指示。

Linux

arm64 - command line installer

1. 將 AWS SAMCLI.zip 檔案下載至您選擇的目錄。
2. (選擇性) 您可以在安裝之前驗證安裝程式的完整性。如需說明，請參閱可選：驗證 AWS SAMCLI安裝程式的完整性。
3. 將安裝檔案解壓縮到您選擇的目錄中。以下是使用`sam-installation`子目錄的範例。

   > **Note**
   >
   > 如果您的作業系統沒有內建 unzip 命令，請使用對等的命令。

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

4. AWS SAMCLI通過運行`install`可執行文件來安裝。此可執行檔位於上一個步驟中使用的目錄中。以下是使用`sam-installation`子目錄的範例：

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

5. 驗證安裝。

```
$ sam --version
```

若要確認安裝成功，您應該會看到類似下列的輸出，但會以最新的 SAM CLI 版本取代括號中的文字：

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

x86_64 - command line installer

1. 將 AWS SAMCLI.zip 檔案下載至您選擇的目錄。
2. (選擇性) 您可以在安裝之前驗證安裝程式的完整性。如需說明，請參閱可選：驗證 AWS SAMCLI安裝程式的完整性。
3. 將安裝檔案解壓縮到您選擇的目錄中。以下是使用`sam-installation`子目錄的範例。

> **Note**
>
> 如果您的作業系統沒有內建 unzip 命令，請使用對等的命令。

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

4. AWS SAMCLI通過運行`install`可執行文件來安裝。此可執行檔位於上一個步驟中使用的目錄中。以下是使用`sam-installation`子目錄的範例：

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

5. 驗證安裝。

```
$ sam --version
```

若要確認安裝成功，您應該會看到一個輸出，將下列括弧文字取代為最新的可用版本：

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

macOS

安裝步驟

使用套件安裝程式來安裝 AWS SAMCLI. 此外，套件安裝程式還有兩種安裝方法可供您選擇：GUI 和命令列。您可以為所有使用者安裝，也可以只安裝目前的使用者。要為所有用戶安裝，需要超級用戶授權。

GUI - All users

若要下載套件安裝程式並安裝 AWS SAMCLI

> **Note**
>
> 如果您之前已安裝 AWS SAMCLI過Homebrew或pip，則需要先將其解除安裝。如需說明，請參閱解除安裝 AWS SAMCLI。

1. pkg將 macOS 下載到您選擇的目錄：
   - 對於運行英特爾處理器的 Mac 電腦，請選擇 x86_64 aws-sam-cli-macos-x86_64.pkg
   - 對於運行蘋果矽晶片的 Mac 電腦，請選擇 aws-sam-cli-macos-arm64.pkg

   > **Note**
   >
   > 您可以選擇在安裝之前驗證安裝程式的完整性。如需說明，請參閱可選：驗證 AWS SAMCLI安裝程式的完整性。

2. 執行下載的檔案，然後依照螢幕上的指示繼續進行簡介、讀我說明和授權步驟。
3. 在 「目的地選取」 中，選取「為此電腦的所有使用者安裝」。
4. 對於「安裝類型」，請選擇要安裝的 AWS SAMCLI位置，然後按「安裝」。建議的預設位置為`/usr/local/aws-sam-cli`。

> **Note**
>
> 若要 AWS SAMCLI使用sam指令叫用，安裝程式會在與/usr/local/aws-sam-cli/sam或您選擇的安裝資料夾之間/usr/local/bin/sam自動建立符號連結。

5. AWS SAMCLI將會安裝和安裝成功的訊息將會顯示。按「關閉」。

驗證安裝成功

- 請執行下列指令，確認 AWS SAMCLI已正確安裝，且符號連結已設定：

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

GUI - Current user

若要下載並安裝 AWS SAMCLI

> **Note**
>
> 如果您之前已安裝 AWS SAMCLI過Homebrew或pip，則需要先將其解除安裝。如需說明，請參閱解除安裝 AWS SAMCLI。

1. pkg將 macOS 下載到您選擇的目錄：
   - 對於運行英特爾處理器的 Mac 電腦，請選擇 x86_64 aws-sam-cli-macos-x86_64.pkg
   - 對於運行蘋果矽晶片的 Mac 電腦，請選擇 aws-sam-cli-macos-arm64.pkg

> **Note**
>
> 您可以選擇在安裝之前驗證安裝程式的完整性。如需說明，請參閱可選：驗證 AWS SAMCLI安裝程式的完整性。

2. 執行下載的檔案，然後依照螢幕上的指示繼續進行簡介、讀我說明和授權步驟。

3. 針對 「目的地選取」，選取「僅為我安裝」。如果沒有看到此選項，請前往下一個步驟。
4. 對於「安裝類型」，請執行下列操作：

   1. 選擇要安裝 AWS SAMCLI的位置。預設位置為 /usr/local/aws-sam-cli。選取您具有寫入權限的位置。若要變更安裝位置，請選取 [本機] 並選擇您的位置。完成後按繼續。
   2. 如果您在上一個步驟中沒有選擇「僅為我安裝」的選項，請選取「變更安裝位置」>「僅為我安裝」，然後按「繼續」。
   3. 按安裝。
5. AWS SAMCLI將會安裝和安裝成功的訊息將會顯示。按「關閉」。

若要建立符號連結

- 若要 AWS SAMCLI使用指sam令叫用，您必須手動建立 AWS SAMCLI程式與您$PATH的. 通過修改並運行以下命令來創建符號鏈接：

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

  - *sudo* — 如果您的使用者具有寫入權限$PATH，sudo則不需要。否則，sudo 是必要的。
  - 路*##* — 安裝程式的 AWS SAMCLI路徑。例如 /Users/myUser/Desktop。
  - *path-to-symlink-directory*— 您的$PATH環境變數。預設位置為 /usr/local/bin。

驗證安裝成功

- 請執行下列指令，確認 AWS SAMCLI已正確安裝，且符號連結已設定：

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

Command line - All users

若要下載並安裝 AWS SAMCLI

> **Note**
>
> 如果您之前已安裝 AWS SAMCLI過Homebrew或pip，則需要先將其解除安裝。如需說明，請參閱解除安裝 AWS SAMCLI。

1. pkg將 macOS 下載到您選擇的目錄：
   - 對於運行英特爾處理器的 Mac 電腦，請選擇 x86_64 aws-sam-cli-macos-x86_64.pkg
   - 對於運行蘋果矽晶片的 Mac 電腦，請選擇 aws-sam-cli-macos-arm64.pkg

   > **Note**
   >
   > 您可以選擇在安裝之前驗證安裝程式的完整性。如需說明，請參閱可選：驗證 AWS SAMCLI安裝程式的完整性。

2. 修改並執行安裝指令碼：

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

> **Note**
>
> 若要 AWS SAMCLI使用sam指令叫用，安裝程式會在/usr/local/bin/sam和/usr/local/aws-sam-cli/sam之間自動建立符號連結。

驗證安裝成功

- 請執行下列指令，確認 AWS SAMCLI已正確安裝，且符號連結已設定：

```
$ which sam
```

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

Command line - Current user

若要下載並安裝 AWS SAMCLI

> **Note**
>
> 如果您之前已安裝 AWS SAMCLI過Homebrew或pip，則需要先將其解除安裝。如需說明，請參閱解除安裝 AWS SAMCLI。

1. pkg將 macOS 下載到您選擇的目錄：

   - 對於運行英特爾處理器的 Mac 電腦，請選擇 x86_64 aws-sam-cli-macos-x86_64.pkg
   - 對於運行蘋果矽晶片的 Mac 電腦，請選擇 aws-sam-cli-macos-arm64.pkg

   > **Note**
   >
   > 您可以選擇在安裝之前驗證安裝程式的完整性。如需說明，請參閱可選：驗證 AWS SAMCLI安裝程式的完整性。

2. 決定您具有寫入權限的安裝目錄。然後，使用範本建立xml檔案並加以修改，以反映您的安裝目錄。目錄必須已存在。

   例如，如果您取代*path-to-my-directory*為/Users/myUser/Desktop，則aws-sam-cli程式資料夾將會安裝在該處。

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

```
      <string>path-to-my-directory</string>
      <key>choiceIdentifier</key>
      <string>default</string>
    </dict>
  </array>
</plist>
```

3. 儲存xml檔案並執行下列命令來確認檔案是否有效：

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

輸出應顯示將應用於 AWS SAMCLI程序的首選項。

4. 執行下列指令以安裝 AWS SAMCLI：

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

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

若要建立符號連結

- 若要 AWS SAMCLI使用指sam令叫用，您必須手動建立 AWS SAMCLI程式與您$PATH的. 通過修改並運行以下命令來創建符號鏈接：

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

  - *sudo* — 如果您的使用者具有寫入權限$PATH，sudo則不需要。否則，sudo 是必要的。
  - 路*##* — 安裝程式的 AWS SAMCLI路徑。例如 /Users/myUser/Desktop。
  - *path-to-symlink-directory*— 您的$PATH環境變數。預設位置為 /usr/local/bin。

驗證安裝成功

- 請執行下列指令，確認 AWS SAMCLI已正確安裝，且符號連結已設定：

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

Windows

視窗安裝程序（MSI）文件是 Windows 操作系統的軟件包安裝程序文件。

請依照下列步驟 AWS SAMCLI使用 MSI 檔案進行安裝。

1. 下載 AWS SAMCLI 64 位元。

   > **Note**
   >
   > 如果您使用 32 位元版本的 Windows，請參閱AWS SAMCLI在 32 位上安裝 Windows。

2. (選擇性) 您可以在安裝之前驗證安裝程式的完整性。如需說明，請參閱可選：驗證 AWS SAMCLI 安裝程式的完整性。
3. 驗證安裝。

   完成安裝後，通過打開新的命令提示符或 PowerShell 提示符進行驗證。您應該能夠sam從命令行調用。

   ```
   sam --version
   ```

   成功安裝之後 AWS SAMCLI，您應該會看到如下所示的輸出：

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

4. 啟用長路徑 (僅限視窗 10 及更新版本)。

> **Important**
>
> AWS SAMCLI可能會與超過 Windows 最大路徑限制的文件路徑進行交互。`sam init`由於 Windows 10 的MAX_PATH限制，這可能會在運行時導致錯誤。若要解決這個問題，必須設定新的長路徑行為。

若要啟用長路徑，請參閱 Microsoft 視窗應用程式開發文件中的「啟用 Windows 10、1607 版及更新版本」中的「啟用長路徑」。

5. 安裝 Git。

   若要使用`sam init`命令下載範例應用程式，您也必須安裝 Git。如需指示，請參閱安裝 Git。

排解安裝錯誤

Linux

碼頭錯誤：「無法連接到 Docker 守護進程。docker 守護程序是否在此主機上運行？」

在某些情況下，若要提供存取 Docker 精靈的權限，您可能必須重新啟動執行個體。`ec2-user`如果您收到此錯誤訊息，請嘗試重新啟動執行個體。

外殼錯誤：「找不到命令」

如果您收到此錯誤，則 shell 無法在路徑中找到可 AWS SAMCLI執行文件。驗證您安裝 AWS SAMCLI可執行檔的目錄位置，然後確認目錄位於您的路徑上。

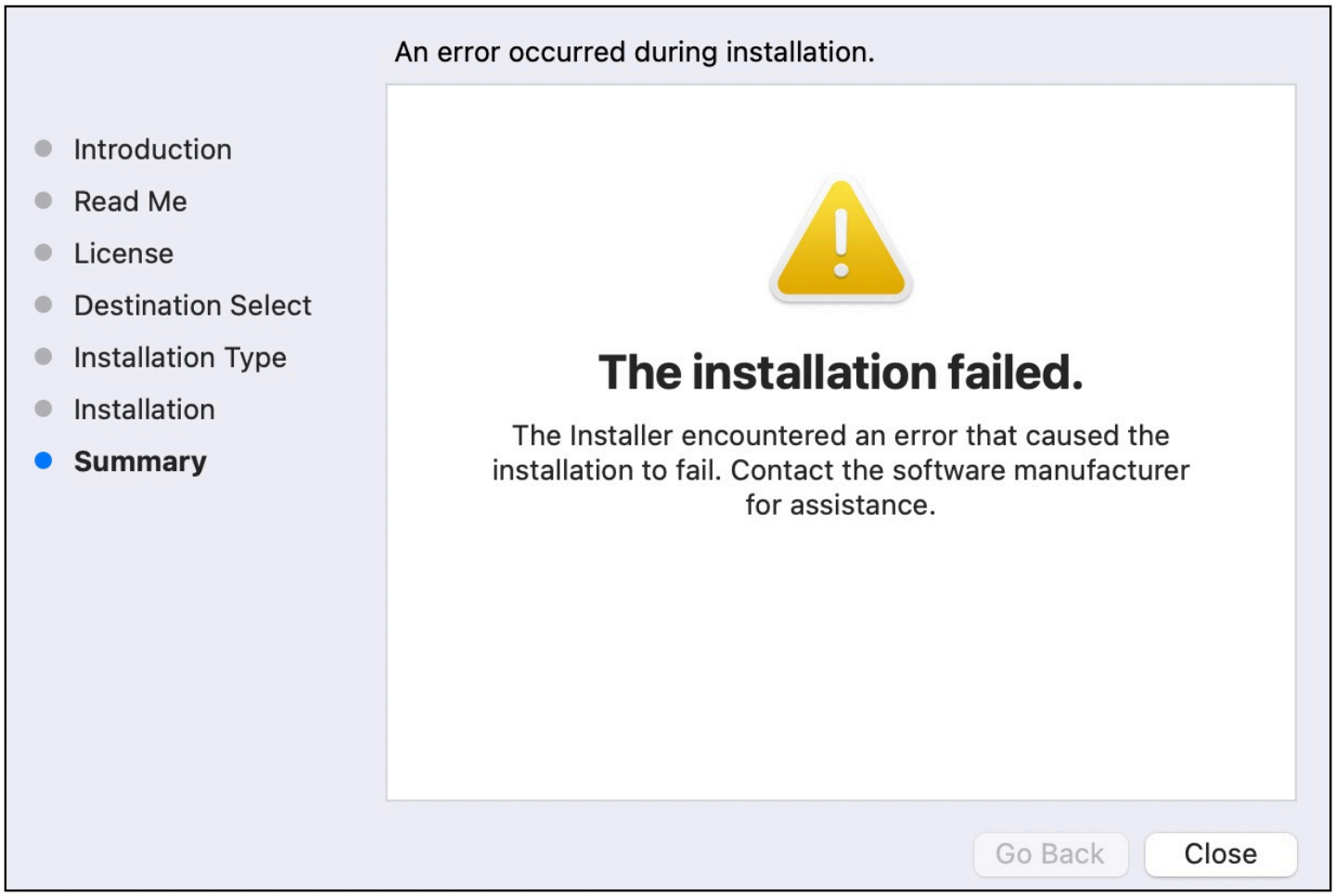
AWS SAMCLI錯誤：「/lib64 /libc.so.6：找不到版本（需要通過/usr /本地/ /dist/ libz.so.1）」aws-sam-cli

如果您收到此錯誤訊息，表示您使用的是不受支援的 Linux 版本，且內建 glibc 版本已過期。請嘗試下列其中一種方法：

- 將你的 Linux 主機升級到最近發行的 CentOS、軟呢、Ubuntu 或 Amazon Linux 2 的 64 位元版本。
- 請按照的指示操作安裝 AWS SAMCLI。

macOS

安裝失敗



如果您要 AWS SAMCLI為您的使用者安裝，並選取了您沒有寫入權限的安裝目錄，則可能會發生此錯誤。請嘗試下列其中一種方法：

1. 選取您具有寫入權限的其他安裝目錄。
2. 刪除安裝程式。然後，下載並再次運行它。

後續步驟

若要深入瞭解 AWS SAMCLI並開始建置您自己的無伺服器應用程式，請參閱下列內容：

- 教學課程：部署 Hello World 應用程式— 下載、建置和部署基本無伺服器應用程式的tep-by-step說明。
- 完整的 AWS SAM 工作坊 — 旨在教您 AWS SAM 提供許多主要功能的工作坊。

- AWS SAM 範例應用程式和模式 — 您可以進一步嘗試的社群作者提供的範例應用程式和模式。

可選：驗證 AWS SAMCLI安裝程式的完整性

使用套件安裝程式安裝 AWS Serverless Application Model 命令列介面 (AWS SAMCLI) 時，您可以在安裝之前驗證其完整性。這是選擇性的步驟，但強烈建議您使用。

您可以使用以下兩種驗證選項：

- 驗證套件安裝程式簽章檔案。
- 驗證套件安裝程式雜湊值。

如果您的平台可用，我們建議您驗證簽名檔案選項。此選項提供額外的安全層，因為金鑰值會在此發佈，並與我們的GitHub儲存庫分開管理。

主題

- 驗證安裝程式簽章檔
- 驗證哈希值

驗證安裝程式簽章檔

Linux

arm64-命令行安裝程序

AWS SAM 使用 GnuPG 來簽署 AWS SAMCLI .zip 安裝程式。驗證是按照以下步驟執行：

1. 使用主要公開金鑰驗證簽署者的公開金鑰。
2. 使用簽署者公開金鑰來驗證 AWS SAMCLI套件安裝程式。

驗證簽署者公開金鑰的完整性

1. 複製主要公開金鑰，並以.txt檔案形式儲存到本機電腦。例如 *primary-public-key.txt*。

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2.0.22 (GNU/Linux)

mQINBGRuSzMBEADsqiwOy78w7F4+sshaMFRIwRGNRm94p5Qey2KMZBxekFtoryVD
```

```
D9jEOnvupx4tvhfBHz5EcUHCEOdl4MTqdBy6vVAshozgxVb9RE8JpECn5lw7XC69
4Y7Gy1TKKQMEWtDXElkGxIFdUWvWjSnPlzfnoXwQYGeE93CUS3h5dImP22Yk1Ct6
eGGhlcbg1X4L8EpFMj7GvcsU8f7ziVI/PyC1Xwy39Q8/I67ip5eU5ddxO/xHqrbL
YC7+8pJPbRMej2twT2LrcpWWYAbprMtRoa6WfE0/thoo3xhHpIMHdPfAA86ZNGIN
kRLjGUg7jnPTRW4Oin3pCc8nT4Tfc1QERkHm641gTC/jUvpmQsM6h/FUVP2i5iE/
JHpJcMuL2Mg6zDo3x+3gTCf+Wqz3rZzxB+wQT3yryZs6efcQy7nROiRxYBxCSXX0
2cNYzsYLb/bYaW8yqWIHD5IqKhw269gp2E5Khs60zgS3CorMb5/xHgXjUCVgcu8a
a8ncdf9fjl3WS5p0ohetPbO2ZjWv+MaqrZOmUIgKbA4RpWZ/fU97P5BW9ylwmIDB
sWy0cMxg8MlvSdLytPieogaM0qMg3u5qXRGBr6Wmevkty0qgnmpGGc5zPiUbtOE8
CnFFqyxBpj5IOnG0KZGVihvn+iRxrv6GO7WWO92+Dc6m94U0EEiBR7QiOwARAQAB
tDRBV1MgU0FNIENMSSBQcmltYXJ5IDxhd3Mtc2FtLWNsaS1wcmltYXJ5QGFtYXpv
bi5jb20+iQI/BBMBCQApBQJkbkszAhsvBQkHhM4ABwsJCAcDAgEGFQgCCQoLBBYC
AwECHgECF4AACgkQQv1fenOtiFqTuhAAzi5+ju5UVOWqHKevOJSO08T4QB8HcqAE
SVO3mY6/j29knkcL8ubZP/DbpV7QpHPI2PB5qSXsiDTP3IYPbeY78zHSDjljaIK3
njJLMScFeGPyfPpwMsuY4nzrRIgAtXShPA8N/k4ZJcafnpNqKj7QnPxiC1KaIQWm
pOtvb8msUF3/s0UTa5Ys/lNRhVC0eGg32ogXGdojZA2kHZWdm9udLo4CDrDcrQT7
NtDcJASapXSQL63XfAS3snEc4e1941YxcjfYZ33rel8K9juyDZfi1slWR/L3AviI
QFIaqSHzyOtP1oinUkoVwL8ThevKD3Ag9CZflZLzNCV7yqlF8RlhEZ4zcE/3s9El
WzCFsozb5HfE1AZonmrDh3SyOEIBMcS6vG5dWnvJrAuSYv2rX38++K5Pr/MIAfOX
DOI1rtA+XDsHNv9lSwSy0lt+iClawZANO9IXCiN1rOYcVQlwzDFwCNWDgkwdOqS0
gOA2f8NF9lE5nBbeEuYquoOl1Vy8+ICbgOFs9LoWZlnVh7/RyY6ssowiU9vGUnHI
L8f9jqRspIz/Fm3JD86ntZxLVGkeZUz62FqErdohYfkFIVcv7GONTEyrz5HLlnpv
FJ0MR0HjrMrZrnOVZnwBKhpbLocTsH+3t5It4ReYEX0f1DIOL/KRwPvjMvBVkXY5
hblRVDQoOWc=
=d9oG
-----END PGP PUBLIC KEY BLOCK-----
```

2. 將主要公開金鑰匯入您的金鑰圈。

```
$ gpg --import primary-public-key.txt

gpg: directory `/home/.../.gnupg' created
gpg: new configuration file `/home/.../.gnupg/gpg.conf' created
gpg: WARNING: options in `/home/.../.gnupg/gpg.conf' are not yet active during this
 run
gpg: keyring `/home/.../.gnupg/secring.gpg' created
gpg: keyring `/home/.../.gnupg/pubring.gpg' created
gpg: /home/.../.gnupg/trustdb.gpg: trustdb created
gpg: key 73AD885A: public key "AWS SAM CLI Primary <aws-sam-cli-
primary@amazon.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
```

3. 複製簽署者公開金鑰，並以.txt檔案形式儲存至本機電腦。例如 *signer-public-key.txt*。

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2.0.22 (GNU/Linux)

mQINBGRtS20BEAC7GjaAwverrB1zNEu2q3EGI6HC37WzwL5dy30f4LirZOWS3piK
oKfTqPjXPrLCf1GL2mMqUSgSnpEbPNXuvWTW1CfSnnjwuH8ZqbvvUQyHJwQyYpKm
KMwb+8V0bzzQkMzDVqolYQCi5XyGpAuo3wroxXSzG6r/mIhbiq3aRnL+2lo4XOYk
r7q9bhBqbJhzjkm7N62PhPWmi/+/EGdEBakA1pReE+cKjP2UAp5L6CPShQl2fRKL
9BumitNfFHHs1JZgZSCCruiWny3XkUaXUEMfyoE9nNbfqNvuqV2KjWguZCXASgz2
ZSPF4DTVIBMfP+xrZGQSWdGU/67QdysDQW81TbFOjK9ZsRwwGC4kbg/K98IsCNHT
ril5RZbyr8pw3fw7jYjjI2ElAacRWp53iRzvutm5AruPpLfoKDQ/tKzBUYItBwlu
Z/diKgcqtw7xDlyqNyTN8xFPFqMO2I8IsZ2Pdl131htdFiZMiin1RQG9pV9p2vHS
eQVY2uKCnvnA6vFCQYKXP7p0IwReuPNzDvECUsidw8VTakTqZsANT/bU17e4KuKn
+JgbNrKOasJX37sDb/9ruysozLvy78ozYKJDLmC3yoRQ8DhEjviT4cnjORgNmvnZ
0a5AA/DJPQW4buRrXdxu+fITzBxQn2+GO/iDNCxtJaq5SYVBKjTmTWPUJwARAQAB
tDBBV1MgU0FNIENMSSBUZWFtIDxhd3Mtc2FtLWNsaS1zaWduZXJAYW1hem9uLmNv
bT6JAj8EEwEJACkFAmRtS20CGy8FCQPCZwAHCwkIBwMCAQYVCAIJCgsEFgIDAQIe
AQIXgAAKCRDHoF9D/grd+lE4D/4kJW65He2LNsbLTta7lcGfsEXCf4zgIvkytS7U
3R36zMD8IEyWJjlZ+aPkIP8/jFJrFl4pVHbU7vX85Iut1vV7m+8BgWt25mJhnoJ9
KPjXGra9mYP+Cj8zFAcjvtl3NBAPodyfcfCTWsU3umF9ArOFICcrGCzHX2SS7wX5
h9n0vYRZxk5Qj5FsgskKAQLq33CKFAMlaqZnL5gWRvTeycSIxsyus+stX+8YBPCO
J64f7+y+MPIP1+m2njlVXg1xLEMMVa08oWccOMiakgzDev3LCrPy+wdwdn7Ut7oA
pna3DNy9aYNd2lh6vUCJeJ+Yi1Bl2jYpzLcCLKrHUmln9/rRSz7Orbg8P181kfPu
G/M7CD5FwhxP3p4+0XoGwxQefrV2jqpSnbLae7xbYJiJAhbpjWDQhuNGUbPcDmqk
aH0Q3XU8AonJ8YqaQ/q3VZ3JBiH3TbBrOXsvd59cwxYyf83aJ/WLCb2P8y75zDad
lnOP713ThF5J/Afj9HjO9waFV0Z2W2ZZe4rU2OJTAiXEtM8xsFMrc7TCUacJtJGs
u4kdBmXREcVpSz65h9ImSy2ner9qktnVVCW4mZPj63IhB37YtoLAMyz3a3R2RFNk
viEX8foOTUg1FmwHoftxZ9P91QwLoTajkDrh26ueIe45sG6Uxua2AP4Vo37cFfCj
ryV8OokCHAQQAQkABgUCZG5MWAAKCRBC/V96c62IWmglD/9idU43kW8Zy8Af1j8l
Am3lI4d9ksOleeKRZqxo/SZ5rovF32DO2nw7XRXq1+EbhgJaI3QwwOi0U0pfAMVT
4b9TdxdH+n+tqzCHh3jZqmo9sw+c9WFXYJN1hU9bLzcHXS8hOTbyoE2EuXx56ds9
L/BWCcd+LIvapw0lggFfavVx/QF4C7nBKjnJ66+xxwfgVIKR7oGlqDiHMfp9ZWh5
HhEqZo/nrNhdY0h3sczEdqC2N6eIa8mgHffHZdKudDMXIXHbgdhW9pcZXDIktVf7
j9wehsWOyYXiRgR0dz7DI26AUG4JLh5FTtx9XuSBdEsI69Jd4dJuibmgtImzbZjn
7un8DJWIyqi7Ckk96Tr4oXB9mYAXaWlR4C9j5XJhMNZgkOycuY2DADnbGmSb+1kA
ju77H4ff84+vMDwUzUt2Wwb+GjzXu2g6Wh+bWhGSirYlel+6xYrI6beu1BDCFLq+
VZFE8WggjJHpwcL7CiqadfVIQaw4HY0jQFTSdwzPWhJvYjXFOhMkyCcjssbtmB+z
/otfgySyQqThrD48RWS5GuyqCA+pK3UNmEJ11c1AXMdTn2VWInR1NOJNALQ2du3y
q8t1vMsErVOJ7pkZ50F4ef17PE6DKrXX8ilwGFyVuX5ddyt/t9J5pC3sRwHWXVZx
GXwoX75FwIEHA3n5Q7rZ69Ea6Q==
=ZIO7
-----END PGP PUBLIC KEY BLOCK-----
```

4. 將簽署者公開金鑰匯入您的金鑰圈。

```
$ gpg --import signer-public-key.txt

gpg: key FE0ADDFA: public key "AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>"
 imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
gpg: no ultimately trusted keys found
```

記下輸出中的索引鍵值。例如 *FE0ADDFA*。

5. 使用金鑰值取得並驗證簽署者的公開金鑰指紋。

```
$ gpg --fingerprint FE0ADDFA

pub   4096R/FE0ADDFA 2023-05-23 [expires: 2025-05-22]
      Key fingerprint = 37D8 BE16 0355 2DA7 BD6A  04D8 C7A0 5F43 FE0A DDFA
uid                  AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>
```

指紋應符合以下內容：

```
37D8 BE16 0355 2DA7 BD6A  04D8 C7A0 5F43 FE0A DDFA
```

如果指紋字串不相符，請勿使用 AWS SAMCLI安裝程式。透過在aws-sam-cli GitHub 存放庫中建立問題，向 AWS SAM 專案團隊升級。

6. 驗證簽署者公開金鑰的簽章：

```
$ gpg --check-sigs FE0ADDFA

pub   4096R/FE0ADDFA 2023-05-23 [expires: 2025-05-22]
uid                  AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>
sig!3        FE0ADDFA 2023-05-23  AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>
sig!         73AD885A 2023-05-24  AWS SAM CLI Primary <aws-sam-cli-
primary@amazon.com>
```

如果您看到`1 signature not checked due to a missing key`，請重複上述步驟，將主要和簽署者公開金鑰匯入金鑰圈。

您應該會看到列出主要公開金鑰和簽署者公開金鑰的金鑰值。

現在您已驗證簽署者公開金鑰的完整性，您可以使用簽署者公開金鑰來驗證 AWS SAMCLI套件安裝程式。

驗證 AWS SAMCLI套件安裝程式的完整性

1. 取得 AWS SAMCLI套件簽章檔案 — 使用下列指令下載 AWS SAMCLI套件安裝程式的簽章檔案：

```
$ wget https://github.com/aws/aws-sam-cli/releases/latest/download/aws-sam-cli-linux-arm64.zip.sig
```

2. 驗證簽名檔案 — 將下載的.sig和檔.zip案作為參數傳遞至指gpg令。以下是範例：

```
$ gpg --verify aws-sam-cli-linux-arm64.zip.sig aws-sam-cli-linux-arm64.zip
```

輸出格式應類似以下內容：

```
gpg: Signature made Tue 30 May 2023 10:03:57 AM UTC using RSA key ID FE0ADDFA
gpg: Good signature from "AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 37D8 BE16 0355 2DA7 BD6A  04D8 C7A0 5F43 FE0A DDFA
```

- 該WARNING: This key is not certified with a trusted signature!消息可以被忽略。這是因為您的個人 PGP 金鑰 (如果有的話) 與 AWS SAM CLI PGP 金鑰之間沒有信任鏈結。如需詳細資訊，請參閱信任網。
- 如果輸出包含片語BAD signature，請檢查您是否正確執行了該程序。如果您繼續收到此回應，請在aws-sam-cli GitHub 存放庫中建立問題，將問題呈報給專案 AWS SAM 團隊，並避免使用下載的檔案。

該Good signature from "AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>"消息表示簽名已驗證，您可以隨著安裝繼續前進。

x86_64-命令行安裝程序

AWS SAM 使用 GnuPG 來簽署 AWS SAMCLI .zip 安裝程式。驗證是按照以下步驟執行：

1. 使用主要公開金鑰驗證簽署者的公開金鑰。
2. 使用簽署者公開金鑰來驗證 AWS SAMCLI套件安裝程式。

驗證簽署者公開金鑰的完整性

1. 複製主要公開金鑰，並以.txt檔案形式儲存到本機電腦。例如 *primary-public-key.txt*。

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2.0.22 (GNU/Linux)

mQINBGRuSzMBEADsqiwOy78w7F4+sshaMFRIwRGNRm94p5Qey2KMZBxekFtoryVD
D9jEOnvupx4tvhfBHz5EcUHCEOdl4MTqdBy6vVAshozgxVb9RE8JpECn5lw7XC69
4Y7Gy1TKKQMEWtDXElkGxIFdUWvWjSnPlzfnoXwQYGeE93CUS3h5dImP22Yk1Ct6
eGGhlcbg1X4L8EpFMj7GvcsU8f7ziVI/PyC1Xwy39Q8/I67ip5eU5ddxO/xHqrbL
YC7+8pJPbRMej2twT2LrcpWWYAbprMtRoa6WfE0/thoo3xhHpIMHdPfAA86ZNGIN
kRLjGUg7jnPTRW4Oin3pCc8nT4Tfc1QERkHm641gTC/jUvpmQsM6h/FUVP2i5iE/
JHpJcMuL2Mg6zDo3x+3gTCf+Wqz3rZzxB+wQT3yryZs6efcQy7nROiRxYBxCSXX0
2cNYzsYLb/bYaW8yqWIHD5IqKhw269gp2E5Khs60zgS3CorMb5/xHgXjUCVgcu8a
a8ncdf9fjl3WS5p0ohetPbO2ZjWv+MaqrZOmUIgKbA4RpWZ/fU97P5BW9ylwmIDB
sWy0cMxg8MlvSdLytPieogaM0qMg3u5qXRGBr6Wmevkty0qgnmpGGc5zPiUbtOE8
CnFFqyxBpj5IOnG0KZGVihvn+iRxrv6GO7WWO92+Dc6m94U0EEiBR7QiOwARAQAB
tDRBV1MgU0FNIENMSSBQcmltYXJ5IDxhd3Mtc2FtLWNsaS1wcmltYXJ5QGFtYXpv
bi5jb20+iQI/BBMBCQApBQJkbkszAhsvBQkHhM4ABwsJCAcDAgEGFQgCCQoLBBYC
AwECHgECF4AACgkQQv1fenOtiFqTuhAAzi5+ju5UVOWqHKevOJSO08T4QB8HcqAE
SVO3mY6/j29knkcL8ubZP/DbpV7QpHPI2PB5qSXsiDTP3IYPbeY78zHSDjljaIK3
njJLMScFeGPyfPpwMsuY4nzrRIgAtXShPA8N/k4ZJcafnpNqKj7QnPxiC1KaIQWm
pOtvb8msUF3/s0UTa5Ys/lNRhVC0eGg32ogXGdojZA2kHZWdm9udLo4CDrDcrQT7
NtDcJASapXSQL63XfAS3snEc4e1941YxcjfYZ33rel8K9juyDZfi1slWR/L3AviI
QFIaqSHzyOtP1oinUkoVwL8ThevKD3Ag9CZflZLzNCV7yqlF8RlhEZ4zcE/3s9El
WzCFsozb5HfE1AZonmrDh3SyOEIBMcS6vG5dWnvJrAuSYv2rX38++K5Pr/MIAfOX
DOI1rtA+XDsHNv9lSwSy0lt+iClawZANO9IXCiN1rOYcVQlwzDFwCNWDgkwdOqS0
gOA2f8NF9lE5nBbeEuYquoOl1Vy8+ICbgOFs9LoWZlnVh7/RyY6ssowiU9vGUnHI
L8f9jqRspIz/Fm3JD86ntZxLVGkeZUz62FqErdohYfkFIVcv7GONTEyrz5HLlnpv
FJ0MR0HjrMrZrnOVZnwBKhpbLocTsH+3t5It4ReYEX0f1DIOL/KRwPvjMvBVkXY5
hblRVDQoOWc=
=d9oG
-----END PGP PUBLIC KEY BLOCK-----
```

2. 將主要公開金鑰匯入您的金鑰圈。

```
$ gpg --import primary-public-key.txt

gpg: directory `/home/.../.gnupg' created
gpg: new configuration file `/home/.../.gnupg/gpg.conf' created
gpg: WARNING: options in `/home/.../.gnupg/gpg.conf' are not yet active during this
 run
```

```
gpg: keyring `/home/.../.gnupg/secring.gpg' created
gpg: keyring `/home/.../.gnupg/pubring.gpg' created
gpg: /home/.../.gnupg/trustdb.gpg: trustdb created
gpg: key 73AD885A: public key "AWS SAM CLI Primary <aws-sam-cli-
primary@amazon.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
```

3. 複製簽署者公開金鑰，並以.txt檔案形式儲存至本機電腦。例如 *signer-public-key.txt*。

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2.0.22 (GNU/Linux)

mQINBGRtS20BEAC7GjaAwverrB1zNEu2q3EGI6HC37WzwL5dy30f4LirZOWS3piK
oKfTqPjXPrLCf1GL2mMqUSgSnpEbPNXuvWTW1CfSnnjwuH8ZqbvvUQyHJwQyYpKm
KMwb+8V0bzzQkMzDVqolYQCi5XyGpAuo3wroxXSzG6r/mIhbiq3aRnL+2lo4XOYk
r7q9bhBqbJhzjkm7N62PhPWmi/+/EGdEBakA1pReE+cKjP2UAp5L6CPShQl2fRKL
9BumitNfFHHs1JZgZSCCruiWny3XkUaXUEMfyoE9nNbfqNvuqV2KjWguZCXASgz2
ZSPF4DTVIBMfP+xrZGQSWdGU/67QdysDQW81TbFOjK9ZsRwwGC4kbg/K98IsCNHT
ril5RZbyr8pw3fw7jYjjI2ElAacRWp53iRzvutm5AruPpLfoKDQ/tKzBUYItBwlu
Z/diKgcqtw7xDlyqNyTN8xFPFqMO2I8IsZ2Pdl131htdFiZMiin1RQG9pV9p2vHS
eQVY2uKCnvnA6vFCQYKXP7p0IwReuPNzDvECUsidw8VTakTqZsANT/bU17e4KuKn
+JgbNrKOasJX37sDb/9ruysozLvy78ozYKJDLmC3yoRQ8DhEjviT4cnjORgNmvnZ
0a5AA/DJPQW4buRrXdxu+fITzBxQn2+GO/iDNCxtJaq5SYVBKjTmTWPUJwARAQAB
tDBBV1MgU0FNIENMSSBUZWFtIDxhd3Mtc2FtLWNsaS1zaWduZXJAYW1hem9uLmNv
bT6JAj8EEwEJACkFAmRtS20CGy8FCQPCZwAHCwkIBwMCAQYVCAIJCgsEFgIDAQIe
AQIXgAAKCRDHoF9D/grd+lE4D/4kJW65He2LNsbLTta7lcGfsEXCf4zgIvkytS7U
3R36zMD8IEyWJjlZ+aPkIP8/jFJrFl4pVHbU7vX85Iut1vV7m+8BgWt25mJhnoJ9
KPjXGra9mYP+Cj8zFAcjvtl3NBAPodyfcfCTWsU3umF9ArOFICcrGCzHX2SS7wX5
h9n0vYRZxk5Qj5FsgskKAQLq33CKFAMlaqZnL5gWRvTeycSIxsyus+stX+8YBPCO
J64f7+y+MPIP1+m2njlVXg1xLEMMVa08oWccOMiakgzDev3LCrPy+wdwdn7Ut7oA
pna3DNy9aYNd2lh6vUCJeJ+Yi1Bl2jYpzLcCLKrHUmln9/rRSz7Orbg8P181kfPu
G/M7CD5FwhxP3p4+0XoGwxQefrV2jqpSnbLae7xbYJiJAhbpjWDQhuNGUbPcDmqk
aH0Q3XU8AonJ8YqaQ/q3VZ3JBiH3TbBrOXsvd59cwxYyf83aJ/WLCb2P8y75zDad
lnOP713ThF5J/Afj9HjO9waFV0Z2W2ZZe4rU2OJTAiXEtM8xsFMrc7TCUacJtJGs
u4kdBmXREcVpSz65h9ImSy2ner9qktnVVCW4mZPj63IhB37YtoLAMyz3a3R2RFNk
viEX8foOTUg1FmwHoftxZ9P91QwLoTajkDrh26ueIe45sG6Uxua2AP4Vo37cFfCj
ryV8OokCHAQQAQkABgUCZG5MWAAKCRBC/V96c62IWmglD/9idU43kW8Zy8Af1j8l
Am3lI4d9ksOleeKRZqxo/SZ5rovF32DO2nw7XRXq1+EbhgJaI3QwwOi0U0pfAMVT
4b9TdxdH+n+tqzCHh3jZqmo9sw+c9WFXYJN1hU9bLzcHXS8hOTbyoE2EuXx56ds9
L/BWCcd+LIvapw0lggFfavVx/QF4C7nBKjnJ66+xxwfgVIKR7oGlqDiHMfp9ZWh5
HhEqZo/nrNhdY0h3sczEdqC2N6eIa8mgHffHZdKudDMXIXHbgdhW9pcZXDIktVf7
j9wehsWOyYXiRgR0dz7DI26AUG4JLh5FTtx9XuSBdEsI69Jd4dJuibmgtImzbZjn
7un8DJWIyqi7Ckk96Tr4oXB9mYAXaWlR4C9j5XJhMNZgkOycuY2DADnbGmSb+1kA
```

```
ju77H4ff84+vMDwUzUt2Wwb+GjzXu2g6Wh+bWhGSirYlel+6xYrI6beu1BDCFLq+
VZFE8WggjJHpwcL7CiqadfVIQaw4HY0jQFTSdwzPWhJvYjXFOhMkyCcjssbtmB+z
/otfgySyQqThrD48RWS5GuyqCA+pK3UNmEJ11c1AXMdTn2VWInR1NOJNALQ2du3y
q8t1vMsErVOJ7pkZ50F4ef17PE6DKrXX8ilwGFyVuX5ddyt/t9J5pC3sRwHWXVZx
GXwoX75FwIEHA3n5Q7rZ69Ea6Q==
=ZIO7
-----END PGP PUBLIC KEY BLOCK-----
```

4. 將簽署者公開金鑰匯入您的金鑰圈。

```
$ gpg --import signer-public-key.txt

gpg: key FE0ADDFA: public key "AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>"
 imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
gpg: no ultimately trusted keys found
```

記下輸出中的索引鍵值。例如 *FE0ADDFA*。

5. 使用金鑰值取得並驗證簽署者的公開金鑰指紋。

```
$ gpg --fingerprint FE0ADDFA

pub   4096R/FE0ADDFA 2023-05-23 [expires: 2025-05-22]
      Key fingerprint = 37D8 BE16 0355 2DA7 BD6A  04D8 C7A0 5F43 FE0A DDFA
uid                  AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>
```

指紋應符合以下內容：

```
37D8 BE16 0355 2DA7 BD6A  04D8 C7A0 5F43 FE0A DDFA
```

如果指紋字串不相符，請勿使用 AWS SAMCLI安裝程式。透過在aws-sam-cli GitHub 存放庫中建立問題，向 AWS SAM 專案團隊升級。

6. 驗證簽署者公開金鑰的簽章：

```
$ gpg --check-sigs FE0ADDFA

pub   4096R/FE0ADDFA 2023-05-23 [expires: 2025-05-22]
uid                  AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>
sig!3        FE0ADDFA 2023-05-23  AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>
```

```
sig!         73AD885A 2023-05-24  AWS SAM CLI Primary <aws-sam-cli-
primary@amazon.com>
```

如果您看到`1 signature not checked due to a missing key`，請重複上述步驟，將主要和簽署者公開金鑰匯入金鑰圈。

您應該會看到列出主要公開金鑰和簽署者公開金鑰的金鑰值。

現在您已驗證簽署者公開金鑰的完整性，您可以使用簽署者公開金鑰來驗證 AWS SAMCLI套件安裝程式。

驗證 AWS SAMCLI套件安裝程式的完整性

1. 取得 AWS SAMCLI套件簽章檔案 — 使用下列指令下載 AWS SAMCLI套件安裝程式的簽章檔案：

```
$ wget https://github.com/aws/aws-sam-cli/releases/latest/download/aws-sam-cli-
linux-x86_64.zip.sig
```

2. 驗證簽名檔案 — 將下載的`.sig`和檔`.zip`案作為參數傳遞至指gpg令。以下是範例：

```
$ gpg --verify aws-sam-cli-linux-x86_64.zip.sig aws-sam-cli-linux-x86_64.zip
```

輸出格式應類似以下內容：

```
gpg: Signature made Tue 30 May 2023 10:03:57 AM UTC using RSA key ID FE0ADDFA
gpg: Good signature from "AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 37D8 BE16 0355 2DA7 BD6A  04D8 C7A0 5F43 FE0A DDFA
```

- 該`WARNING: This key is not certified with a trusted signature!`消息可以被忽略。這是因為您的個人 PGP 金鑰 (如果有的話) 與 AWS SAM CLI PGP 金鑰之間沒有信任鏈結。如需詳細資訊，請參閱信任網。
- 如果輸出包含片語`BAD signature`，請檢查您是否正確執行了該程序。如果您繼續收到此回應，請在aws-sam-cli GitHub 存放庫中建立問題，將問題呈報給專案 AWS SAM 團隊，並避免使用下載的檔案。

該Good signature from "AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>"消息表示簽名已驗證，您可以隨著安裝繼續前進。

macOS

GUI 和命令行安裝程序

您可以使用pkgutil工具或手動驗證 AWS SAMCLI套件安裝程式簽章檔案的完整性。

若要使用 Pkgutil 進行驗證

1. 執行下列命令，提供本機電腦上已下載安裝程式的路徑：

```
$ pkgutil --check-signature /path/to/aws-sam-cli-installer.pkg
```

以下是範例：

```
$ pkgutil --check-signature /Users/user/Downloads/aws-sam-cli-macos-arm64.pkg
```

2. 從輸出中，找到SHA256 fingerprint的Developer ID Installer: AMZN Mobile LLC。以下是範例：

```
Package "aws-sam-cli-macos-arm64.pkg":
   Status: signed by a developer certificate issued by Apple for distribution
   Notarization: trusted by the Apple notary service
   Signed with a trusted timestamp on: 2023-05-16 20:29:29 +0000
   Certificate Chain:
    1. Developer ID Installer: AMZN Mobile LLC (94KV3E626L)
       Expires: 2027-06-28 22:57:06 +0000
       SHA256 Fingerprint:
           49 68 39 4A BA 83 3B F0 CC 5E 98 3B E7 C1 72 AC 85 97 65 18 B9 4C
           BA 34 62 BF E9 23 76 98 C5 DA
       ------------------------------------------------------------------------
    2. Developer ID Certification Authority
       Expires: 2031-09-17 00:00:00 +0000
       SHA256 Fingerprint:
           F1 6C D3 C5 4C 7F 83 CE A4 BF 1A 3E 6A 08 19 C8 AA A8 E4 A1 52 8F
           D1 44 71 5F 35 06 43 D2 DF 3A
       ------------------------------------------------------------------------
    3. Apple Root CA
       Expires: 2035-02-09 21:40:36 +0000
```

```
        SHA256 Fingerprint:
            B0 B1 73 0E CB C7 FF 45 05 14 2C 49 F1 29 5E 6E DA 6B CA ED 7E 2C
            68 C5 BE 91 B5 A1 10 01 F0 24
```

3. Developer ID Installer: AMZN Mobile LLC SHA256 fingerprint應該符合下列值：

```
49 68 39 4A BA 83 3B F0 CC 5E 98 3B E7 C1 72 AC 85 97 65 18 B9 4C BA 34 62 BF E9 23
 76 98 C5 DA
```

如果指紋字串不相符，請勿使用 AWS SAMCLI安裝程式。透過在aws-sam-cli GitHub 存放庫中建立問題，向 AWS SAM 專案團隊升級。如果指紋字串不相符，您可以使用套件安裝程式繼續前進。

手動驗證套件安裝程式

- 請參閱 Apple 支援網站上的「如何驗證手動下載的 Apple 軟件更新」的真實性。

Windows

AWS SAMCLI安裝程式會封裝為Windows作業系統的MSI檔案。

驗證安裝程式的完整性

1. 右鍵單擊安裝程序並打開屬性窗口。
2. 選擇 數位簽章 索引標籤。
3. 從「簽名清單」中選擇「Amazon Web Services 公司」，然後選擇「詳細資料」。
4. 選擇 General (一般) 索引標籤 (如果尚未選取)，然後選擇 View Certificate (檢視憑證)。
5. 選擇 [詳細資料] 索引標籤，然後在 [顯示] 下拉式清單中選擇 [全部] (如果尚未選取)。
6. 向下捲動到看見 Thumbprint (指紋) 欄位為止，然後選擇 Thumbprint (指紋)。這會在下方的視窗中顯示整個指紋值。
7. 將指紋值與下列值相符。如果值匹配，請繼續安裝。如果沒有，請在aws-sam-cli GitHub 存放庫中建立問題，以呈報給 AWS SAM 專案團隊。

```
c011d416e99a1142c0e0235118ef64c2681f3db9
```

驗證哈希值

Linux

x86_64-命令行安裝程序

使用下列指令產生雜湊值，以驗證已下載安裝程式檔案的完整性與真實性：

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

輸出應如下列範例所示：

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

將 64 個字元的 SHA-256 雜湊值與上GitHub的版本說明中所需 AWS SAMCLI版本的雜湊值進AWS SAMCLI行比較。

macOS

GUI 和命令行安裝程序

使用下列命令產生雜湊值，以驗證下載安裝程式的完整性與真實性：

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

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

將 64 個字元的 SHA-256 雜湊值與AWS SAMCLI版本說明GitHub儲存庫中的對應值進行比較。

教學課程：部署 Hello World 應用程式

在本自學課程中，您將使用 AWS Serverless Application Model 指令行介面 (AWS SAMCLI) 完成以下作業：

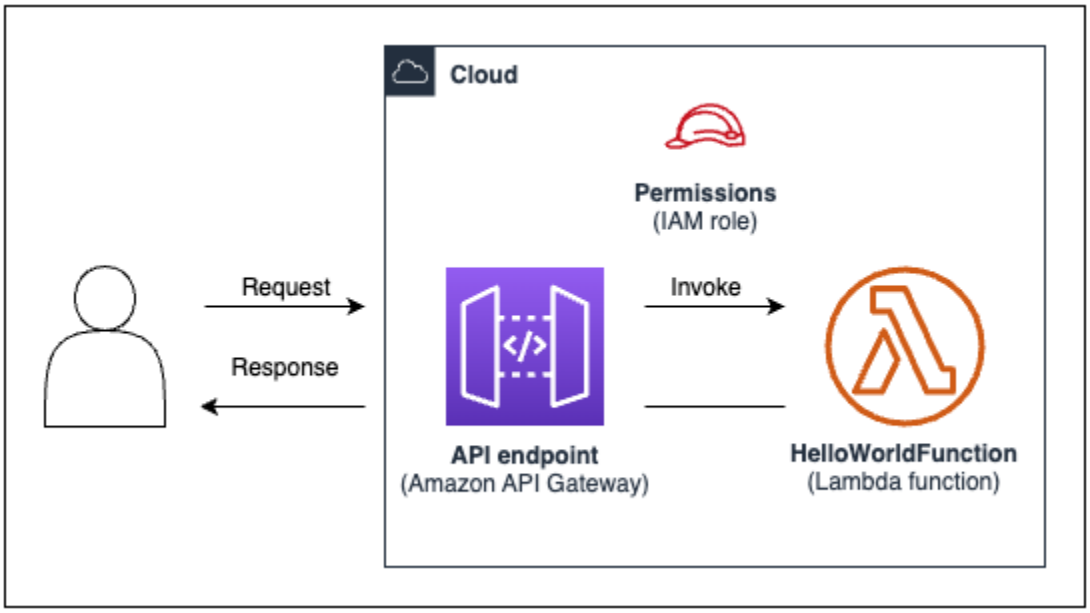
- 初始化、建置和部署範例 Hello World 應用程式。
- 進行本機變更並同步至 AWS CloudFormation。
- 在中測試您的應用程式 AWS 雲端。

- 或者，在您的開發主機上執行本機測試。
- 從中刪除範例應用程式 AWS 雲端。

您好世界應用程式範例會實作基本的 API 後端程式。它由以下資源組成：

- Amazon API Gateway — 您將用來調用函數的 API 端點。
- AWS Lambda— 處理 HTTP API GET 要求並傳回`hello world`訊息的函數。
- AWS Identity and Access Management (IAM) 角色 — 為服務佈建許可以安全地互動。

下圖顯示此應用程式的組件：



主題

- 故障診斷
- 進一步了解

必要條件

確認您已完成下列項目：

- AWS SAM 前提
- 安裝 AWS SAMCLI

步驟 1：初始化範例 Hello World 應用程式

在此步驟中，您將使用在 AWS SAMCLI本機電腦上建立範例 Hello World 應用程式專案。

若要初始化範例 Hello World 應用程式

1. 在命令行中，從您選擇的起始目錄運行以下命令：

```
$ sam init
```

> **Note**
>
> 此指令會初始化您的無伺服器應用程式，並建立專案目錄。該目錄將包含多個文件和文件夾。最重要的文件是`template.yaml`. 這是您的 AWS SAM 範本。您的 python 版本必須與命sam init令創建的`template.yaml`文件中列出的 python 版本匹配。

2. AWS SAMCLI將引導您完成初始化新的應用程式。設定下列項目：
   1. 選取「AWS 快速入門範本」以選擇起始範本。
   2. 選擇你好世界範例範本並下載。
   3. 使用Python執行階段和`zip`套件類型。
   4. 在本教學課程中，請選擇退出 AWS X-Ray 追蹤。若要深入瞭解，請參閱什麼是 AWS X-Ray？在AWS X-Ray 開發人員指南中。
   5. 在本教學課程中，請選擇退出使 CloudWatch 用 Amazon 應用程式洞察進行監控。若要進一步了解，請參閱 Amazon CloudWatch 使用者指南中的 Amazon 應用 CloudWatch 程式深入解析。

6. 在本教學課程中，請選擇不在 Lambda 函數上設定 JSON 格式的結構化記錄。
7. 將您的應用程序命名為 sam-app。

若要使用 AWS SAMCLI互動式流程：

- 括號 ([ ]) 表示預設值。將您的答案留空，以選取預設值。
- 輸入**y**代表 「是」，**n**表示「否」。

以下是`sam init`互動式流程的範例：

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

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

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

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

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

```
Would you like to set Structured Logging in JSON format on your Lambda functions?
 [y/N]: ENTER

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

3. 會下 AWS SAMCLI載您的起始範本，並在您的本機電腦上建立應用程式專案目錄結構。以下是 AWS SAMCLI輸出的範例：

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

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

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


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

4. 從命令列移至新建立的sam-app目錄。下面是什麼 AWS SAMCLI已創建的一個例子：

```
$ cd sam-app

$ tree

### README.md
### __init__.py
### events
#   ### event.json
```

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

6 directories, 14 files
```

要突出顯示的一些重要文件：

- hello_world/app.py— 包含您的 Lambda 函數程式碼。
- hello_world/requirements.txt— 包含您的 Lambda 函數需要的任何Python相依性。
- samconfig.toml— 應用程式的組態檔案，儲存使用的預設參數 AWS SAMCLI。
- template.yaml— 包含應用程式基礎結構程式碼的 AWS SAM 範本。

您現在在本機電腦上擁有完全撰寫的無伺服器應用程式！

步驟 2：建置您的應用程式

在此步驟中，您可 AWS SAMCLI以使用建置應用程式並準備部署。當您建置時， AWS SAMCLI會建立一個.aws-sam目錄，並在該處組織函式相依性、專案程式碼和專案檔案。

建立您的應用程式

- 在命令行中，從sam-app項目目錄中運行以下命令：

```
$ sam build
```

> **Note**
>
> 如果您的本地計算機Python上沒有，請改用sam build --use-container 命令。 AWS SAMCLI將會建立一個Docker容器，其中包含函式的執行階段和相依性。此指令需要Docker在本機電腦上使用。若要安裝Docker，請參閱安裝 Docker。

以下是 AWS SAMCLI輸出的範例：

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

Build Succeeded

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

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

以下是 AWS SAM CLI 建立之目`.aws-sam`錄的簡短範例：

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

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

要突出顯示的一些重要文件：

- `build/HelloWorldFunction`— 包含您的 Lambda 函數程式碼和相依性。 AWS SAMCLI會為應用程式中的每個函數建立一個目錄。
- `build/template.yaml`— 包含部署 AWS CloudFormation 時參考的 AWS SAM 範本副本。
- `build.toml`— 儲存建置和部署應用程式 AWS SAMCLI時所參考的預設參數值的組態檔案。

您現在可以將應用程式部署到 AWS 雲端.

步驟 3：將您的應用程式部署到 AWS 雲端

> **Note**
>
> 此步驟需要設定 AWS 認證。如需詳細資訊，請參閱 AWS SAM 前提 中的 步驟 5：使用設 AWS CLI 定 AWS 認證。

在此步驟中，您可 AWS SAMCLI以使用將應用程式部署到 AWS 雲端. AWS SAMCLI將執行以下操作：

- 引導您完成設定用於部署的應用程式設定。
- 將您的應用程式檔案上傳到亞馬遜簡易儲存服務 (Amazon S3)。
- 將 AWS SAM 範本轉換為 AWS CloudFormation 範本。然後，它將您的模板上傳到 AWS CloudFormation 服務以佈建您的 AWS 資源。

部署您的 應用程式

1. 在命令行中，從`sam-app`項目目錄中運行以下命令：

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

2. 遵循 AWS SAMCLI互動式流程來設定您的應用程式設定。設定下列項目：

1. AWS CloudFormation 堆疊名稱 — 堆疊是您可以作為單一單元來管理的 AWS 資源集合。若要深入瞭解，請參閱使用AWS CloudFormation 者指南中的使用堆疊。
2. AWS 區域要將您的 AWS CloudFormation 堆疊部署到. 如需詳細資訊，請參閱AWS CloudFormation 使用指南中的AWS CloudFormation 端點。
3. 在本教學課程中，請在部署前選擇退出確認變更。
4. 允許 IAM 角色建立 — 這可讓您 AWS SAM 建立 API Gateway 資源和 Lambda 函數資源進行互動所需的 IAM 角色。
5. 在本教學課程中，請選擇不停用復原功能。
6. HelloWorldFunction 未定義授權即允許 — 顯示此訊息是因為您的 API Gateway 端點設定為可公開存取，未經授權。由於這是您 Hello World 應用程式的預定組態，因此請允許 AWS SAMCLI繼續。如需有關配置授權的詳細資訊，請參閱使用 AWS SAM 範本控制 API 存取。
7. 將參數儲存至組態檔案 — 這會使用您的部署偏好設定更新應用程式的`samconfig.toml`檔案。
8. 選取預設組態檔案名稱。
9. 選取預設組態環境。

以下是`sam deploy --guided`互動式流程的範例輸出：

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

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

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

    Setting default arguments for 'sam deploy'
    =========================================
    Stack Name [sam-app]: ENTER
    AWS Region [us-west-2]: ENTER
    #Shows you resources changes to be deployed and require a 'Y' to initiate
 deploy
    Confirm changes before deploy [Y/n]: n
    #SAM needs permission to be able to create roles to connect to the resources in
 your template
    Allow SAM CLI IAM role creation [Y/n]: ENTER
```

```
    #Preserves the state of previously provisioned resources when an operation
 fails
    Disable rollback [y/N]: ENTER
    HelloWorldFunction may not have authorization defined, Is this okay? [y/N]: y
    Save arguments to configuration file [Y/n]: ENTER
    SAM configuration file [samconfig.toml]: ENTER
    SAM configuration environment [default]: ENTER
```

3. 透過執行下列動作來 AWS SAMCLI部署您的應用程式：

   - AWS SAMCLI創建一個 Amazon S3 存儲桶並上傳您的.aws-sam目錄。
   - 將您的 AWS SAM 模板 AWS SAMCLI轉換為 AWS CloudFormation 並將其上傳到 AWS CloudFormation 服務。
   - AWS CloudFormation 提供您的資源。

   在部署期間，會 AWS SAMCLI顯示您的進度。下面是一個示例輸出：

```
Looking for resources needed for deployment:

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

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

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

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

    Deploying with following values
    ===============================
    Stack name                   : sam-app
    Region                       : us-west-2
```

```
    Confirm changeset            : False
    Disable rollback             : False
    Deployment s3 bucket         : aws-sam-cli-managed-default-
samclisourcebucket-1a4x26zbcdkqr
    Capabilities                 : ["CAPABILITY_IAM"]
    Parameter overrides          : {}
    Signing Profiles             : {}

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

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


Waiting for changeset to be created..

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


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


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

CloudFormation events from stack operations (refresh every 0.5 seconds)
---------------------------------------------------------------------------------------------
ResourceStatus                      ResourceType
 LogicalResourceId                   ResourceStatusReason
---------------------------------------------------------------------------------------------
```

```
UPDATE_IN_PROGRESS                  AWS::Lambda::Function
 HelloWorldFunction                  -
UPDATE_COMPLETE                     AWS::Lambda::Function
 HelloWorldFunction                  -
UPDATE_COMPLETE_CLEANUP_IN_PROGRE   AWS::CloudFormation::Stack          sam-app
                          -
SS
DELETE_IN_PROGRESS                  AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt   -
                                                                        ack
DELETE_COMPLETE                     AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt   -
                                                                        ack
UPDATE_COMPLETE                     AWS::CloudFormation::Stack          sam-app
                          -
-----------------------------------------------------------------------------------------

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

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

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


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

您的應用程式現在已部署並在中執行 AWS 雲端!

步驟 4：執行您的應用程式

在此步驟中，您將向 API 端點傳送 GET 請求，並查看您的 Lambda 函數輸出。

取得您的 API 端點值

1. 從上一個步驟 AWS SAMCLI中顯示的資訊中，找出該Outputs區段。在本節中，找到您的HelloWorldApi資源以查找 HTTP 端點值。下面是一個示例輸出：

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

2. 或者，您可以使用sam list endpoints --output json命令來獲取此信息。下面是一個示例輸出：

```
$ sam list endpoints --output json
2023-03-15 12:39:19 Loading policies from IAM...
2023-03-15 12:39:25 Finished loading policies from IAM.
[
  {
    "LogicalResourceId": "HelloWorldFunction",
    "PhysicalResourceId": "sam-app-HelloWorldFunction-yQDNe17r9maD",
    "CloudEndpoint": "-",
    "Methods": "-"
  },
  {
    "LogicalResourceId": "ServerlessRestApi",
    "PhysicalResourceId": "ets1gv8lxi",
    "CloudEndpoint": [
      "https://ets1gv8lxi.execute-api.us-west-2.amazonaws.com/Prod",
      "https://ets1gv8lxi.execute-api.us-west-2.amazonaws.com/Stage"
    ],
    "Methods": [
      "/hello['get']"
    ]
```

```
  }
]
```

調用你的函數

- 使用瀏覽器或命令列，將 GET 要求傳送至您的 API 端點。以下是使用 curl 命令的示例：

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

第 5 步：與您的功能進行交互 AWS 雲端

在此步驟中，您可以使 AWS SAMCLI用在中叫用您的 Lambda 函數 AWS 雲端。

若要在雲端中叫用 Lambda 函數

1. 記下上一步LogicalResourceId中的功能。它應該是HelloWorldFunction。
2. 在命令行中，從sam-app項目目錄中運行以下命令：

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

3. 會在雲端 AWS SAMCLI叫用您的函式並傳回回應。下面是一個示例輸出：

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

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

步驟 6：修改您的應用程序並將其同步到 AWS 雲端

在此步驟中，您可以使用 AWS SAMCLIsam sync --watch指令將本機變更同步至 AWS 雲端.

若要使用 sam 同步

1. 在命令行中，從sam-app項目目錄中運行以下命令：

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

2. 系 AWS SAMCLI統會提示您確認是否正在同步開發堆疊。由於該sam sync --watch命令會即時自動將本機變更部署到，因此我們建議您僅 AWS 雲端 在開發環境中使用。

   會在開始監視本機變更之前 AWS SAMCLI執行初始部署。下面是一個示例輸出：

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

Confirm that you are synchronizing a development stack.

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

Build Succeeded

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


    Deploying with following values
    ===============================
    Stack name                   : sam-app
    Region                       : us-west-2
    Disable rollback             : False
```

```
    Deployment s3 bucket         : aws-sam-cli-managed-default-
samclisourcebucket-1a4x26zbcdkqr
    Capabilities                 : ["CAPABILITY_NAMED_IAM",
 "CAPABILITY_AUTO_EXPAND"]
    Parameter overrides          : {}
    Signing Profiles             : null

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


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

CloudFormation events from stack operations (refresh every 0.5 seconds)
---------------------------------------------------------------------------------------------
ResourceStatus                       ResourceType
 LogicalResourceId                    ResourceStatusReason
---------------------------------------------------------------------------------------------
UPDATE_IN_PROGRESS                   AWS::CloudFormation::Stack          sam-app
                          Transformation succeeded
CREATE_IN_PROGRESS                   AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt    -
                                                                          ack
CREATE_IN_PROGRESS                   AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt    Resource creation Initiated
                                                                          ack
CREATE_COMPLETE                      AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt    -
                                                                          ack
UPDATE_IN_PROGRESS                   AWS::Lambda::Function
 HelloWorldFunction                   -
UPDATE_COMPLETE                      AWS::Lambda::Function
 HelloWorldFunction                   -
UPDATE_COMPLETE_CLEANUP_IN_PROGRE    AWS::CloudFormation::Stack          sam-app
                            -
SS
UPDATE_COMPLETE                      AWS::CloudFormation::Stack          sam-app
                            -
---------------------------------------------------------------------------------------------

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

```
Key                 HelloWorldFunctionIamRole
Description         Implicit IAM Role created for Hello World function
Value               arn:aws:iam::012345678910:role/sam-app-
HelloWorldFunctionRole-15GLOUR9LMT1W

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

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


Stack update succeeded. Sync infra completed.

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

接下來，您將修改 Lambda 函數程式碼。 AWS SAMCLI會自動偵測此變更，並將您的應用程式同步至 AWS 雲端.

若要修改和同步您的應用程式

1. 在您選擇的 IDE 中，打開該sam-app/hello_world/app.py文件。
2. 變更message並儲存檔案。以下是範例：

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

```
    }
```

3. 會 AWS SAMCLI偵測您的變更，並將您的應用程式同步到 AWS 雲端. 下面是一個示例輸出：

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

4. 若要驗證您的變更，請再次傳送 GET 要求至您的 API 端點。

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

步驟 7：（可選）在本地測試您的應用程序

> **Note**
>
> 此步驟是可選的，因為它需要Docker在本地計算機上。

> **Important**
>
> 若要使用進 AWS SAMCLI行本機測試，您必須已Docker安裝並設定。如需詳細資訊，請參閱安裝 Docker。

在此步驟中，您可以使用 AWS SAMCLIsam local命令在本機測試應用程式。若要完成此操作，會使用 AWS SAMCLI建立本端環境Docker。這個本機環境會模擬 Lambda 函數的雲端式執行環境。

您可執行下列項目：

1. 為您的 Lambda 函數建立本機環境並呼叫它。
2. 在本機託管您的 HTTP API 端點，並使用它來叫用您的 Lambda 函數。

若要在本機叫用 Lambda 函數

1. 在命令行中，從sam-app項目目錄中運行以下命令：

```
$ sam local invoke
```

2. 會 AWS SAMCLI建立本機Docker容器並叫用您的函式。下面是一個示例輸出：

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

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

若要在本機託管您的 API

1. 在命令行中，從sam-app項目目錄中運行以下命令：

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

2. 會為您的 Lambda 函數 AWS SAMCLI建立本機Docker容器，並建立本機 HTTP 伺服器來模擬您的 API 端點。下面是一個示例輸出：

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

Mounting /Users/.../Demo/sam-tutorial1/sam-app/.aws-sam/build/HelloWorldFunction
 as /var/task:ro,delegated inside runtime container
Containers Initialization is done.
```

```
Mounting HelloWorldFunction at http://127.0.0.1:3000/hello [GET]
You can now browse to the above endpoints to invoke your functions. You do not
 need to restart/reload SAM CLI while working on your functions, changes will be
 reflected instantly/automatically. If you used sam build before running local
 commands, you will need to re-run sam build for the changes to be picked up. You
 only need to restart SAM CLI if you update your AWS SAM template
2023-03-15 14:25:21 WARNING: This is a development server. Do not use it in a
 production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:3000
2023-03-15 14:25:21 Press CTRL+C to quit
```

3. 使用瀏覽器或命令列，將 GET 要求傳送至本機 API 端點。以下是使用 curl 命令的示例：

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

步驟 8：從中刪除您的應用程序 AWS 雲端

在此步驟中，您可以使用 AWS SAMCLIsam delete指令從中刪除您的應用程式 AWS 雲端。

若要刪除您的應用程式 AWS 雲端

1. 在命令行中，從sam-app項目目錄中運行以下命令：

```
$ sam delete
```

2. AWS SAMCLI將要求您確認。然後，它將刪除您應用程序的 Amazon S3 存儲桶和 AWS CloudFormation 堆棧。下面是一個示例輸出：

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

```
Deleted successfully
```

故障診斷

若要疑難排解 AWS SAMCLI，請參閱AWS SAMCLI疑難排。

進一步了解

若要繼續學習 AWS SAM，請參閱下列資源：

- 完整的 AWS SAM 工作坊 — 旨在教您 AWS SAM 提供許多主要功能的工作坊。
- 與 SAM 的會話 — 由我們的 AWS 無伺服器開發人員倡導團隊建立的影片系列，內容包括 AWS SAM
- 無伺服器 Land — 整合無 AWS 伺服器的最新資訊、部落格、影片、程式碼和學習資源的網站。

如何使用 AWS Serverless Application Model （ AWS SAM ）

您用來開發應用程式的主要工具是AWS SAMCLI和AWS SAM 範本和 AWS SAM 專案 (這是您的應用程式專案目錄)。您可以使用這些工具來：

1. 開發您的應用(這包括初始化您的應用程式、定義資源以及建置應用程式)。
2. 測試您的應用.
3. 偵錯應用程式.
4. 部署您的應用程式和資源.
5. 監控您的應用.

AWS SAM 在您執行sam init指令並完成其後續工作流程後建立 AWS SAM 專案。您可以透過將程式碼新增至 AWS SAM 專案來定義無伺服器應用程式。雖然您的 AWS SAM 專案包含一組檔案和資料夾，但其中最重要的檔案是您的 AWS SAM 範本 (名為 template.yaml)。在此範本中，您可以撰寫程式碼來表示資源、事件來源對應，以及定義無伺服器應用程式的其他屬性。

包 AWS SAMCLI含您在專案中使用的指令的儲存 AWS SAM 庫。更具體地說， AWS SAMCLI是您用來建置、轉換、部署、偵錯、封裝、初始化和同步處理 AWS SAM 專案的功能。換句話說，這是您用來將 AWS SAM 專案轉換為無伺服器應用程式的方式。

主題

- 該 AWS SAMCLI
- 項 AWS SAM 目和 AWS SAM 模板

該 AWS SAMCLI

命 AWS Serverless Application Model 令列介面 (AWS SAMCLI) 是您用來在應用 AWS SAM 程式專案目錄上執行命令，並最終將其轉換為無伺服器應用程式的工具。更具體地說， AWS SAMCLI可讓您建置、轉換、部署、偵錯、封裝、初始化和同步處理您的 AWS SAM 應用程式專案目錄。

AWS SAMCLI和 AWS SAM 範本隨附支援的第三方整合，可用來建置和執行您的無伺服器應用程式。

主題

- 如何記錄 AWS SAMCLI命令

- 配置 AWS SAMCLI
- AWS SAMCLI核心命令

如何記錄 AWS SAMCLI命令

AWS SAMCLI使用下列格式記錄指令：

- 提Linux示 — 依預設會記錄提示，並顯示為 ($)。對於Windows特定的指令，(>) 用作提示。鍵入命令時，請不要包含該提示。
- 目錄 – 命令必須從特定的目錄執行時，該目錄名稱會顯示在提示符號的前方。
- 使用者輸入 – 於命令列輸入的命令文字採用 **`user input`** 格式。
- 可取代文字 — 可變文字 (例如檔案名稱和參數) 會格式化為*#####*。在需要特定鍵盤輸入的多行指令或指令中，鍵盤輸入也可以顯示為可取代文字。例如，*##*。
- 輸出 — 作為指令回應傳回的輸出格式為`computer output`。

下面的`sam deploy`命令和輸出是一個例子：

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

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

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

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

```
    SAM configuration environment [default]: ENTER
```

1. `sam deploy --guided --template template.yaml`是您在指令行中輸入的指令。
2. **`sam deploy --guided --template`**應按原樣提供。
3. *`## .yaml`* 可以用您的特定檔案名稱取代。
4. 輸出開始於`Configuring SAM deploy`。
5. 在輸出中，*`ENTER`* 和 *`y`* 表示您提供的可取代值。

配置 AWS SAMCLI

其中一個好處 AWS SAM 是它通過刪除重複的任務來優化開發人員的時間。 AWS SAMCLI包含為此目`samconfig`的命名的組態檔案。依預設，不需要組態，但 AWS SAMCLI是您可以更新組態檔案，讓 AWS SAM 您以較少的參數執行指令，方法是允許在組態檔案中參照您的自訂參數。下表中的範例顯示如何最佳化指令：

原始的	最佳化 **`samconfig`**
sam build --cached --parallel --use-containers	sam build
sam local invoke --env-vars locals.json	sam local invoke
sam local start-api --env-vars locals.json --warm-containers EAGER	sam local start-api

提 AWS SAMCLI供一組命令來協助開發人員建立、開發和部署無伺服器應用程式。這些命令中的每一個都可以根據應用程序和開發人員的首選項配置可選標誌。如需詳細資訊，請參閱中的AWS SAMCLI內容 GitHub

本節中的主題說明如何建立AWS SAMCLI配置文件及自訂其預設設定，以最佳化無伺服器應用程式的開發時間。

主題

- 如何建立您的組態檔案 (samconfig檔案)
- 設定專案設定
- 設定認證和基本設定

如何建立您的組態檔案 (samconfig檔案)

AWS SAMCLI設定檔 (檔案名稱samconfig) 是一個文字檔，通常使用 TOML 結構，但也可以在 YAML 中。使用 AWS 快速入門樣板時，會在您執行sam init指令時建立此檔案。您可以在使用sam deploy -\-guided指令部署應用程式時更新此檔案。

部署完成後，samconfig檔案會包含一個名為的紀要 (default如果您使用預設值)。當您重新執行deploy命令時，會 AWS SAM 套用此設定檔中的預存組態設定。

該samconfig文件的好處是 AWS SAM 存儲除了 deploy 命令之外可用的任何其他命令的配置設置。除了在新部署中建立的這些值之外，您還可以在samconfig檔案中設定許多屬性，這些屬性可以簡化開發人員工作流程的其他層面 AWS SAMCLI。

設定專案設定

您可以在規劃檔中指定專案特有的設定 (例如指 AWS SAMCLI令參數值)，以配合使用。 AWS SAMCLI如需有關此組態檔案的更多資訊，請參閱AWS SAMCLI配置文件。

使用組態檔

組態檔案由環境、指令和參數值建構。如需詳細資訊，請參閱 組態檔案基本概。

若要設定新環境

1. 在組態檔案中指定新環境。

   以下是指定新prod環境的範例：

   TOML

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

   YAML

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

2. 在組態檔案的參數區段中，將參數值指定為鍵值配對。

   以下是為prod環境指定應用程式堆疊名稱的範例。

TOML

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

YAML

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

3. 使用--config-env此選項指定要使用的環境。

   以下是範例：

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

若要設定參數值

1. AWS SAMCLI指定您要為其配置參數值的命令。若要設定所有 AWS SAMCLI指令的參數值，請使用global識別碼。

   以下是指定default環境指sam deploy令參數值的範例：

   TOML

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

   YAML

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

   以下是為default環境中所有指 AWS SAMCLI令指定參數值的範例：

TOML

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

YAML

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

2. 您也可以透過 AWS SAMCLI互動式流程指定參數值並修改組態檔案。

   以下是sam deploy --guided互動式流程的範例：

```
$ sam deploy --guided

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

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

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

如需詳細資訊，請參閱 建立和修改組態檔。

範例

基本TOML範例

以下是組samconfig.toml態檔案的範例：

```
...
version = 0.1

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

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

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

[default.sync.parameters]
watch = true

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

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

基本YAML範例

以下是組samconfig.yaml態檔案的範例：

```
version 0.1
default:
  global:
```

```
    parameters:
      stack_name: sam-app
  build:
    parameters:
      cached: true
      parallel: true
  deploy:
    parameters:
      capabilities: CAPABILITY_IAM
      confirm_changeset: true
      resolve_s3: true
  sync:
    parameters:
      watch: true
  local_start_api:
    parameters:
      warm_containers: EAGER
prod:
  sync:
    parameters:
      watch: false
```

設定認證和基本設定

使用 AWS Command Line Interface (AWS CLI) 可配置基本設定，例如認 AWS 證、預設區域名稱和預設輸出格式。設定完成後，您可以將這些設定與 AWS SAMCLI. 若要進一步了解，請參閱使AWS Command Line Interface 用者指南中的以下內容：

- 配置基本知
- 組態和認證檔案設定
- 已命名的設定檔 AWS CLI
- 使用已啟用 IAM 身分中心的具名設定檔

如需快速設定指示，請參閱步驟 5：使用設 AWS CLI 定 AWS 認證。

AWS SAMCLI核心命令

AWS SAMCLI您可以使用一些基本命令來建立、建置、測試、部署和同步處理無伺服器應用程式。下表列出了這些命令，並提供了每個命令的更多信息的鏈接。

如需指 AWS SAMCLI令的完整清單，請參閱AWS SAMCLI指令參考。

Command	它做了什麼	相關主題
sam build	為開發人員工作流程中的後續步驟準備應用程式，例如本機測試或部署到 AWS 雲端。	• 使用sam build指令建立簡介 • sam build
sam deploy	使用將應用程式部署到 AWS 雲端 AWS CloudFormation。	• 使用sam deploy指令部署簡介 • sam deploy
sam init	提供初始化和建立新無伺服器應用程式的選項。	• 使用sam init指令建立您的應用程式 • sam init
sam local	提供子命令以在本機測試無伺服器應用程式。	• 使用sam local指令進行測試簡介 • sam local generate-event • sam local invoke • sam local start-api • sam local start-lambda
sam remote invoke	提供存取和管理 AWS Lambda 函數可共用測試事件的方法。	• 雲端測試簡介 sam remote invoke • sam remote invoke
sam remote test-event	提供與雲端中支援的 AWS 資源互動的方 AWS 式。	• 雲端測試簡介 sam remote test-event • sam remote test-event
sam sync	提供將本機應用程式變更快速同步至 AWS 雲端的選項。	• 使用同步sam sync到簡介 AWS 雲端 • sam sync

項 AWS SAM 目和 AWS SAM 模板

執行sam init命令並完成其後續工作流程之後， AWS SAM 會建立您的應用程式專案目錄，也就是您的 AWS SAM 專案。您可以透過將程式碼新增至 AWS SAM 專案來定義無伺服器應用程式。雖然 AWS SAM 專案由一組檔案和資料夾組成，但您主要使用的檔案是您的 AWS SAM 樣板 (具名`template.yaml`)。在此範本中，您可以撰寫程式碼來表示資源、事件來源對應，以及定義無伺服器應用程式的其他屬性。

> **Note**
>
> AWS SAM 模板的一個關鍵要素是 AWS SAM 模板規範。此規格提供簡短語法，與之相比 AWS CloudFormation，可讓您使用較少的程式碼行來定義無伺服器應用程式的資源、事件來源對應、權限、API 及其他屬性。

本節提供如何使用 AWS SAM 範本中的區段來定義資源類型、資源屬性、資料類型、資源屬性、內建函數和 API Gateway 延伸模組的詳細資訊。

AWS SAM 模板是模 AWS CloudFormation 板的擴展，具有唯一的語法類型，使用速記語法與更少的代碼行比. AWS CloudFormation這可加快您在建置無伺服器應用程式時的開發速度。如需詳細資訊，請參閱 AWS SAM 資源和屬性。如需 AWS CloudFormation 範本的完整參考資料，請參閱《AWS CloudFormation 使用指南》中的〈AWS CloudFormation 範本參考〉。

主題

- AWS SAM 範本解剖學
- AWS SAM 資源和屬性
- 產生的 AWS CloudFormation 資源
- 支援的資源屬性 AWS SAM
- API Gateway 擴充功能
- 內部函數

AWS SAM 範本解剖學

AWS SAM 範本檔案緊跟 AWS CloudFormation 範本檔案的格式，如《使用指南》中的〈範本剖析 AWS CloudFormation〉中所述。範本檔案和 AWS SAM 範本檔案之 AWS CloudFormation 間的主要差異如下：

- 轉換聲明。 AWS SAM 範本檔案需要宣告`Transform: AWS::Serverless-2016-10-31`。此宣告會將 AWS CloudFormation 範本檔案識別為 AWS SAM 範本檔案。如需有關轉換的詳細資訊，請參閱AWS CloudFormation 使用指南中的「轉換」。
- 「全局變量」部分。此`Globals`區段是唯一的 AWS SAM。它定義了所有無伺服器函數和 API 通用的屬性。所有`AWS::Serverless::Function`、`AWS::Serverless::Api`和`AWS::Serverless::SimpleTable`資源都會繼承`Globals`區段中定義的屬性。如需有關本節的更多資訊，請參閱模板的全局部分 AWS SAM。
- 資源部分。在 AWS SAM 模板中，該`Resources`部分可以包含 AWS CloudFormation 資源和 AWS SAM 資源的組合。如需有關資 AWS CloudFormation 源的詳細資訊，請參閱《AWS CloudFormation 使用指南》中的AWS 資源和屬性類型參考資料。如需有關資源的詳細 AWS SAM 資訊，請參閱AWS SAM 資源和屬性。

AWS SAM 範本檔案的所有其他區段都對應於同名的 AWS CloudFormation 範本檔案區段。

YAML

下列範例顯示 YAML 格式的範本片段。

```
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
```

範本區段

AWS SAM 樣板可以包括幾個主要部分。只有Transform和Resources區段是必需的。

您可以按任何順序包括模板部分。但是，如果使用語言擴充功能，您應該在無伺服器轉換AWS::LanguageExtensions之前 (也就是之前AWS::Serverless-2016-10-31) 新增，如下列範例所示：

```
Transform:
  - AWS::LanguageExtensions
  - AWS::Serverless-2016-10-31
```

當您建立範本時，使用下列清單中顯示的邏輯順序會很有幫助。這是因為某個區段中的值可能會參考上一節中的值。

轉換 (必要)

對於 AWS SAM 範本，您必須包含此區段的值為AWS::Serverless-2016-10-31。

其他轉換是可選的。如需有關轉換的詳細資訊，請參閱AWS CloudFormation 使用指南中的「轉換」。

全局變量（可選）

所有無伺服器函數、API 和簡單表格通用的屬性。所有AWS::Serverless::FunctionAWS::Serverless::Api、和AWS::Serverless::SimpleTable資源都會繼承Globals區段中定義的屬性。

此區段是唯一的 AWS SAM。 AWS CloudFormation 範本中沒有對應的區段。

描述 (選用)

說明範本的文字字串。

本節與 AWS CloudFormation 模板Description部分直接對應。

Metadata (選用)

提供範本其他資訊的物件。

本節與 AWS CloudFormation 模板`Metadata`部分直接對應。

Parameters (選用)

要在執行時間傳遞至您範本的值 (當您建立或更新堆疊時)。您可以參照範本之 `Resources` 和 `Outputs` 區段中的參數。`Parameters`區段中宣告的物件會導致指`sam deploy --guided`令向使用者顯示其他提示。

使用`sam deploy`指令`--parameter-overrides`參數傳入的值 (以及組態檔案中的項目) 會先於範本檔案中的項目。 AWS SAM 若要取得有關`sam deploy`指令的更多資訊，請參閱 AWS SAMCLI指令參考sam deploy中的〈〉。如需組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

Mappings (選用)

可用來指定條件式參數值之索引鍵與相關聯值的映射，與查詢表格類似。您可以使用和區段中的`Fn::FindInMap`內建函式，將索引鍵`Resources`與`Outputs`對應的值相符。

本節與 AWS CloudFormation 模板`Mappings`部分直接對應。

Conditions (選用)

條件，控制是否建立特定資源，或是否在建立或更新堆疊期間指派特定資源屬性的值。例如，您可以有條件地建立資源，取決於堆疊適用於生產還是測試環境。

本節與 AWS CloudFormation 模板`Conditions`部分直接對應。

Resources (必要)

堆疊資源及其屬性，例如亞馬遜彈性運算雲端 (Amazon EC2) 執行個體或亞馬遜簡單儲存服務 (Amazon S3) 儲存貯體。您可以參照範本之 `Resources` 和 `Outputs` 區段中的資源。

本節類似於模 AWS CloudFormation 板`Resources`部分。在 AWS SAM 範本中，除了 AWS SAM 資源之外，此區段還可以包含 AWS CloudFormation 資源。

Outputs (選用)

每當您查看堆棧的屬性時返回的值。例如，您可以宣告 S3 儲存貯體名稱的輸出，然後呼叫 `aws cloudformation describe-stacks` AWS Command Line Interface (AWS CLI) 命令來檢視名稱。

本節與 AWS CloudFormation 模板`Outputs`部分直接對應。

後續步驟

若要下載並部署包含範本檔案的無伺服器應用程式 AWS SAM 範例，請參閱開始使用 AWS SAM並遵循中教學課程：部署 Hello World 應用程式的指示。

模板的全局部分 AWS SAM

有時，您在AWS SAM模板中聲明的資源具有通用配置。例如，您的應用程式可能包含多個具有相同`Runtime`、、`Memory`、`VPCConfig`、`Environment`和`Cors`組態的`AWS::Serverless::Function`資源。您可以在`Globals`部分中聲明一次，並讓您的資源繼承它們，而不是在每個資源中複製這些信息。

此`Globals`區段支援下列AWS SAM資源類型：

- `AWS::Serverless::Api`
- `AWS::Serverless::Function`
- `AWS::Serverless::HttpApi`
- `AWS::Serverless::SimpleTable`
- `AWS::Serverless::StateMachine`

範例：

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

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

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

在這個例子中，HelloWorldFunction並ThumbnailFunction使用「nodejs12.x」表示Runtime，「180」秒代表Timeout，並使用「索引。Handler HelloWorldFunction除了繼承的 TABLE_NAME 之外，還會新增訊息環境變數。 ThumbnailFunction繼承所有Globals屬性並添加 API 事件源。

支援的資源和屬性

AWS SAM支援下列資源和屬性。

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

  Function:
    Architectures:
    AssumeRolePolicyDocument:
```

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

  HttpApi:
    AccessLogSettings:
    Auth:
    PropagateTags:
    StageVariables:
    Tags:

  SimpleTable:
    SSESpecification:

  StateMachine:
    PropagateTags:
```

> **Note**
>
> 不支援任何未包含在先前清單中的資源和屬性。不支持它們的一些原因包括：1）它們會打開潛在的安全問題，或 2）它們使模板難以理解。

隱含的 API

AWS SAM當您在區段中宣告 API 時，會建立隱含Events的 API。您可以使Globals用覆蓋隱式 API 的所有屬性。

可重新定義屬性

資源可以覆寫您在Globals區段中宣告的屬性。例如，您可以將新變數新增至環境變數 map，或者您可以覆寫全域宣告的變數。但資源無法移除Globals區段中指定的屬性。

更一般地說，該Globals部分聲明了所有資源共享的屬性。某些資源可以為全域宣告的屬性提供新值，但無法移除它們。如果某些資源使用屬性，但其他資源不使用，那麼您不能在該Globals部分中聲明它們。

下列各節說明覆寫如何針對不同的資料類型運作。

原始數據類型被替換

原始數據類型包括字符串，數字，布爾值等。

區段中指定的值會取代Resources區Globals段中的值。

範例：

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

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

用Runtime於MyFunction設定為python3.9。

地圖被合併

地圖也被稱為字典或鍵值對的集合。

Resources區段中的地圖項目會與全域地圖項目合併。如果存在重複項目，則Resource剖面項目會取代Globals剖面項目。

範例：

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

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

的環境變數設MyFunction定為下列項目：

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

列表是可添加的

列表也被稱為數組。

Globals區段中的清單項目會附加在區段中的清單之Resources前。

範例：

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

Resources:
```

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

for `SecurityGroupIds` 設定`VpcConfig`為下列項目：`MyFunction`

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

AWS SAM 資源和屬性

本節說明特定的資源和屬性類型。 AWS SAM您可以使用速記語法來定義這些資源 AWS SAM 和屬性。 AWS SAM 還支持 AWS CloudFormation 資源和屬性類型。如需所有 AWS 資源和屬性類型 AWS CloudFormation 和 AWS SAM 支援的參考資訊，請參閱《AWS CloudFormation 使用指南》中的AWS資源和屬性類型參考資料。

主題

- AWS::Serverless::Api
- AWS::Serverless::Application
- AWS::Serverless::Connector
- AWS::Serverless::Function
- AWS::Serverless::GraphQLApi
- AWS::Serverless::HttpApi
- AWS::Serverless::LayerVersion
- AWS::Serverless::SimpleTable
- AWS::Serverless::StateMachine

AWS::Serverless::Api

建立可透過 HTTPS 端點叫用的 Amazon API Gateway 資源和方法的集合。

資AWS::Serverless::Api源不需要明確新增至 AWS 無伺服器應用程式定義範本。這種類型的資源是從模板中定義的AWS::Serverless::Function資源上定義的不引用資源的 Api 事件聯集隱式創建的。AWS::Serverless::Api

應使用AWS::Serverless::Api資源來定義和記錄 API OpenApi，這可提供更多設定基礎 Amazon API Gateway 資源的能力。

我們建議您使用 AWS CloudFormation 勾點或 IAM 政策來驗證 API Gateway 資源是否已附加授權人，以控制對其存取權限。

如需有關使用 AWS CloudFormation 勾點的詳細資訊，請參閱 AWS CloudFormation CLI 使用手冊和apigw-enforce-authorizer GitHub 存放庫中的註冊勾點。

如需使用 IAM 政策的詳細資訊，請參閱 API Gateway 開發人員指南中的「要求 API 路由具有授權」。

> **Note**
>
> 當您部署到時 AWS CloudFormation，將您的 AWS SAM 資源 AWS SAM 轉換為 AWS CloudFormation 資源。如需詳細資訊，請參閱 產生的 AWS CloudFormation 資源。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
Type: AWS::Serverless::Api
Properties:
  AccessLogSetting: AccessLogSetting
  AlwaysDeploy: Boolean
  ApiKeySourceType: String
  Auth: ApiAuth
  BinaryMediaTypes: List
  CacheClusterEnabled: Boolean
  CacheClusterSize: String
  CanarySetting: CanarySetting
  Cors: String | CorsConfiguration
  DefinitionBody: JSON
  DefinitionUri: String | ApiDefinition
  Description: String
  DisableExecuteApiEndpoint: Boolean
  Domain: DomainConfiguration
  EndpointConfiguration: EndpointConfiguration
  FailOnWarnings: Boolean
  GatewayResponses: Map
```

```
  MergeDefinitions: Boolean
  MethodSettings: MethodSettings
  MinimumCompressionSize: Integer
  Mode: String
  Models: Map
  Name: String
  OpenApiVersion: String
  PropagateTags: Boolean
  StageName: String
  Tags: Map
  TracingEnabled: Boolean
  Variables: Map
```

屬性

`AccessLogSetting`

設定階段的存取日誌設定。

類型:AccessLogSetting

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::ApiGateway::Stage`資源的`AccessLogSetting`屬性。

`AlwaysDeploy`

始終部署 API，即使沒有檢測到對 API 的更改也是如此。

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`ApiKeySourceType`

根據用量計畫進行量測請求的 API 金鑰來源，有效值為 `HEADER` 和 `AUTHORIZER`。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::ApiGateway::RestApi資源的ApiKeySourceType屬性。

Auth

設定授權以控制 API Gateway API 的存取權。

如需使用配置存取權的詳細資訊， AWS SAM 請參閱使用 AWS SAM 範本控制 API 存取。

類型:ApiAuth

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

BinaryMediaTypes

您的 API 可以返回的 MIME 類型列表。使用此選項可啟用 API 的二進位支援。在 MIME 類型中使用～ 1 而不是/。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性類似於AWS::ApiGateway::RestApi資源的BinaryMediaTypes屬性。的清單 BinaryMediaTypes 會新增至 AWS CloudFormation 資源和 OpenAPI 文件中。

CacheClusterEnabled

指出是否為階段啟用快取。若要快取回應，您也必須CachingEnabled將設定為 true [下] MethodSettings。

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::ApiGateway::Stage資源的CacheClusterEnabled屬性。

CacheClusterSize

階段的快取叢集大小。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::ApiGateway::Stage`資源的`CacheClusterSize`屬性。

`CanarySetting`

將初期測試設定設定設定為一般部署的階段。

類型:CanarySetting

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::ApiGateway::Stage`資源的`CanarySetting`屬性。

`Cors`

管理所有 API Gateway API 的跨來源資源共用 (CORS)。指定允許作為字符串的域或指定帶有其他 CRS 配置的字典。

> **Note**
>
> CORS 需要修 AWS SAM 改您的 OpenAPI 定義。在中建立內嵌 OpenAPI 定義以`DefinitionBody`開啟 CORS。

如需有關 CORS 的詳細資訊，請參閱 API Gateway 開發人員指南中的針對 API Gateway REST API 資源啟用 CO RS。

類型：字符串 | CorsConfiguration

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`DefinitionBody`

用於描述您的 API 的 OpenAPI 規範。如果沒有指定`DefinitionUri`也沒`DefinitionBody`有指定，SAM 將根據您`DefinitionBody`的模板配置為您生成一個。

若要參考定義 API 的本機OpenAPI檔案，請使用AWS::Include轉換。如需進一步了解，請參閱部署時上傳本機檔案。

類型：JSON

必要：否

AWS CloudFormation 兼容性：此屬性類似於AWS::ApiGateway::RestApi資源的Body屬性。如果提供了某些屬性，則可以在將內容插入或修改到中，然後DefinitionBody 再將內容傳遞給 CloudFormation。屬性包括AuthBinaryMediaTypesCors、GatewayResponses、Models、和類EventSource型為 Api 的對應AWS::Serverless::Function。

`DefinitionUri`

Amazon S3 Uri、本機檔案路徑或定義 OpenAPI 件的位置物件。此屬性參考的 Amazon S3 物件必須是有效的 OpenAPI 檔案。如果沒有指定DefinitionUri也沒DefinitionBody有指定，SAM 將根據您DefinitionBody的模板配置為您生成一個。

如果提供了本機檔案路徑，則範本必須經過包含sam deploy或sam package指令的工作流程，才能正確轉換定義。

所參考的外部 OpenApi 檔案不支援內建函式。DefinitionUri改為搭配「包含轉換」使用DefinitionBody屬性，將定 OpenApi 義匯入範本。

類型：字符串 | ApiDefinition

必要：否

AWS CloudFormation 兼容性：此屬性類似於AWS::ApiGateway::RestApi資源的BodyS3Location屬性。巢狀 Amazon S3 屬性的命名方式不同。

`Description`

Api 資源的描述。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::ApiGateway::RestApi資源的Description屬性。

`DisableExecuteApiEndpoint`

指定用戶端是否可以使用預設 `execute-api` 端點叫用您的 API。默認情況下，客戶端可以使用默認調用您的 API `https://{api_id}.execute-api.{region}.amazonaws.com`。如要要求用戶端使用自訂網域名稱來叫用 API，請指定 `True`。

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性類似於`AWS::ApiGateway::RestApi`資源的 `DisableExecuteApiEndpoint`屬性。它會直接傳遞至擴充功能的`disableExecuteApiEndpoint`屬性，而 `x-amazon-apigateway-endpoint-configuration`擴充功能會新增至`AWS::ApiGateway::RestApi`資源的 `Body`屬性。

`Domain`

設定此 API Gateway API 的自訂網域。

類型:DomainConfiguration

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`EndpointConfiguration`

其餘 API 的端點類型。

類型:EndpointConfiguration

必要：否

AWS CloudFormation 兼容性：此屬性類似於`AWS::ApiGateway::RestApi`資源的`EndpointConfiguration`屬性。巢狀組態屬性的命名方式不同。

`FailOnWarnings`

指定遇到警告時是否回滾 API 創建（`truefalse`）或不（ ）。預設值為 `false`。

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::ApiGateway::RestApi資源的FailOnWarnings屬性。

GatewayResponses

設定 API 的閘道回應。閘道回應是 API Gateway 直接或透過 Lambda 授權人傳回的回應。如需詳細資訊，請參閱閘道回應的 Api 閘道 OpenApi 延伸模組文件。

類型：地圖

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

MergeDefinitions

AWS SAM 從您的 API 事件源生成OpenAPI規範。指定true將其 AWS SAM 合併到資AWS::Serverless::Api源中定義的內嵌OpenAPI規格中。指false定不合併。

MergeDefinitions需要定義AWS::Serverless::Api的DefinitionBody屬性。MergeDefinitions與的DefinitionUri屬性不相容AWS::Serverless::Api。

預設值：false

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

MethodSettings

配置 API 階段的所有設置，包括日誌記錄，指標，緩存，節流。

類型： MethodSetting 的清單

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::ApiGateway::Stage資源的MethodSettings屬性。

`MinimumCompressionSize`

允許基於客戶端的接受編碼頭響應主體的壓縮。當回應主體大小大於或等於您設定的閾值時，就會觸發壓縮。最大主體大小閾值為 10 MB（10,485,760 字節）。-支援下列壓縮類型：gzip、放氣和識別。

類型：整數

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::ApiGateway::RestApi`資源的`MinimumCompressionSize`屬性。

`Mode`

只有當您使用 OpenAPI 定義 REST API 時，才會套用此屬性。`Mode` 可判定 API Gateway 如何處理資源更新。如需詳細資訊，請參閱AWS::ApiGateway::RestApi資源類型的模式屬性。

有效值：`overwrite` 或 `merge`

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::ApiGateway::RestApi`資源的`Mode`屬性。

`Models`

您的 API 方法要使用的結構描述。這些結構描述可以使用 JSON 或 YAML 來描述。有關示例模型，請參閱本頁底部的「示例」部分。

類型：地圖

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Name`

API Gateway RestApi 資源的名稱

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::ApiGateway::RestApi`資源的`Name`屬性。

`OpenApiVersion`

OpenApi 要使用的版本。這可以是 Swagger 規範，也可以是 OpenApi 3.0 版本之一，`2.0`如。`3.0.1`如需有關 OpenAPI 的詳細資訊，請參閱 OpenAPI 規格。

> **Note**
>
> AWS SAM 會建立`Stage`依預設呼叫的階段。將此屬性設定為任何有效值將會阻止建立階段`Stage`。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`PropagateTags`

指出是否要將標籤從`Tags`屬性傳遞至您`AWS::Serverless::Api`產生的資源。指`True`定在產生的資源中傳播標籤。

類型：布林值

必要：否

預設：`False`

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`StageName`

階段的名稱，API Gateway 使用做為叫用統一資源識別元 (URI) 中的第一個路徑區段。

若要參照階段資源，請使用`<api-logical-id>.Stage`。如需有關參考指定資源時產生之`AWS::Serverless::Api`資源的詳細資訊，請參閱AWS CloudFormation 指定時產生AWS::Serverless::Api的資源。如需有關已產生 AWS CloudFormation 資源的一般資訊，請參閱產生的 AWS CloudFormation 資源。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性類似於AWS::ApiGateway::Stage資源的StageName屬性。它在 SAM 中是必需的，但在 API Gateway 中不是必需的

附加說明：隱式 API 的階段名稱為「Prod」。

`Tags`

映射（字符串到字符串），指定要添加到此 API Gateway 階段的標籤。如需有關標籤的有效鍵和值的詳細資訊，請參閱《AWS CloudFormation 使用指南》中的〈Resource 標籤〉。

類型：地圖

必要：否

AWS CloudFormation 兼容性：此屬性類似於AWS::ApiGateway::Stage資源的Tags屬性。SAM 中的標籤屬性由鍵：值對組成; CloudFormation 其中包含標籤對象的列表。

`TracingEnabled`

指出舞台是否已啟用使用 X-Ray 的作用中追蹤。如需有關 X-Ray 的詳細資訊，請參閱 API Gateway 開發人員指南中的使用 X-Ray 追蹤使用者對 REST API 的要求。

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::ApiGateway::Stage資源的TracingEnabled屬性。

`Variables`

定義階段變數的對映 (字串到字串)，其中變數名稱是索引鍵，變數值是值。變數名稱限制為英數字元。值必須符合下列規則表達式：[A-Za-z0-9._~:/?#&=,-]+。

類型：地圖

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::ApiGateway::Stage資源的Variables屬性。

傳回值

Ref

將此資源的邏輯 ID 提供給`Ref`內建函數時，它會傳回基礎 API Gateway API 的識別碼。

若要取得有關使用`Ref`功能的更多資訊，請參閱《使AWS CloudFormation 用指南》*Ref*中的〈〉

Fn: GetAtt

`Fn::GetAtt` 會傳回此類型之指定屬性的值。以下為可用屬性及傳回值的範例。

若要取得有關使用的更多資訊`Fn::GetAtt`，請參閱使AWS CloudFormation 用指南*Fn::GetAtt*中的〈〉

`RootResourceId`

`RestApi` 資源的根資源 ID，例如：`a0bc123d4e`。

範例

SimpleApiExample

包含具有 API 端點的 Lambda 函數的「你好世界」 AWS SAM 範本檔案。這是一個工作的無服務器應用程序的完整 AWS SAM 模板文件。

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: AWS SAM template with a simple API definition
Resources:
  ApiGatewayApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: prod
  ApiFunction: # Adds a GET method at the root resource via an Api event
    Type: AWS::Serverless::Function
    Properties:
      Events:
        ApiEvent:
          Type: Api
          Properties:
```

```
            Path: /
            Method: get
            RestApiId:
              Ref: ApiGatewayApi
      Runtime: python3.10
      Handler: index.handler
      InlineCode: |
        def handler(event, context):
            return {'body': 'Hello World!', 'statusCode': 200}
```

ApiCorsExample

在外部 Swagger 檔案中定義具有 API 的 AWS SAM 範本程式碼片段，以及 Lambda 整合和 CORS 設定。這只是顯示AWS::Serverless::Api定義的 AWS SAM 模板文件的一部分。

YAML

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

ApiCognitoAuthExample

具有 API 的 AWS SAM 範本程式碼片段，該程式碼片段使用 Amazon Cognito 針對 API 授權請求。這只是顯示AWS::Serverless::Api定義的 AWS SAM 模板文件的一部分。

YAML

```
Resources:
  ApiGatewayApi:
    Type: AWS::Serverless::Api
```

```
    Properties:
      StageName: Prod
      Cors: "'*'"
      Auth:
        DefaultAuthorizer: MyCognitoAuthorizer
        Authorizers:
          MyCognitoAuthorizer:
            UserPoolArn:
              Fn::GetAtt: [MyCognitoUserPool, Arn]
```

ApiModelsExample

包含 AWS SAM 模型結構描述的 API 的範本程式碼片段。這只是 AWS SAM 範本檔案的一部分，顯示具有兩個模型結構描述的AWS::Serverless::Api定義。

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
```

緩存示例

包含具有 API 端點的 Lambda 函數的「你好世界」 AWS SAM 範本檔案。該 API 已啟用一個資源和方法的緩存。如需快取的詳細資訊，請參閱 API Gateway 開發人員指南中的啟用 API 快取以增強回應能力。

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: AWS SAM template with a simple API definition with caching turned on
Resources:
  ApiGatewayApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: prod
      CacheClusterEnabled: true
      CacheClusterSize: '0.5'
      MethodSettings:
        - ResourcePath: /
          HttpMethod: GET
          CachingEnabled: true
          CacheTtlInSeconds: 300
      Tags:
        CacheMethods: All

  ApiFunction: # Adds a GET method at the root resource via an Api event
    Type: AWS::Serverless::Function
    Properties:
      Events:
        ApiEvent:
          Type: Api
          Properties:
            Path: /
            Method: get
            RestApiId:
              Ref: ApiGatewayApi
      Runtime: python3.10
      Handler: index.handler
      InlineCode: |
        def handler(event, context):
```

```
            return {'body': 'Hello World!', 'statusCode': 200}
```

ApiAuth

設定授權以控制 API Gateway API 的存取權。

如需使用來配置存取權的詳細資訊和範例， AWS SAM 請參閱使用 AWS SAM 範本控制 API 存取。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
AddApiKeyRequiredToCorsPreflight: Boolean
AddDefaultAuthorizerToCorsPreflight: Boolean
ApiKeyRequired: Boolean
Authorizers: CognitoAuthorizer | LambdaTokenAuthorizer | LambdaRequestAuthorizer
DefaultAuthorizer: String
InvokeRole: String
ResourcePolicy: ResourcePolicyStatement
UsagePlan: ApiUsagePlan
```

屬性

`AddApiKeyRequiredToCorsPreflight`

如果設定了`ApiKeyRequired`和`Cors`屬性，則設定`AddApiKeyRequiredToCorsPreflight`會導致 API 金鑰新增至`Options`屬性。

類型：布林值

必要：否

預設：`True`

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`AddDefaultAuthorizerToCorsPreflight`

如果設定`DefaultAuthorizer`和`Cors`屬性，則設定`AddDefaultAuthorizerToCorsPreflight`將導致預設授權者新增至 OpenAPI 區段中的`Options`屬性。

類型：布林值

必要：否

預設值：真

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`ApiKeyRequired`

如果設置為 true，則所有 API 事件都需要 API 密鑰。如需 API 金鑰的詳細資訊，請參閱 API Gateway 開發人員指南中的建立和使用搭配 API 金鑰的使用方案。

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Authorizers`

用來控制 API Gateway API 存取權的授權者。

如需詳細資訊，請參閱 使用 AWS SAM 範本控制 API 存取。

類型:CognitoAuthorizerLambdaTokenAuthorizer| LambdaRequestAuthorizer

必要：否

預設：無

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

附加說明：SAM 將授權者添加到 Api 的 OpenApi 定義中。

`DefaultAuthorizer`

指定 API Gateway API 的預設授權程式，預設情況下將用於授權 API 呼叫。

> **Note**
>
> 如果與此 API EventSource 關聯之函數的 Api 設定為使用 IAM 許可，則必須將此屬性設定為`AWS_IAM`，否則將導致錯誤。

類型：字串

必要：否

預設：無

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`InvokeRole`

將所有資源和方法的整合認證設定為此值。

`CALLER_CREDENTIALS`映射到`arn:aws:iam::*:user/*`，它使用呼叫者認證來調用端點。

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

類型：字串

必要：否

預設：`CALLER_CREDENTIALS`

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`ResourcePolicy`

為 API 上的所有方法和路徑配置資源策略。

類型:ResourcePolicyStatement

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

附加注意事項：此設定也可以個別`AWS::Serverless::Function`使用ApiFunctionAuth. 這對於使用`EndpointConfiguration: PRIVATE`.

`UsagePlan`

設定與此 API 相關聯的使用方案。如需使用方案的詳細資訊，請參閱 API Gateway 開發人員指南中的建立和使用搭配 API 金鑰的使用方案。

設定此 AWS SAM 屬性時 AWS::ApiGateway::UsagePlan，此屬性會產生三個額外的 AWS CloudFormation 資源：a AWS::ApiGateway::UsagePlanKey、a 和 AWS::ApiGateway::ApiKey. 如需有關此案例的資訊，請參閱UsagePlan屬性已指定。如需有關已產生 AWS CloudFormation 資源的一般資訊，請參閱產生的 AWS CloudFormation 資源。

類型:ApiUsagePlan

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

CognitoAuth

Cognito 身份驗證示例

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

設定 API Gateway API 的使用計劃。如需使用方案的詳細資訊，請參閱 API Gateway 開發人員指南中的搭配 API 金鑰建立和使用使用方案。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

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

屬性

`CreateUsagePlan`

決定此使用量計劃的配置方式。有效值為 PER_API、SHARED 和 NONE。

PER_API建立AWS::ApiGateway::UsagePlan此 API 專屬的AWS::ApiGateway::ApiKey、和AWS::ApiGateway::UsagePlanKey資源。這些資源分別具有*<api-logical-id>*UsagePlan*<api-logical-id>*ApiKey*<api-logical-id>*UsagePlanKey、和的邏輯 ID。

SHARED建立AWS::ApiGateway::UsagePlanAWS::ApiGateway::ApiKey、和AWS::ApiGateway::UsagePlanKey資源，這些 API 在同一 AWS SAM 範本中也包含CreateUsagePlan: SHARED的任何 API 共用。這些資源分別具有ServerlessUsagePlanServerlessApiKeyServerlessUsagePlanKey、和的邏輯 ID。如果您使用此選項，建議您僅在一個 API 資源上新增此使用方案的其他設定，以避免發生衝突的定義和不確定的狀態。

NONE停用使用計劃與此 API 的建立或關聯。只有在中指定SHARED或PER_API時，才需要此操作模板的全局部分 AWS SAM。

有效值：PER_API、SHARED 與 NONE

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Description`

用量計劃的描述。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::ApiGateway::UsagePlan`資源的`Description`屬性。

`Quota`

設定使用者可在指定間隔內發出的請求數。

類型:QuotaSettings

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::ApiGateway::UsagePlan`資源的`Quota`屬性。

`Tags`

要與用量計劃關聯的任意標籤陣列 (金鑰值對)。

此性質使用「標CloudFormation 籤類型」。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::ApiGateway::UsagePlan`資源的`Tags`屬性。

`Throttle`

設定整體請求速率 (每秒平均請求數) 和高載容量。

類型:ThrottleSettings

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::ApiGateway::UsagePlan資源的Throttle屬性。

UsagePlanName

用量計劃的名稱。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::ApiGateway::UsagePlan資源的UsagePlanName屬性。

範例

UsagePlan

以下是使用計劃範例。

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

定義 Amazon Cognito 用者集區授權者。

如需詳細資訊和範例，請參閱 使用 AWS SAM 範本控制 API 存取。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
AuthorizationScopes: List
Identity: CognitoAuthorizationIdentity
UserPoolArn: String
```

屬性

`AuthorizationScopes`

此授權者的授權範圍清單。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Identity`

此屬性可用於在授權者`IdentitySource`的傳入請求中指定。

類型:CognitoAuthorizationIdentity

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`UserPoolArn`

可以參考使用者集區/指定要新增這個認知授權者的使用者集區 arn

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

CognitoAuth

Cognito 身份驗證示例

YAML

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

CognitoAuthorizationIdentity

此屬性可用於在授權者 IdentitySource 的傳入請求中指定。如需有關的詳細資訊， IdentitySource 請參閱ApiGateway 授權者 OpenApi 擴充功能。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

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

屬性

`Header`

在定 OpenApi 義中指定「授權」的標頭名稱。

類型：字串

必要：否

預設值：授權

AWS CloudFormation兼容性：此屬性是唯一的，AWS SAM並且沒有相AWS CloudFormation等的屬性。

`ReauthorizeEvery`

指定 API Gateway 快取授權者結果的時間 time-to-live (TTL) 期間 (以秒為單位)。如果您指定的值大於 0，則 API Gateway 會快取授權方回應。根據預設，API Gateway 會將此屬性設為 300。值的上限為 3600 (1 小時)。

類型：整數

必要：否

預設值：

AWS CloudFormation兼容性：此屬性是唯一的，AWS SAM並且沒有相AWS CloudFormation等的屬性。

`ValidationExpression`

指定驗證運算式以驗證傳入的身份

類型：字串

必要：否

AWS CloudFormation兼容性：此屬性是唯一的，AWS SAM並且沒有相AWS CloudFormation等的屬性。

範例

CognitoAuthIdentity

YAML

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

LambdaRequestAuthorizer

設定 Lambda 授權器，以透過 Lambda 函數控制對您 API 的存取。

如需詳細資訊和範例，請參閱 使用 AWS SAM 範本控制 API 存取。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
DisableFunctionDefaultPermissions: Boolean
FunctionArn: String
FunctionInvokeRole: String
FunctionPayloadType: String
Identity: LambdaRequestAuthorizationIdentity
```

屬性

`DisableFunctionDefaultPermissions`

指定`true`以防止 AWS SAM 自動建立`AWS::Lambda::Permissions`資源，以便在資`AWS::Serverless::Api`源與授權者 Lambda 函數之間佈建權限。

預設值：`false`

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`FunctionArn`

指定提供 API 授權之 Lambda 函數的函數 ARN。

> **Note**
>
> AWS SAM 當指定給時`FunctionArn`，會自動建立`AWS::Lambda::Permissions`資源`AWS::Serverless::Api`。資`AWS::Lambda::Permissions`源會在您的 API 和授權者 Lambda 函數之間佈建權限。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`FunctionInvokeRole`

將授權者認證新增至 Lambda 授權者的 OpenApi 定義。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`FunctionPayloadType`

此屬性可用來定義 API 的 Lambda 授權器類型。

有效值：`TOKEN` 或 `REQUEST`

類型：字串

必要：否

預設：`TOKEN`

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Identity`

此屬性可用於在授權者`IdentitySource`的傳入請求中指定。只有在屬性設定為時，才需要此`FunctionPayloadType`屬性`REQUEST`。

類型:LambdaRequestAuthorizationIdentity

必要：有條件

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

LambdaRequestAuth

YAML

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

LambdaRequestAuthorizationIdentity

此屬性可用於在授權者 IdentitySource 的傳入請求中指定。如需有關的詳細資訊， IdentitySource 請參閱ApiGateway 授權者 OpenApi 擴充功能。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

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

屬性

`Context`

將給定的上下文字符串轉換為格式的映射表達式`context.contextString`。

類型：清單

必要：否

AWS CloudFormation兼容性：此屬性是唯一的，AWS SAM並且沒有相AWS CloudFormation等的屬性。

`Headers`

將標頭轉換為以逗號分隔的格式對應運算式`method.request.header.name`字串。

類型：清單

必要：否

AWS CloudFormation兼容性：此屬性是唯一的，AWS SAM並且沒有相AWS CloudFormation等的屬性。

`QueryStrings`

將指定的查詢字串轉換為以逗號分隔的格式對應運算式`method.request.querystring.queryString`字串。

類型：清單

必要：否

AWS CloudFormation兼容性：此屬性是唯一的，AWS SAM並且沒有相AWS CloudFormation等的屬性。

`ReauthorizeEvery`

指定 API Gateway 快取授權者結果的時間 time-to-live (TTL) 期間 (以秒為單位)。如果您指定的值大於 0，則 API Gateway 會快取授權方回應。根據預設，API Gateway 會將此屬性設為 300。值的上限為 3600 (1 小時)。

類型：整數

必要：否

預設值：

AWS CloudFormation兼容性：此屬性是唯一的，AWS SAM並且沒有相AWS CloudFormation等的屬性。

`StageVariables`

將指定的階段變數轉換為以逗號分隔的格式對應運算式`stageVariables.stageVariable`字串。

類型：清單

必要：否

AWS CloudFormation兼容性：此屬性是唯一的，AWS SAM並且沒有相AWS CloudFormation等的屬性。

範例

LambdaRequestIdentity

YAML

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

LambdaTokenAuthorizer

設定 Lambda 授權器，以透過 Lambda 函數控制對您 API 的存取。

如需詳細資訊和範例，請參閱 使用 AWS SAM 範本控制 API 存取。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
DisableFunctionDefaultPermissions: Boolean
```

```
FunctionArn: String
FunctionInvokeRole: String
FunctionPayloadType: String
Identity: LambdaTokenAuthorizationIdentity
```

屬性

`DisableFunctionDefaultPermissions`

指定`true`以防止 AWS SAM 自動建立`AWS::Lambda::Permissions`資源，以便在資`AWS::Serverless::Api`源與授權者 Lambda 函數之間佈建權限。

預設值：`false`

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`FunctionArn`

指定提供 API 授權之 Lambda 函數的函數 ARN。

> **Note**
>
> AWS SAM 當指定給時`FunctionArn`，會自動建立`AWS::Lambda::Permissions`資源`AWS::Serverless::Api`。資`AWS::Lambda::Permissions`源會在您的 API 和授權者 Lambda 函數之間佈建權限。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`FunctionInvokeRole`

將授權者認證新增至 Lambda 授權者的 OpenApi 定義。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`FunctionPayloadType`

此屬性可用來定義 API 的 Lambda 授權器類型。

有效值：`TOKEN` 或 `REQUEST`

類型：字串

必要：否

預設：`TOKEN`

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Identity`

此屬性可用於在授權者`IdentitySource`的傳入請求中指定。只有在屬性設定為時，才需要此`FunctionPayloadType`屬性`REQUEST`。

類型:LambdaTokenAuthorizationIdentity

必要：有條件

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

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

此屬性可用於在授權者 IdentitySource 的傳入請求中指定。如需有關的詳細資訊， IdentitySource 請參閱ApiGateway 授權者 OpenApi 擴充功能。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

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

屬性

`Header`

在定 OpenApi 義中指定「授權」的標頭名稱。

類型：字串

必要：否

預設值：授權

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`ReauthorizeEvery`

指定 API Gateway 快取授權者結果的時間 time-to-live (TTL) 期間 (以秒為單位)。如果您指定的值大於 0，則 API Gateway 會快取授權方回應。根據預設，API Gateway 會將此屬性設為 300。值的上限為 3600 (1 小時)。

類型：整數

必要：否

預設值：

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`ValidationExpression`

指定驗證運算式，以驗證傳入的身份。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

LambdaTokenIdentity

YAML

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

```
  ReauthorizeEvery: 30
```

ResourcePolicyStatement

為 API 的所有方法和路徑配置資源策略。如需有關資源政策的詳細資訊，請參閱《API Gateway 開發人員指南》中的使用 API Gateway 資源政策控制 API 的存取。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

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

屬性

`AwsAccountBlacklist`

要封鎖的 AWS 帳戶。

類型：字串的清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`AwsAccountWhitelist`

要允許的 AWS 帳戶。如需此屬性的使用範例，請參閱本頁底部的「範例」一節。

類型：字串的清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`CustomStatements`

要套用至此 API 的自訂資源政策陳述式清單。如需此屬性的使用範例，請參閱本頁底部的「範例」一節。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`IntrinsicVpcBlacklist`

要封鎖的虛擬私有雲端 (VPC) 清單，其中每個 VPC 都指定為參考，例如動態參考或`Ref`內建函數。如需此屬性的使用範例，請參閱本頁底部的「範例」一節。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`IntrinsicVpcWhitelist`

要允許的 VPC 清單，其中每個 VPC 都被指定為參考，例如動態參考或`Ref`內建函數。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`IntrinsicVpceBlacklist`

要封鎖的 VPC 端點清單，其中每個 VPC 端點都指定為參考，例如動態參考或`Ref`內建函數。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`IntrinsicVpceWhitelist`

要允許的 VPC 端點清單，其中每個 VPC 端點都指定為參考，例如動態參考或Ref內建函數。如需此屬性的使用範例，請參閱本頁底部的「範例」一節。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`IpRangeBlacklist`

要封鎖的 IP 位址或位址範圍。如需此屬性的使用範例，請參閱本頁底部的「範例」一節。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`IpRangeWhitelist`

要允許的 IP 位址或位址範圍。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`SourceVpcBlacklist`

要封鎖的來源 VPC 或 VPC 端點。來源 VPC 名稱必須以開頭，`"vpc-"`且來源 VPC 端點名稱必須以開頭。`"vpce-"`如需此屬性的使用範例，請參閱本頁底部的「範例」一節。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`SourceVpcWhitelist`

要允許的來源 VPC 或 VPC 端點。來源 VPC 名稱必須以開頭，`"vpc-"`且來源 VPC 端點名稱必須以開頭。`"vpce-"`

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

資源策略範例

下列範例會封鎖兩個 IP 位址和一個來源 VPC，並允許 AWS 帳戶。

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

一個 OpenAPI 文檔定義的 API。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

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

屬性

`Bucket`

存放 OpenAPI 檔案所在的 Amazon S3 儲存貯體的名稱。

類型：字串

必要：是

AWS CloudFormation兼容性：此屬性直接傳遞給`AWS::ApiGateway::RestApiS3Location`數據類型的`Bucket`屬性。

`Key`

OpenAPI 文件的 Amazon S3 密鑰。

類型：字串

必要：是

AWS CloudFormation兼容性：此屬性直接傳遞給AWS::ApiGateway::RestApiS3Location數據類型的Key屬性。

Version

對於已建立版本的物件，則為 OpenAPI 檔案的版本。

類型：字串

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給AWS::ApiGateway::RestApiS3Location數據類型的Version屬性。

範例

定義 Uri 範例

API 定義範例

YAML

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

CorsConfiguration

管理 API Gateway API 的跨來源資源共用 (CORS)。指定允許作為字符串的域或指定帶有其他 CRS 配置的字典。

> **Note**
>
> CORS 需要修 AWS SAM 改您的 OpenAPI 定義。在中建立內嵌 OpenAPI 定義以DefinitionBody開啟 CORS。如果CorsConfiguration在 OpenAPI 定義中設定，也在屬性層級設定，則會將它們 AWS SAM 合併。屬性層級的優先順序高於 OpenAPI 定義。

如需有關 CORS 的詳細資訊，請參閱 API Gateway 開發人員指南中的針對 API Gateway REST API 資源啟用 CO RS。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

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

屬性

`AllowCredentials`

布林值，指出是否允許要求包含認證。

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`AllowHeaders`

要允許的標頭字串。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`AllowMethods`

包含要允許的 HTTP 方法的字串。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`AllowOrigin`

允許的原始字符串。這可以是字串格式的逗號分隔清單。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`MaxAge`

包含緩存 CORS 預檢請求的秒數的字符串。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

CorsConfiguration

CORS 配置示例。這只是 AWS SAM 模板文件的一部分，顯示配置了 AWS::Serverless::Function CORS 的AWS::Serverless::Api定義和. 如果您使用 Lambda 代理整合或 HTTP 代理整合，您的後端必須傳回`Access-Control-Allow-Origin`、`Access-Control-Allow-Methods`、和`Access-Control-Allow-Headers`標頭。

YAML

```
Resources:
  ApiGatewayApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      Cors:
```

```
        AllowMethods: "'POST, GET'"
        AllowHeaders: "'X-Forwarded-For'"
        AllowOrigin: "'www.example.com'"
        MaxAge: "'600'"
        AllowCredentials: true
  ApiFunction: # Adds a GET method at the root resource via an Api event
    Type: AWS::Serverless::Function
    Properties:
      Events:
        ApiEvent:
          Type: Api
          Properties:
            Path: /
            Method: get
            RestApiId:
              Ref: ApiGatewayApi
      Runtime: python3.10
      Handler: index.handler
      InlineCode: |
        import json
        def handler(event, context):
          return {
          'statusCode': 200,
          'headers': {
            'Access-Control-Allow-Headers': 'Content-Type',
            'Access-Control-Allow-Origin': 'www.example.com',
            'Access-Control-Allow-Methods': 'POST, GET'
            },
          'body': json.dumps('Hello from Lambda!')
          }
```

DomainConfiguration

設定 API 的自訂網域。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
  BasePath: List
  NormalizeBasePath: Boolean
  CertificateArn: String
```

```
  DomainName: String
  EndpointConfiguration: String
  MutualTlsAuthentication: MutualTlsAuthentication
  OwnershipVerificationCertificateArn: String
  Route53: Route53Configuration
  SecurityPolicy: String
```

屬性

`BasePath`

要使用 Amazon API Gateway 網域名稱設定的基本路徑清單。

類型：清單

必要：否

預設值：/

AWS CloudFormation兼容性：此屬性類似於`AWS::ApiGateway::BasePathMapping`資源的`BasePath`屬性。 AWS SAM會建立多個`AWS::ApiGateway::BasePathMapping`資源，每個在此屬性中`BasePath`指定一個資源。

`NormalizeBasePath`

指出屬性定義的`BasePath`基本路徑中是否允許使用非英數字元。設定為時`True`，會從基本路徑中移除非英數字元。

`NormalizeBasePath`與`BasePath`屬性一起使用。

類型：布林值

必要：否

預設值：真

AWS CloudFormation兼容性：此屬性是唯一的，AWS SAM並且沒有相AWS CloudFormation等的屬性。

`CertificateArn`

此網域名稱端點的AWS受管憑證的 Amazon 資源名稱 (ARN)。 AWS Certificate Manager是唯一受支援的來源。

類型：字串

必要：是

AWS CloudFormation兼容性：此屬性類似於AWS::ApiGateway::DomainName資源的CertificateArn屬性。如果設定EndpointConfiguration為 REGIONAL (預設值)，則對CertificateArn映至RegionalCertificateArn中AWS::ApiGateway::DomainName。如果設定EndpointConfiguration為EDGE，則對CertificateArn映至CertificateArn中AWS::ApiGateway::DomainName。

其他注意事項：對於EDGE端點，您必須在「us-east-1AWS區域」中建立憑證。

DomainName

API Gateway API 的自訂網域名稱。不支援大寫字母。

AWS SAM設置此屬性時生成一個AWS::ApiGateway::DomainName資源。如需有關此案例的資訊，請參閱DomainName屬性已指定。如需有關已產生AWS CloudFormation資源的資訊，請參閱產生的 AWS CloudFormation 資源。

類型：字串

必要：是

AWS CloudFormation兼容性：此屬性直接傳遞給AWS::ApiGateway::DomainName資源的DomainName屬性。

EndpointConfiguration

定義要對應至自訂網域的 API Gateway 端點類型。此屬性的值決定了對應CertificateArn性質的方式AWS CloudFormation。

有效值：REGIONAL 或 EDGE

類型：字串

必要：否

預設：REGIONAL

AWS CloudFormation兼容性：此屬性是唯一的，AWS SAM並且沒有相AWS CloudFormation等的屬性。

`MutualTlsAuthentication`

自訂網域名稱的相互傳輸層安全性 (TLS) 驗證組態。

類型:MutualTlsAuthentication

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給`AWS::ApiGateway::DomainName`資源的`MutualTlsAuthentication`屬性。

`OwnershipVerificationCertificateArn`

ACM 核發之公有憑證的 ARN，用於驗證您的自訂網域的擁有權。只有當您設定相互 TLS，並指定 ACM 匯入或私有 CA 憑證 ARN 給. `CertificateArn`

類型：字串

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給`AWS::ApiGateway::DomainName`資源的`OwnershipVerificationCertificateArn`屬性。

`Route53`

定義 Amazon 路線 53 配置。

類型：路線 53 配置

必要：否

AWS CloudFormation兼容性：此屬性是唯一的，AWS SAM並且沒有相AWS CloudFormation等的屬性。

`SecurityPolicy`

TLS 版本加上此域名的密碼套件。

類型：字串

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給`AWS::ApiGateway::DomainName`資源的`SecurityPolicy`屬性。

範例

DomainName

DomainName 例子

YAML

```
Domain:
  DomainName: www.example.com
  CertificateArn: arn-example
  EndpointConfiguration: EDGE
  Route53:
    HostedZoneId: Z1PA6795UKMFR9
  BasePath:
    - foo
    - bar
```

Route53Configuration

設定 API 的路由 53 記錄集。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

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

屬性

`DistributionDomainName`

設定 API 自訂網域名稱的自訂分發。

類型：字串

必要：否

預設值：使用 API Gateway 分發。

AWS CloudFormation兼容性：此屬性直接傳遞給AWS::Route53::RecordSetGroup AliasTarget資源的DNSName屬性。

其他注意事項：CloudFront發行版的網域名稱。

EvaluateTargetHealth

當 EvaluateTargetHealth 為 true 時，別名記錄會繼承參考AWS資源的健全狀況，例如 Elastic Load Balancing 負載平衡器或託管區域中的其他記錄。

類型：布林值

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給AWS::Route53::RecordSetGroup AliasTarget資源的EvaluateTargetHealth屬性。

其他注意事項：當別名目標是 CloudFront 發佈時，您無法設定 EvaluateTargetHealth 為 true。

HostedZoneId

託管區域的 ID，您要在其中建立記錄。

請指定 HostedZoneName 或 HostedZoneId 其中之一。若有多個託管區域的網域名稱相同，則您必須使用 HostedZoneId 以明確指定託管區域。

類型：字串

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給AWS::Route53::RecordSetGroup RecordSet資源的HostedZoneId屬性。

HostedZoneName

您要在其中建立記錄的託管區域名稱。

請指定 HostedZoneName 或 HostedZoneId 其中之一。若有多個託管區域的網域名稱相同，則您必須使用 HostedZoneId 以明確指定託管區域。

類型：字串

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給AWS::Route53::RecordSetGroup RecordSet資源的HostedZoneName屬性。

IpV6

設定此屬性後，AWS SAM會為提供的建立AWS::Route53::RecordSet資源並將「類型」設定AAAA為 HostedZone。

類型：布林值

必要：否

AWS CloudFormation兼容性：此屬性是唯一的，AWS SAM並且沒有相AWS CloudFormation等的屬性。

Region

僅限延遲型資源記錄集：您建立資源的 Amazon EC2 區域，這是此資源記錄集指向的資源。資源一般是 AWS 資源 (例如 EC2 執行個體或 ELB 負載平衡器)，且以 IP 地址或 DNS 網域名稱表示 (視記錄類型而定)。

當 Amazon Route 53 收到網域名稱和類型的 DNS 查詢，而您已建立其延遲資源記錄集時，Route 53 會選取最低延遲介於最終使用者與相關聯 Amazon EC2 區域之間的延遲資源記錄集。Route 53 接著會傳回與所選資源記錄集相關聯的值。

注意下列事項：

- 每個延遲資源記錄集只能指定一個 ResourceRecord。
- 每個 Amazon EC2 區域都只能建立一個延遲資源記錄集。
- 您不必為所有的 Amazon EC2 區域建立延遲資源記錄集。Route 53 會從您建立延遲資源記錄集的區域中，選擇具有最佳延遲的區域。
- 您無法建立和延遲資源記錄集的 Name 和 Type 元素具有相同值的非延遲資源記錄集。

類型：字串

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給AWS::Route53::RecordSetGroupRecordSet數據類型的 Region屬性。

`SetIdentifier`

沒有簡單路由政策的資源記錄集：在具有相同名稱和類型組合的多個資源記錄集中用以區隔的識別碼，例如，多個加權資源記錄集名為 acme.example.com 且類型為 A。在一組具有相同名稱和類型的資源記錄集中，每個資源記錄集的 `SetIdentifier` 值都必須是唯一的。

如需有關路由政策的資訊，請參閱 Amazon Route 53 開發人員指南中的選擇路由政策。

類型：字串

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給`AWS::Route53::RecordSetGroupRecordSet`數據類型的 `SetIdentifier`屬性。

範例

基本範例

在此範例中，我們為 API 設定自訂網域和 Route 53 記錄集。

YAML

```
Resources:
  MyApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      Domain:
        DomainName: www.example.com
        CertificateArn: arn:aws:acm:us-east-1:123456789012:certificate/
abcdef12-3456-7890-abcd-ef1234567890
        EndpointConfiguration: REGIONAL
        Route53:
          HostedZoneId: ABCDEFGHIJKLMNOP
```

`EndpointConfiguration`

其餘 API 的端點類型。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
  Type: String
  VPCEndpointIds: List
```

屬性

Type

其餘 API 的端點類型。

有效值：EDGEREGIONAL或 PRIVATE

類型：字串

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給AWS::ApiGateway::RestApiEndpointConfiguration數據類型的Types屬性。

VPCEndpointIds

要針對其建立 Route53 別名的其他 API 的 VPC 擬私人雲端端點識別碼清單。

類型：列表

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給AWS::ApiGateway::RestApiEndpointConfiguration數據類型的VpcEndpointIds屬性。

範例

EndpointConfiguration

端點組態範例

YAML

```
EndpointConfiguration:
  Type: PRIVATE
  VPCEndpointIds:
```

```
    - vpce-123a123a
    - vpce-321a321a
```

AWS::Serverless::Application

將 Amazon S3 儲存貯體AWS Serverless Application Repository或來自 Amazon S3 儲存貯體的無伺服器應用程式嵌入為巢狀應用程式。嵌套應用程序部署為嵌套AWS::CloudFormation::Stack資源，其中可以包含多個其他資源，包括其他AWS::Serverless::Application資源。

> **Note**
>
> 當您部署到時 AWS CloudFormation，將您的 AWS SAM 資源 AWS SAM 轉換為 AWS CloudFormation 資源。如需詳細資訊，請參閱 產生的 AWS CloudFormation 資源。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
Type: AWS::Serverless::Application
Properties:
  Location: String | ApplicationLocationObject
  NotificationARNs: List
  Parameters: Map
  Tags: Map
  TimeoutInMinutes: Integer
```

屬性

`Location`

巢狀應用程式的範本 URL、檔案路徑或位置物件。

如果提供範本 URL，它必須遵循CloudFormation TemplateUrl 文件中指定的格式，並包含有效 CloudFormation 或 SAM 範本。ApplicationLocationObject可用來指定已發行至的應用程式AWS Serverless Application Repository。

如果提供了本機檔案路徑，範本必須經過包含`sam deploy`或`sam package`指令的工作流程，才能正確轉換應用程式。

類型：字符串 | ApplicationLocationObject

必要：是

AWS CloudFormation 兼容性：此屬性類似於AWS::CloudFormation::Stack資源的TemplateURL屬性。 CloudFormation 版本不需ApplicationLocationObject要從中擷取應用程式 AWS Serverless Application Repository。

`NotificationARNs`

現有 Amazon SNS 主題清單，其中會傳送有關堆疊事件的通知。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::CloudFormation::Stack資源的NotificationARNs屬性。

`Parameters`

應用程式參數值。

類型：地圖

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::CloudFormation::Stack資源的Parameters屬性。

`Tags`

映射（字符串到字符串），指定要添加到此應用程序的標籤。鍵和值僅限於字母數字字元。密鑰的長度可以是 1 到 127 個萬國碼字符，並且不能以 aws: 作為前綴。值的長度可以是 1 到 255 個萬國碼字元。

類型：地圖

必要：否

AWS CloudFormation 兼容性：此屬性類似於AWS::CloudFormation::Stack資源的Tags屬性。SAM 中的標籤屬性由鍵：值對組成; CloudFormation 其中包含標籤對象的列表。建立堆疊後，SAM 會自動將lambda:createdBy:SAM標籤新增至此應用程式。此外，如果這個應用程序是來自的 AWS Serverless Application Repository，那麼 SAM 也會自動添加兩個額外的標籤serverlessrepo:applicationId:*ApplicationId*和serverlessrepo:semanticVersion:*S*

`TimeoutInMinutes`

AWS CloudFormation 等待巢狀堆疊到達`CREATE_COMPLETE`狀態的時間長度 (以分鐘為單位)。預設值為無逾時。當 AWS CloudFormation 檢測到嵌套堆棧已達到`CREATE_COMPLETE`狀態時，它將嵌套堆棧資源標記為`CREATE_COMPLETE`在父堆棧中，並繼續創建父堆棧。如果逾時期限在巢狀堆疊到達之前到期`CREATE_COMPLETE`，則會將巢狀堆疊 AWS CloudFormation 標記為失敗，並同時復原巢狀堆疊和父系堆疊。

類型：整數

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::CloudFormation::Stack`資源的`TimeoutInMinutes`屬性。

傳回值

Ref

當此資源的邏輯 ID 被提供給`Ref`內在函數，它返回基礎`AWS::CloudFormation::Stack`資源的資源名稱。

若要取得有關使用`Ref`功能的更多資訊，請參閱《使AWS CloudFormation 用指南》*Ref*中的〈〉

Fn: GetAtt

`Fn::GetAtt` 會傳回此類型之指定屬性的值。以下為可用屬性及傳回值的範例。

若要取得有關使用的更多資訊`Fn::GetAtt`，請參閱使AWS CloudFormation 用指南*Fn::GetAtt*中的〈〉

`Outputs.ApplicationOutputName`

與名稱的堆棧輸出的值*ApplicationOutputName*。

範例

特區申請

使用無伺服器應用程式儲存區域中範本的應用程式

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
```

正常應用

來自 S3 網址的應用程序

YAML

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

ApplicationLocationObject

已發行至的應用程式AWS Serverless Application Repository。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

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

屬性

`ApplicationId`

應用程式的 Amazon Resource Name (ARN)。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`SemanticVersion`

應用程式的語義版本。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

我的應用

範例應用程式位置物

YAML

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

AWS::Serverless::Connector

設定兩個資源之間的權限。如需連接器的簡介，請參閱使用 AWS SAM 連接器管理資源權限。

如需產生 AWS CloudFormation 資源的詳細資訊，請參閱AWS CloudFormation指定時產生的資源 AWS::Serverless::Connector。

若要提供有關連接器的意見反應，請在serverless-application-model AWS GitHub 儲存庫中提交新問題。

> **Note**
>
> 當您部署到時 AWS CloudFormation，將您的 AWS SAM 資源 AWS SAM 轉換為 AWS CloudFormation 資源。如需詳細資訊，請參閱 產生的 AWS CloudFormation 資源。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列任一語法。

> **Note**
>
> 我們建議在大多數使用案例中使用嵌入式連接器語法。嵌入到源資源中，可以隨著時間的推移更容易閱讀和維護。當您需要引用不在同一個 AWS SAM 模板中的源資源（例如嵌套堆棧中的資源或共享資源）時，請使用`AWS::Serverless::Connector`語法。

嵌入式接頭

```
<source-resource-logical-id>:
  Connectors:
    <connector-logical-id>:
      Properties:
        Destination: ResourceReference | List of ResourceReference
        Permissions: List
        SourceReference: SourceReference
```

AWS::Serverless::Connector

```
Type: AWS::Serverless::Connector
Properties:
  Destination: ResourceReference | List of ResourceReference
  Permissions: List
  Source: ResourceReference
```

屬性

`Destination`

目標資源。

類型：ResourceReference| 列表 ResourceReference

必要：是

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Permissions`

允許來源資源對目標資源執行的權限類型。

`Read`包括允許從資源讀取資料的 AWS Identity and Access Management (IAM) 動作。

`Write`包括允許啟動資料並將資料寫入資源的 IAM 動作。

有效值：`Read` 或 `Write`

類型：清單

必要：是

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Source`

來源資源。使用`AWS::Serverless::Connector`語法時需要。

類型:ResourceReference

必要：有條件

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`SourceReference`

來源資源。

> **Note**
>
> 定義來源資源的其他屬性時，請搭配內嵌連接器語法使用。

類型:SourceReference

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

嵌入式接頭

下列範例使用內嵌連接器來定義 AWS Lambda 函`Write`數和 Amazon DynamoDB 表之間的資料連線：

```
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyTable:
    Type: AWS::Serverless::SimpleTable
  MyFunction:
    Type: AWS::Serverless::Function
    Connectors:
      MyConn:
        Properties:
          Destination:
            Id: MyTable
          Permissions:
            - Write
    ...
```

下列範例使用內嵌連接器來定義`Read`和`Write`權限：

```
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Connectors:
      MyConn:
        Properties:
          Destination:
            Id: MyTable
          Permissions:
            - Read
            - Write
  MyTable:
    Type: AWS::DynamoDB::Table
    ...
```

下列範例會使用內嵌連接器來定義具有以外屬性的來源資源`Id`：

```
Transform: AWS::Serverless-2016-10-31
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyApi:
    Type: AWS::Serverless::Api
    Connectors:
      ApitoLambdaConn:
        Properties:
          SourceReference:
            Qualifier: Prod/GET/foobar
          Destination:
            Id: MyTable
          Permissions:
            - Read
            - Write
  MyTable:
    Type: AWS::DynamoDB::Table
    ...
```

AWS::Serverless::Connector

下列範例會使用AWS::Serverless::Connector資源來讀取和寫入 Amazon DynamoDB 表格的 AWS Lambda 函數：

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

下列範例會使用AWS::Serverless::Connector資源讓 Lambda 函數寫入 Amazon SNS 主題，且兩個資源都位於相同範本中：

```
MyConnector:
  Type: AWS::Serverless::Connector
  Properties:
```

```
    Source:
      Id: MyLambda
    Destination:
      Id: MySNSTopic
    Permissions:
      - Write
```

下列範例會使用AWS::Serverless::Connector資源將 Amazon SNS 主題寫入 Lambda 函數，然後將所有資源寫入 Amazon DynamoDB 表格，並將所有資源放在相同範本中：

```
Transform: AWS::Serverless-2016-10-31
Resources:
  Topic:
    Type: AWS::SNS::Topic
    Properties:
      Subscription:
        - Endpoint: !GetAtt Function.Arn
          Protocol: lambda

  Function:
    Type: AWS::Serverless::Function
    Properties:
      Runtime: nodejs16.x
      Handler: index.handler
      InlineCode: |
        const AWS = require('aws-sdk');
        exports.handler = async (event, context) => {
          const docClient = new AWS.DynamoDB.DocumentClient();
          await docClient.put({
            TableName: process.env.TABLE_NAME,
            Item: {
              id: context.awsRequestId,
              event: JSON.stringify(event)
            }
          }).promise();
        };
      Environment:
        Variables:
          TABLE_NAME: !Ref Table

  Table:
    Type: AWS::Serverless::SimpleTable
```

```
  TopicToFunctionConnector:
    Type: AWS::Serverless::Connector
    Properties:
      Source:
        Id: Topic
      Destination:
        Id: Function
      Permissions:
        - Write

  FunctionToTableConnector:
    Type: AWS::Serverless::Connector
    Properties:
      Source:
        Id: Function
      Destination:
        Id: Table
      Permissions:
        - Write
```

以下是從上面的例子中轉換的 AWS CloudFormation 模板：

```
"FunctionToTableConnectorPolicy": {
  "Type": "AWS::IAM::ManagedPolicy",
  "Metadata": {
    "aws:sam:connectors": {
      "FunctionToTableConnector": {
        "Source": {
          "Type": "AWS::Lambda::Function"
        },
        "Destination": {
          "Type": "AWS::DynamoDB::Table"
        }
      }
    }
  },
  "Properties": {
    "PolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": [
```

```
            "dynamodb:PutItem",
            "dynamodb:UpdateItem",
            "dynamodb:DeleteItem",
            "dynamodb:BatchWriteItem",
            "dynamodb:PartiQLDelete",
            "dynamodb:PartiQLInsert",
            "dynamodb:PartiQLUpdate"
          ],
          "Resource": [
            {
              "Fn::GetAtt": [
                "MyTable",
                "Arn"
              ]
            },
            {
              "Fn::Sub": [
                "${DestinationArn}/index/*",
                {
                  "DestinationArn": {
                    "Fn::GetAtt": [
                      "MyTable",
                      "Arn"
                    ]
                  }
                }
              ]
            }
          ]
        },
        "Roles": [
          {
            "Ref": "MyFunctionRole"
          }
        ]
      }
    }
```

ResourceReference

資源類型使用之資AWS::Serverless::Connector源的參照。

> **Note**
>
> 對於相同範本中的資源，請提供Id. 對於不在相同範本中的資源，請使用其他屬性的組合。如需詳細資訊，請參閱 AWS SAM 連接器參考。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

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

屬性

`Arn`

資源的 ARN。

類型：字串

必要：有條件

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Id`

相同範本中資源的邏輯 ID。

> **Note**
>
> 指定`Id`時，如果連接器產生 AWS Identity and Access Management (IAM) 政策，則會從資源`Id`推斷與這些政策相關聯的 IAM 角色。如果`Id`未指定，請提供連接器`RoleName`的資源，以將產生的 IAM 政策附加到 IAM 角色。

類型：字串

必要：有條件

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Name`

資源的名稱。

類型：字串

必要：有條件

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Qualifier`

縮小其範圍之資源的限定詞。 `Qualifier`取代資源約束 ARN 結束時的`*`值。如需範例，請參閱叫用 Lambda 函數的 API Gateway。

> **Note**
>
> 限定詞定義會因資源類型而異。如需支援的來源和目標資源類型的清單，請參閱AWS SAM 連接器參考。

類型：字串

必要：有條件

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`QueueUrl`

Amazon SQS 佇列網址。此屬性僅適用於 Amazon SQS 資源。

類型：字串

必要：有條件

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`ResourceId`

資源的識別碼。例如，API Gateway API 識別碼。

類型：字串

必要：有條件

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`RoleName`

與資源相關聯的角色名稱。

> **Note**
>
> 指定`Id`時，如果連接器產生 IAM 政策，則會從資源`Id`推斷與這些政策相關聯的 IAM 角色。如果`Id`未指定，請提供連接器`RoleName`的資源，以將產生的 IAM 政策附加到 IAM 角色。

類型：字串

必要：有條件

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Type`

資源的 AWS CloudFormation 類型。如需詳細資訊，請前往AWS 資源和屬性類型參考。

類型：字串

必要：有條件

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

叫用 Lambda 函數的 API Gateway

下列範例使用AWS::Serverless::Connector資源允許 Amazon API Gateway 叫用 AWS Lambda 函數。

YAML

```
Transform: AWS::Serverless-2016-10-31
Resources:
  MyRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Effect: Allow
            Action: sts:AssumeRole
            Principal:
              Service: lambda.amazonaws.com
      ManagedPolicyArns:
        - !Sub arn:${AWS::Partition}:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole

  MyFunction:
    Type: AWS::Lambda::Function
    Properties:
      Role: !GetAtt MyRole.Arn
      Runtime: nodejs16.x
      Handler: index.handler
      Code:
        ZipFile: |
          exports.handler = async (event) => {
            return {
              statusCode: 200,
              body: JSON.stringify({
                "message": "It works!"
              }),
            };
```

```
          };

  MyApi:
    Type: AWS::ApiGatewayV2::Api
    Properties:
      Name: MyApi
      ProtocolType: HTTP

  MyStage:
    Type: AWS::ApiGatewayV2::Stage
    Properties:
      ApiId: !Ref MyApi
      StageName: prod
      AutoDeploy: True

  MyIntegration:
    Type: AWS::ApiGatewayV2::Integration
    Properties:
      ApiId: !Ref MyApi
      IntegrationType: AWS_PROXY
      IntegrationUri: !Sub arn:aws:apigateway:${AWS::Region}:lambda:path/2015-03-31/
functions/${MyFunction.Arn}/invocations
      IntegrationMethod: POST
      PayloadFormatVersion: "2.0"

  MyRoute:
    Type: AWS::ApiGatewayV2::Route
    Properties:
      ApiId: !Ref MyApi
      RouteKey: GET /hello
      Target: !Sub integrations/${MyIntegration}

  MyConnector:
    Type: AWS::Serverless::Connector
    Properties:
      Source: # Use 'Id' when resource is in the same template
        Type: AWS::ApiGatewayV2::Api
        ResourceId: !Ref MyApi
        Qualifier: prod/GET/hello # Or "*" to allow all routes
      Destination: # Use 'Id' when resource is in the same template
        Type: AWS::Lambda::Function
        Arn: !GetAtt MyFunction.Arn
      Permissions:
        - Write
```

```
Outputs:
  Endpoint:
    Value: !Sub https://${MyApi}.execute-api.${AWS::Region}.${AWS::URLSuffix}/prod/
hello
```

SourceReference

資源類型使用之來源AWS::Serverless::Connector資源的參照。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
Qualifier: String
```

屬性

`Qualifier`

縮小其範圍之資源的限定詞。 `Qualifier`取代資源約束 ARN 結束時的`*`值。

> **Note**
>
> 限定詞定義因資源類型而異。如需支援的來源和目標資源類型的清單，請參閱AWS SAM 連接器參考。

類型：字串

必要：有條件

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

下列範例會使用內嵌連接器來定義具有以外屬性的來源資源**`Id`**：

```
Transform: AWS::Serverless-2016-10-31
```

```
...
Resources:
  MyApi:
    Type: AWS::Serverless::Api
    Connectors:
      ApitoLambdaConn:
        Properties:
          SourceReference:
            Qualifier: Prod/GET/foobar
          Destination:
            Id: MyTable
          Permissions:
            - Read
            - Write
  MyTable:
    Type: AWS::DynamoDB::Table
    ...
```

AWS::Serverless::Function

建立 AWS Lambda 函數、 AWS Identity and Access Management (IAM) 執行角色，以及觸發函數的事件來源對應。

資AWS::Serverless::Function源也支援 `Metadata` resource 屬性，因此您可以指示建置應用程式所 AWS SAM 需的自訂執行階段。如需建立自訂執行階段的詳細資訊，請參閱〈〉使用自訂執行階段建置 Lambda 函數。

> **Note**
>
> 當您部署到時 AWS CloudFormation，將您的 AWS SAM 資源 AWS SAM 轉換為 AWS CloudFormation 資源。如需詳細資訊，請參閱 產生的 AWS CloudFormation 資源。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
Type: AWS::Serverless::Function
Properties:
  Architectures: List
```

```
  AssumeRolePolicyDocument: JSON
  AutoPublishAlias: String
  AutoPublishAliasAllProperties: Boolean
  AutoPublishCodeSha256: String
  CodeSigningConfigArn: String
  CodeUri: String | FunctionCode
  DeadLetterQueue: Map | DeadLetterQueue
  DeploymentPreference: DeploymentPreference
  Description: String
  Environment: Environment
  EphemeralStorage: EphemeralStorage
  EventInvokeConfig: EventInvokeConfiguration
  Events: EventSource
  FileSystemConfigs: List
  FunctionName: String
  FunctionUrlConfig: FunctionUrlConfig
  Handler: String
  ImageConfig: ImageConfig
  ImageUri: String
  InlineCode: String
  KmsKeyArn: String
  Layers: List
  LoggingConfig: LoggingConfig
  MemorySize: Integer
  PackageType: String
  PermissionsBoundary: String
  Policies: String | List | Map
  PropagateTags: Boolean
  ProvisionedConcurrencyConfig: ProvisionedConcurrencyConfig
  ReservedConcurrentExecutions: Integer
  Role: String
  RolePath: String
  Runtime: String
  RuntimeManagementConfig: RuntimeManagementConfig
  SnapStart: SnapStart
  Tags: Map
  Timeout: Integer
  Tracing: String
  VersionDescription: String
  VpcConfig: VpcConfig
```

屬性

`Architectures`

函數的指令集架構。

如需有關此屬性的詳細資訊，請參閱AWS Lambda 開發人員指南中的 Lambda 指令集架構。

有效值：其中一個`x86_64`或 `arm64`

類型：清單

必要：否

預設：`x86_64`

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::Function`資源的`Architectures`屬性。

`AssumeRolePolicyDocument`

添加 AssumeRolePolicyDocument 為此函數創建`Role`的默認值。如果未指定此屬性，請為此函數 AWS SAM 新增預設假設角色。

類型：JSON

必要：否

AWS CloudFormation 兼容性：此屬性類似於`AWS::IAM::Role`資源的`AssumeRolePolicyDocument`屬性。 AWS SAM 將此屬性新增至此函數產生的 IAM 角色。如果為此函數提供角色的 Amazon 資源名稱 (ARN)，則此屬性不會執行任何動作。

`AutoPublishAlias`

Lambda 別名的名稱。如需 Lambda 別名的詳細資訊，請參閱AWS Lambda 開發人員指南中的 Lambda 函數別名。如需使用此性質的範例，請參閱逐步部署無伺服器應用程。

AWS SAM 設置此屬性時生成AWS::Lambda::Version和AWS::Lambda::Alias資源。如需有關此案例的資訊，請參閱AutoPublishAlias 屬性已指定。如需有關已產生 AWS CloudFormation 資源的一般資訊，請參閱產生的 AWS CloudFormation 資源。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`AutoPublishAliasAllProperties`

指定何時創建一[AWS::Lambda::Version](#)個新的。何時`true`，會在修改 Lambda 函數中的任何屬性時建立新的 Lambda 版本。如果`false`，只有在修改下列任何屬性時，才會建立新的 Lambda 版本：

- `Environment`, `MemorySize`, 或 `SnapStart`.
- 導致`Code`屬性更新的任何變更，例如`CodeDictImageUri`、或`InlineCode`。

需要`AutoPublishAlias`定義此屬性。

如果`AutoPublishSha256`也指定，則其行為優先於`AutoPublishAliasAllProperties: true`。

類型：布林值

必要：否

預設值：`false`

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`AutoPublishCodeSha256`

使用時，此字串會與`CodeUri`值搭配使用，以判斷是否需要發行新的 Lambda 版本。此屬性通常用於解決以下部署問題：部署套件存放在 Amazon S3 位置，並由具有更新 Lambda 函數程式碼的新部署套件取代，但該`CodeUri`屬性保持不變 (與新的部署套件上傳到新 Amazon S3 位置並變更為新位置相反)。`CodeUri`

這個問題是由具有下列特性的 AWS SAM 範本所標示：

- `DeploymentPreference`物件已配置為漸進式部署 (如中所述[逐步部署無伺服器應用程](#))
- 內`AutoPublishAlias`容已設定，且不會在部署之間變更
- 內`CodeUri`容已設定，且不會在部署之間變更。

在這個案例中，更新會`AutoPublishCodeSha256`導致成功建立新的 Lambda 版本。不過，將無法辨識部署到 Amazon S3 的新函數程式碼。若要辨識新的函數程式碼，請考慮在 Amazon S3 儲存貯體中使用版本控制。指定 Lambda 函數的`Version`屬性，並將儲存貯體設定為永遠使用最新的部署套件。

在這個案例中，若要成功觸發逐步部署，您必須提供的唯一值AutoPublishCodeSha256。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

CodeSigningConfigArn

AWS::Lambda::CodeSigningConfig資源的 ARN，用於啟用此功能的代碼簽名。如需程式碼簽章的詳細資訊，請參閱為您的應用程式設定 AWS SAM 程式碼簽章。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::Function資源的CodeSigningConfigArn屬性。

CodeUri

函數的程式碼。接受的值包括：

- 該函數的 Amazon S3 URI。例如 s3://bucket-123456789/sam-app/1234567890abcdefg。
- 函數的本機路徑。例如 hello_world/。
- FunctionCode 物件。

> **Note**
>
> 如果您提供函數的 Amazon S3 URI 或FunctionCode物件，則必須參考有效的 Lambda 部署套件。
> 如果您提供本機檔案路徑，請使 AWS SAMCLI用在部署時上傳本機檔案。如需進一步了解，請參閱如何在部署時上傳本地文件 AWS SAMCLI。
> 如果在CodeUri屬性中使用內置函數， AWS SAM 將無法正確解析值。請考慮改用AWS::Language擴充功能轉換。

類型:[字串 |FunctionCode]

必要：有條件限制。當設定PackageType為時Zip，需要其中一InlineCode個CodeUri或。

AWS CloudFormation 兼容性：此屬性類似於AWS::Lambda::Function資源的 Code屬性。巢狀 Amazon S3 屬性的命名方式不同。

`DeadLetterQueue`

設定亞馬遜簡單通知服務 (Amazon SNS) 主題或亞 Amazon Simple Queue Service (Amazon SQS) 佇列，Lambda 會在其中傳送無法處理的事件。如需有關無效字母佇列功能的詳細資訊，請參閱開發人員指南中的無效字母佇列。AWS Lambda

> **Note**
>
> 如果 Lambda 函數的事件來源是 Amazon SQS 佇列，請為來源佇列設定無效字母佇列，而不是 Lambda 函數。您為函數設定的無效字母佇列會用於函數的非同步叫用佇列，而不是用於事件來源佇列。

類型:地圖 | DeadLetterQueue

必要：否

AWS CloudFormation 兼容性：此屬性類似於AWS::Lambda::Function資源的DeadLetterConfig屬性。 AWS CloudFormation 在類型是從衍生而來的TargetArn，而在中， AWS SAM 您必須將類型與TargetArn.

`DeploymentPreference`

啟用漸進式 Lambda 部署的設定。

如果指定了DeploymentPreference對象，則 AWS SAM 創建一個AWS::CodeDeploy::Application被調用ServerlessDeploymentApplication（每個堆棧一個），一個AWS::CodeDeploy::DeploymentGroup被調*<function-logical-id>*DeploymentGroup用和一個AWS::IAM::Role調用CodeDeployServiceRole。

類型:DeploymentPreference

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

另請參閱：如需此性質的詳細資訊，請參閱逐步部署無伺服器應用程。

`Description`

函數的敘述。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::Function`資源的`Description`屬性。

`Environment`

執行階段環境的組態。

類型：Environment

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::Function`資源的`Environment`屬性。

`EphemeralStorage`

一個物件，指定您的 Lambda 函數可用的磁碟空間 (以 MB 為單位) `/tmp`。

如需有關此屬性的詳細資訊，請參閱AWS Lambda 開發人員指南中的 Lambda 執行環境。

類型:EphemeralStorage

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::Function`資源的`EphemeralStorage`屬性。

`EventInvokeConfig`

描述事件的物件叫用 Lambda 函數上的組態。

類型:EventInvokeConfiguration

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Events`

指定觸發此函數的事件。事件包含一個類型和一組依賴於類型的屬性。

類型:EventSource

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`FileSystemConfigs`

指定 Amazon 彈性檔案系統 (Amazon EFS) 檔案系統連線設定的FileSystemConfig物件清單。

如果您的範本包含AWS::EFS::MountTarget資源，您還必須指定DependsOn資源屬性，以確保在函數之前建立或更新掛載目標。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::Function`資源的`FileSystemConfigs`屬性。

`FunctionName`

函數名稱。如果您未指定名稱，則會為您產生唯一的名稱。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::Function`資源的`FunctionName`屬性。

`FunctionUrlConfig`

描述函數 URL 的物件。函數 URL 是一個 HTTPS 端點，您可以使用它來調用您的函數。

如需詳細資訊，請參閱AWS Lambda 開發人員指南中的函式 URL。

類型:FunctionUrlConfig

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Handler`

程式碼中呼叫以開始執行的函數。只有在屬性設定為時，才需要此`PackageType`屬性`Zip`。

類型：字串

必要：有條件

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::Function`資源的`Handler`屬性。

`ImageConfig`

用來設定 Lambda 容器映像設定的物件。如需詳細資訊，請參閱AWS Lambda 開發人員指南中的搭配 Lambda 使用容器映像。

類型:ImageConfig

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::Function`資源的`ImageConfig`屬性。

`ImageUri`

用於 Lambda 函數容器映像的 Amazon Elastic Container Registry (Amazon ECR) 存儲庫的 URI。只有在屬性設定為時，此`PackageType`屬性才適用`Image`，否則會忽略此屬性。如需詳細資訊，請參閱AWS Lambda 開發人員指南中的搭配 Lambda 使用容器映像。

> **Note**
>
> 如果內`PackageType`容設定為`Image`，則`ImageUri`為必要項目，或者您必須使用 AWS SAM 範本檔案中的必要`Metadata`項目來建置應用程式。如需詳細資訊，請參閱 默認構建 AWS SAM。

使用必要Metadata條目構建應用程序的優先級高於ImageUri，因此，如果同時指定兩者，則將ImageUri被忽略。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::FunctionCode數據類型的ImageUri屬性。

InlineCode

直接在範本中撰寫的 Lambda 函數程式碼。只有在屬性設定為時，此PackageType屬性才適用Zip，否則會忽略此屬性。

> **Note**
>
> 如果將PackageType內容設定為 Zip (預設值)，則需要其中一InlineCode個CodeUri或。

類型：字串

必要：有條件

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::FunctionCode數據類型的ZipFile屬性。

KmsKeyArn

Lambda 用來加密和解密函數環境變數的 AWS Key Management Service (AWS KMS) 金鑰的 ARN。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::Function資源的KmsKeyArn屬性。

Layers

這個函數應該使用的 LayerVersion ARN 的列表。此處指定的順序是執行 Lambda 函數時匯入的順序。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::Function資源的Layers屬性。

LoggingConfig

該功能的 Amazon CloudWatch 日誌配置設置。

類型:LoggingConfig

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::Function資源的LoggingConfig屬性。

MemorySize

每次呼叫函式所配置的記憶體大小 (以 MB 為單位)。

類型：整數

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::Function資源的MemorySize屬性。

PackageType

Lambda 函數的部署套件類型。如需詳細資訊，請參閱AWS Lambda 開發人員指南中的 Lambda 部署套件。

備註：

1. 如果將此屬性設定為 Zip (預設值)，則會InlineCode套用CodeUri或，並忽ImageUri略。

2. 如果將此屬性設定為Image，則僅ImageUri套用，CodeUri且InlineCode都會忽略和。存放函數容器映像所需的 Amazon ECR 儲存庫可由. AWS SAMCLI 如需詳細資訊，請參閱 sam deploy。

有效值：Zip 或 Image

類型：字串

必要：否

預設：Zip

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::Function資源的PackageType屬性。

PermissionsBoundary

權限界限的 ARN，用於此函數的執行角色。只有在為您產生角色時，此屬性才有效。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::IAM::Role資源的PermissionsBoundary屬性。

Policies

此功能的權限原則。政策會附加至函數的預設 AWS Identity and Access Management (IAM) 執行角色。

此屬性接受單一值或值清單。允許數值包括：

- AWS SAM策略範本.
- AWS 受管理策略或客戶管理策略ARN的。
- 下列清單中 AWS 受管理策略的名稱。
- 格式化YAML為地圖的內嵌 IAM 政策。

> **Note**
>
> 如果您設定Role屬性，則會忽略此屬性。

類型：字串 | 清單 | 地圖

必要：否

AWS CloudFormation 兼容性：此屬性類似於AWS::IAM::Role資源的Policies屬性。

`PropagateTags`

指出是否要將標籤從`Tags`屬性傳遞至您AWS::Serverless::Function產生的資源。指`True`定在產生的資源中傳播標籤。

類型：布林值

必要：否

預設：`False`

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`ProvisionedConcurrencyConfig`

函數別名的佈建並行配置。

> **Note**
>
> `ProvisionedConcurrencyConfig`只有在設定時`AutoPublishAlias`才能指定。否則，會導致錯誤。

類型:ProvisionedConcurrencyConfig

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::Alias`資源的`ProvisionedConcurrencyConfig`屬性。

`ReservedConcurrentExecutions`

您要為函數保留的並行執行數目上限。

如需有關此屬性的詳細資訊，請參閱AWS Lambda 開發人員指南中的 Lambda 函數調整。

類型：整數

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::Function`資源的`ReservedConcurrentExecutions`屬性。

`Role`

IAM 角色的 ARN，用作此函數的執行角色。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性類似於`AWS::Lambda::Function`資源的`Role`屬性。這在中是必需的 AWS CloudFormation ，但在中不是 AWS SAM。如果未指定角色，則會使用邏輯 ID 為為您建立角色*`<function-logical-id>`*`Role`。

`RolePath`

函數 IAM 執行角色的路徑。

為您產生角色時，請使用此屬性。當角色與`Role`屬性一起指定時，請勿使用。

類型：字串

必要：有條件

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::IAM::Role`資源的`Path`屬性。

`Runtime`

函數的執行時間的識別符。只有在屬性設定為時，才需要此`PackageType`屬性`Zip`。

> **Note**
>
> 如果您指定此屬性的`provided`識別碼，您可以使用 `Metadata` resource 屬性 AWS SAM 來指示建置此函數所需的自訂執行階段。如需建立自訂執行階段的詳細資訊，請參閱〈〉使用自訂執行階段建置 Lambda 函數。

類型：字串

必要：有條件

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::Function`資源的`Runtime`屬性。

`RuntimeManagementConfig`

設定 Lambda 函數的執行階段管理選項，例如執行階段環境更新、復原行為，以及選取特定執行階段版本。若要深入了解，請參閱AWS Lambda 開發人員指南中的 Lambda 執行階段更新

類型:RuntimeManagementConfig

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::Function`資源的`RuntimeManagementConfig`屬性。

`SnapStart`

建立任何新 Lambda 函數版本的快照。快照是初始化函數的緩存狀態，包括其所有依賴項。該函數只被初始化一次，緩存狀態將被重複用於所有 future 的調用，通過減少您的函數必須初始化的次數來提高應用程序性能。若要深入了解，請參閱AWS Lambda 開發人員指南SnapStart中的使用 Lambda 改善啟動效能。

類型:SnapStart

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::Function`資源的`SnapStart`屬性。

`Tags`

映射（字符串到字符串），指定添加到此函數的標籤。如需有關標籤的有效金鑰和值的詳細資訊，請參閱AWS Lambda 開發人員指南中的標籤金鑰和值需求。

建立堆疊時， AWS SAM 會自動將`lambda:createdBy:SAM`標籤新增至此 Lambda 函數，以及為此函數產生的預設角色。

類型：地圖

必要：否

AWS CloudFormation 兼容性：此屬性類似於`AWS::Lambda::Function`資源的`Tags`屬性。中的`Tags`屬性 AWS SAM 由鍵值對組成（而在 AWS CloudFormation 此屬性中由對`Tag`象列表組成）。此外， AWS SAM 會自動將`lambda:createdBy:SAM`標籤新增至此 Lambda 函數，以及為此函數產生的預設角色。

`Timeout`

函數在停止前可執行的時間上限 (以秒為單位)。

類型：整數

必要：否

預設：3

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::Function`資源的`Timeout`屬性。

`Tracing`

字串；指定函數的 X-Ray 追蹤模式。

- `Active`— 啟動該功能的 X-Ray 追蹤。
- `Disabled`— 停用 X-Ray 的功能.
- `PassThrough`— 啟動該功能的 X-Ray 追蹤。抽樣決策委託給下游服務。

如果指定為`Active`或`PassThrough`且未設定`Role`內容，則會將`arn:aws:iam::aws:policy/AWSXrayWriteOnlyAccess`原則 AWS SAM 新增至為您建立的 Lambda 執行角色。

如需有關 X-Ray 的詳細資訊，請參閱AWS Lambda 開發人員指南 AWS X-Ray中的AWS Lambda 搭配使用。

有效值：[`Active`| `Disabled` |`PassThrough`]

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性類似於`AWS::Lambda::Function`資源的`TracingConfig`屬性。

`VersionDescription`

指定新增至新 Lambda 版本資源的`Description`欄位。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::Version`資源的`Description`屬性。

`VpcConfig`

啟用此功能以存取虛擬私有雲 (VPC) 內私有資源的組態。

類型:VpcConfig

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::Function`資源的`VpcConfig`屬性。

傳回值

Ref

將此資源的邏輯 ID 提供給`Ref`內建函數時，會傳回基礎 Lambda 函數的資源名稱。

若要取得有關使用`Ref`功能的更多資訊，請參閱《使AWS CloudFormation 用指南》*`Ref`*中的〈〉

Fn: GetAtt

`Fn::GetAtt` 會傳回此類型之指定屬性的值。以下為可用屬性及傳回值的範例。

若要取得有關使用的更多資訊`Fn::GetAtt`，請參閱使AWS CloudFormation 用指南*`Fn::GetAtt`*中的〈〉

`Arn`

基礎 Lambda 函數的 ARN。

範例

簡單的功能

以下是 Amazon S3 儲存貯體中套件類型 `Zip` (預設) AWS::Serverless::Function 資源和函數程式碼的基本範例。

YAML

```
Type: AWS::Serverless::Function
Properties:
  Handler: index.handler
  Runtime: python3.9
  CodeUri: s3://bucket-name/key-name
```

函數屬性示例

以下是使用InlineCode、、、Layers和Api事件來源AWS::Serverless::Function的封裝類型 Zip (預設) 範例。Tracing Policies Amazon EFS

YAML

```
Type: AWS::Serverless::Function
DependsOn: MyMountTarget        # This is needed if an AWS::EFS::MountTarget resource
 is declared for EFS
Properties:
  Handler: index.handler
  Runtime: python3.9
  InlineCode: |
    def handler(event, context):
      print("Hello, world!")
  ReservedConcurrentExecutions: 30
  Layers:
    - Ref: MyLayer
  Tracing: Active
  Timeout: 120
  FileSystemConfigs:
    - Arn: !Ref MyEfsFileSystem
      LocalMountPath: /mnt/EFS
  Policies:
    - AWSLambdaExecute
    - Version: '2012-10-17'
      Statement:
        - Effect: Allow
          Action:
            - s3:GetObject
            - s3:GetObjectACL
          Resource: 'arn:aws:s3:::my-bucket/*'
  Events:
    ApiEvent:
```

```
      Type: Api
      Properties:
        Path: /path
        Method: get
```

ImageConfig例子

以下是封裝類型之 `ImageConfig` Lambda 函數的範例`Image`。

YAML

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

RuntimeManagementConfig 例子

設定為根據目前行為更新其執行階段環境的 Lambda 函數：

```
TestFunction
  Type: AWS::Serverless::Function
  Properties:
    ...
    Runtime: python3.9
    RuntimeManagementConfig:
      UpdateRuntimeOn: Auto
```

設定為在函數更新時更新其執行階段環境的 Lambda 函數：

```
TestFunction
  Type: AWS::Serverless::Function
  Properties:
    ...
    Runtime: python3.9
```

```
    RuntimeManagementConfig:
      UpdateRuntimeOn: FunctionUpdate
```

設定為手動更新其執行階段環境的 Lambda 函數：

```
TestFunction
  Type: AWS::Serverless::Function
  Properties:
    ...
    Runtime: python3.9
    RuntimeManagementConfig:
      RuntimeVersionArn: arn:aws:lambda:us-
east-1::runtime:4c459dd0104ee29ec65dcad056c0b3ddbe20d6db76b265ade7eda9a066859b1e
      UpdateRuntimeOn: Manual
```

SnapStart例子

針對 future 版本 SnapStart 開啟的 Lambda 函數範例：

```
TestFunc
  Type: AWS::Serverless::Function
  Properties:
    ...
    SnapStart:
      ApplyOn: PublishedVersions
```

DeadLetterQueue

指定 SQS 佇列或 SNS 主題，以便 AWS Lambda (Lambda) 在無法處理事件時傳送事件。如需無效字母佇列功能的詳細資訊，請參閱AWS Lambda 開發人員指南中的無效字母佇列。

SAM 會自動將適當的權限新增至您的 Lambda 函數執行角色，讓 Lambda 服務存取資源。sqs:SendMessage 將會新增至 SQS 佇列和 SNS：針對 SNS 主題發佈。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
  TargetArn: String
  Type: String
```

屬性

`TargetArn`

Amazon SQS 佇列或 Amazon SNS 主題的 Amazon Resource Name (ARN)。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::FunctionDeadLetterConfig`數據類型的`TargetArn`屬性。

`Type`

無效字母佇列的類型。

有效值：`SNS`、`SQS`

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

DeadLetterQueue

SNS 主題的無效信件佇列範例。

YAML

```
DeadLetterQueue:
  Type: SNS
  TargetArn: arn:aws:sns:us-east-2:123456789012:my-topic
```

DeploymentPreference

指定要啟用漸進 Lambda 部署的組態。如需設定漸進 Lambda 部署的詳細資訊，請參閱逐步部署無伺服器應用程。

> **Note**
>
> 您必須`AutoPublishAlias`在中指定一個AWS::Serverless::Function才能使用`DeploymentPreference`對象，否則將導致錯誤。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
Alarms: List
Enabled: Boolean
Hooks: Hooks
PassthroughCondition: Boolean
Role: String
TriggerConfigurations: List
Type: String
```

屬性

`Alarms`

您希望由部署引發的任何錯誤觸發的 CloudWatch 警示清單。

此屬性接受`Fn::If`內在函數。如需使用的範例範本，請參閱本主題底部的「範例」一節`Fn::If`。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Enabled`

是否已啟用此部署偏好設定。

類型：布林值

必要：否

預設值：真

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Hooks`

驗證流量轉移之前和之後執行的 Lambda 函數。

類型：掛鉤

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`PassthroughCondition`

如果為 True，且啟用此部署偏好設定，則函數的「條件」會傳遞至所產生的 CodeDeploy 資源。一般而言，您應該將其設定為 True。否則，即使函數的「條件」解析為 False，也會建立 CodeDeploy 資源。

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Role`

CodeDeploy 將用於流量轉移的 IAM 角色 ARN。如果提供了 IAM 角色，則不會建立該角色。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`TriggerConfigurations`

您要與部署群組產生關聯的觸發程式組態清單。用於通知有關生命週期事件的 SNS 主題。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::CodeDeploy::DeploymentGroup資源的TriggerConfigurations屬性。

Type

目前有兩種部署類型：線性和 Canary。如需可用部署類型的詳細資訊，請參閱逐步部署無伺服器應用程。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

DeploymentPreference 具有交通前和交通後掛鉤。

包含流量前後掛接的範例部署偏好設定。

YAML

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

DeploymentPreference 與 Fn:: 如果內在函數

用Fn::If於設定警示的範例部署偏好設定。在此範例中，Alarm1將會設定如果MyCondition是true，Alarm2且如果MyCondition是，則Alarm5會設定false。

YAML

```
DeploymentPreference:
  Enabled: true
  Type: Canary10Percent10Minutes
  Alarms:
    Fn::If:
      - MyCondition
      - - Alarm1
      - - Alarm2
        - Alarm5
```

Hooks

驗證流量轉移之前和之後執行的 Lambda 函數。

> **Note**
>
> 此屬性中參考的 Lambda 函數會設定所產生資AWS::Lambda::Alias源的CodeDeployLambdaAliasUpdate物件。如需詳細資訊，請參閱《AWS CloudFormation 使用指南》中的「CodeDeployLambdaAliasUpdate 策略」。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

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

屬性

`PostTraffic`

流量轉移後運行的 Lambda 函數。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`PreTraffic`

在流量轉移之前執行的 Lambda 函數。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

掛鉤

掛接函數示例

YAML

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

EventInvokeConfiguration

非同步 Lambda 別名或版本叫用的組態選項。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
  DestinationConfig: EventInvokeDestinationConfiguration
  MaximumEventAgeInSeconds: Integer
  MaximumRetryAttempts: Integer
```

屬性

`DestinationConfig`

組態物件，指定在 Lambda 處理過後事件的目標。

類型:EventInvokeDestinationConfiguration

必要：否

AWS CloudFormation 兼容性：此屬性類似於`AWS::Lambda::EventInvokeConfig`資源的`DestinationConfig`屬性。SAM 需要一個不存在的額外參數「類型」 CloudFormation。

`MaximumEventAgeInSeconds`

Lambda 傳送至函數以進行處理的請求時間上限。

類型：整數

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::EventInvokeConfig`資源的`MaximumEventAgeInSeconds`屬性。

`MaximumRetryAttempts`

函數傳回錯誤之前要重試的次數上限。

類型：整數

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::EventInvokeConfig`資源的`MaximumRetryAttempts`屬性。

範例

MaximumEventAgeInSeconds

MaximumEventAgeInSeconds 例子

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

組態物件，指定在 Lambda 處理過後事件的目標。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
  OnFailure: OnFailure
  OnSuccess: OnSuccess
```

屬性

`OnFailure`

處理失敗事件的目標。

類型:OnFailure

必要：否

AWS CloudFormation 兼容性：此屬性類似於`AWS::Lambda::EventInvokeConfig`資源的`OnFailure`屬性。需要`Type`一個額外的僅 SAM 屬性。

`OnSuccess`

已順利處理的事件目標。

類型:OnSuccess

必要：否

AWS CloudFormation 兼容性：此屬性類似於AWS::Lambda::EventInvokeConfig資源的OnSuccess屬性。需要Type一個額外的僅 SAM 屬性。

範例

OnSuccess

OnSuccess 例子

YAML

```
EventInvokeConfig:
  DestinationConfig:
    OnSuccess:
      Type: SQS
      Destination: arn:aws:sqs:us-west-2:012345678901:my-queue
    OnFailure:
      Type: Lambda
      Destination: !GetAtt DestinationLambda.Arn
```

OnFailure

處理失敗事件的目標。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

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

屬性

Destination

目標資源的 Amazon Resource Name (ARN)。

類型：字串

必要：有條件

AWS CloudFormation 兼容性：此屬性類似於`AWS::Lambda::EventInvokeConfig`資源的`OnFailure`屬性。SAM 會將任何必要的權限新增至與此函數相關聯的自動產生 IAM 角色，以存取此屬性中參照的資源。

附加說明：如果類型為 Lambda/EventBridge，則需要目的地。

`Type`

目標中參照的資源類型。支援的類型有`SQS``SNS``Lambda`、、和`EventBridge`。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

其他注意事項：如果類型為 SQS/SNS，且內`Destination`容保留空白，則 SAM 會 auto 產生 SQS/SNS 資源。若要參照資源，請使*`<function-logical-id>`*`.DestinationQueue`用 SQS 或 *`<function-logical-id>`*`.DestinationTopic` SNS。如果類型是 Lambda/EventBridge，則`Destination`是必需的。

範例

EventInvoke SQS 和 Lambda 目的地的組態範例

在此範例中，沒有為 SQS OnSuccess 組態指定目的地，因此 SAM 會隱含地建立 SQS 佇列並新增任何必要的權限。此外，在此範例中，在範本檔案中宣告的 Lambda 資源目標也會在 OnFailure 組態中指定，因此 SAM 會將必要的權限新增至此 Lambda 函數，以呼叫目標 Lambda 函數。

YAML

```
EventInvokeConfig:
  DestinationConfig:
    OnSuccess:
      Type: SQS
    OnFailure:
      Type: Lambda
      Destination: !GetAtt DestinationLambda.Arn  # Arn of a Lambda function declared
 in the template file.
```

EventInvoke SNS 目的地的組態範例

在此範例中，會為 OnSuccess 組態的範本檔案中宣告的 SNS 主題提供目的地。

YAML

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

OnSuccess

已順利處理的事件目標。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

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

屬性

`Destination`

目標資源的 Amazon Resource Name (ARN)。

類型：字串

必要：有條件

AWS CloudFormation 兼容性：此屬性類似於`AWS::Lambda::EventInvokeConfig`資源的`OnSuccess`屬性。SAM 會將任何必要的權限新增至與此函數相關聯的自動產生 IAM 角色，以存取此屬性中參照的資源。

附加說明：如果類型為 Lambda/EventBridge，則需要目的地。

Type

目標中參照的資源類型。支援的類型有SQSSNSLambda、、和EventBridge。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

其他注意事項：如果類型為 SQS/SNS，且內Destination容保留空白，則 SAM 會 auto 產生 SQS/SNS 資源。若要參照資源，請使*<function-logical-id>*.DestinationQueue用 SQS 或 *<function-logical-id>*.DestinationTopic SNS。如果類型是 Lambda/EventBridge，則Destination是必需的。

範例

EventInvoke SQS 和 Lambda 目的地的組態範例

在此範例中，沒有為 SQS OnSuccess 組態指定目的地，因此 SAM 會隱含地建立 SQS 佇列並新增任何必要的權限。此外，在此範例中，在範本檔案中宣告的 Lambda 資源目標也會在 OnFailure 組態中指定，因此 SAM 會將必要的權限新增至此 Lambda 函數，以呼叫目標 Lambda 函數。

YAML

```
EventInvokeConfig:
  DestinationConfig:
    OnSuccess:
      Type: SQS
    OnFailure:
      Type: Lambda
      Destination: !GetAtt DestinationLambda.Arn  # Arn of a Lambda function declared
 in the template file.
```

EventInvoke SNS 目的地的組態範例

在此範例中，會為 OnSuccess 組態的範本檔案中宣告的 SNS 主題提供目的地。

YAML

```
EventInvokeConfig:
```

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

EventSource

描述觸發函數之事件來源的物件。每個事件都包含一個類型和一組依賴於該類型的屬性。如需有關每個事件來源屬性的詳細資訊，請參閱與該類型對應的主題。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
  Properties: AlexaSkill | Api | CloudWatchEvent | CloudWatchLogs | Cognito
| DocumentDB | DynamoDB | EventBridgeRule | HttpApi | IoTRule | Kinesis | MQ | MSK
| S3 | Schedule | ScheduleV2 | SelfManagedKafka | SNS | SQS
  Type: String
```

屬性

`Properties`

描述此事件對應屬性的物件。性質集必須符合定義的「類型」。

類型:AlexaSkill| API | | Cognito CloudWatchEventCloudWatchLogs| DocumentDB | | | IOTRULE | 室內運 Dynamo DB EventBridgeRuleHttpApi| MQ | MSK | 排程 | 排程 2 | 社交網站 | SQS SelfManagedKafka

必要：是

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Type`

事件類型。

有效值：`AlexaSkill`,`Api`,`CloudWatchEvent`,`CloudWatchLogs`,`Cognito`,,`DocumentDB`,`DynamoDB`,`Ever SQS`

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

API 事件

使用 API 事件的範例

YAML

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

AlexaSkill

描述AlexaSkill事件來源類型的物件。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
  SkillId: String
```

屬性

SkillId

您 Alexa 技能的技能識別碼。如需技能 ID 的詳細資訊，請參閱 Alexa 技能套件文件中的設定 Lambda 函數的觸發器。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

AlexaSkillTrigger

技能事件範例

YAML

```
AlexaSkillEvent:
  Type: AlexaSkill
```

Api

描述Api事件來源類型的物件。如果已定義AWS::Serverless::Api資源，路徑和方法值必須對應至 API OpenAPI 定義中的作業。

如果沒AWS::Serverless::Api有定義，函數的輸入和輸出是 HTTP 請求和 HTTP 響應的表示。

例如，使用 JavaScript API，響應的 statusCode 和主體可以通過返回與密鑰 StatusCode 和 body 的對象來控制。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
  Auth: ApiFunctionAuth
  Method: String
  Path: String
  RequestModel: RequestModel
  RequestParameters: List of [ String | RequestParameter ]
  RestApiId: String
  TimeoutInMillis: Integer
```

屬性

`Auth`

此特定 API + 路徑 + 方法的身份驗證配置。

當未指定或覆蓋默`ApiKeyRequired`認`DefaultAuthorizer`設置時，用於在單個路徑上覆蓋 API 的設`DefaultAuthorizer`置身份驗證配置非常有用。

類型:ApiFunctionAuth

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Method`

呼叫此函數的 HTTP 方法。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Path`

呼叫此函式的 Uri 路徑。必須以開頭`/`。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`RequestModel`

請求模型用於此特定的 API + 路徑 + 方法。這應該引用AWS::Serverless::Api資源`Models`部分中指定的模型的名稱。

類型:RequestModel

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`RequestParameters`

請求此特定 API + 路徑 + 方法的參數配置。所有參數名稱必須以開頭，`method.request`且必須限制為`method.request.header`、`method.request.querystring`、或`method.request.path`。

清單可以同時包含參數名稱字串和RequestParameter物件。對於字串，`Required`和`Caching`屬性預設為`false`。

類型：[字串 | RequestParameter] 清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`RestApiId`

RestApi 資源的標識符，其中必須包含具有給定路徑和方法的操作。一般而言，這會設定為參照此範本中定義的AWS::Serverless::Api資源。

如果您未定義此屬性，請使用產生的`OpenApi`文件 AWS SAM 建立預設AWS::Serverless::Api資源。該資源包含由不指定的相同範本中的`Api`事件所定義的所有路徑和方法的聯集`RestApiId`。

這不能引用另一個模板中定義的AWS::Serverless::Api資源。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`TimeoutInMillis`

自訂介於 50 和 29,000 毫秒之間的逾時。

> **Note**
>
> 當您指定這個屬性時，請 AWS SAM 修改您的 OpenAPI 定義。OpenAPI 定義必須以內嵌方式使用`DefinitionBody`屬性來指定。

類型：整數

必要：否

預設值：29,000 毫秒或 29 秒

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

基本範例

YAML

```
Events:
  ApiEvent:
    Type: Api
    Properties:
      Path: /path
      Method: get
      RequestParameters:
        - method.request.header.Authorization
        - method.request.querystring.keyword:
            Required: true
            Caching: false
```

ApiFunctionAuth

在事件層級設定特定 API、路徑和方法的授權。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
  ApiKeyRequired: Boolean
  AuthorizationScopes: List
  Authorizer: String
  InvokeRole: String
  OverrideApiAuth: Boolean
```

```
ResourcePolicy: ResourcePolicyStatement
```

屬性

`ApiKeyRequired`

需要此 API、路徑和方法的 API 金鑰。

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`AuthorizationScopes`

要套用至此 API、路徑和方法的授權範圍。

如果您已指定屬性套用的任何範圍，您指定的範圍將覆寫該`DefaultAuthorizer`屬性套用的任何範圍。

類型：列表

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Authorizer`

對`Authorizer`於特定功能。

如果您為`AWS::Serverless::Api`資源指定了全域授權者，則可以將設`Authorizer`定為覆寫授權者。`NONE`如需範例，請參閱覆寫 Amazon API Gateway REST API 的全球授權者。

> **Note**
>
> 如果您使用`AWS::Serverless::Api`資源的`DefinitionBody`屬性來描述您的 API，則必須使`OverrideApiAuth`用 for 覆寫您`Authorizer`的全域授權者。如需詳細資訊，請參閱`OverrideApiAuth`。

有效值：AWS_IAMNONE、或 AWS SAM 範本中定義之任何授權者的邏輯 ID。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

InvokeRole

指定用InvokeRole於AWS_IAM授權的。

類型：字串

必要：否

預設：CALLER_CREDENTIALS

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

其他注意事項：CALLER_CREDENTIALS映射到arn:aws:iam::*:user/*，它使用呼叫者認證來調用端點。

OverrideApiAuth

指定true為可覆寫資AWS::Serverless::Api源的全域授權者組態。只有在您指定全域授權者並使用AWS::Serverless::Api資源的DefinitionBody屬性來描述您的 API 時，才需要此屬性。

> **Note**
>
> 當您指定OverrideApiAuth為時true， AWS SAM 會以、或ResourcePolicy提供的任何值覆寫您的ApiKeyRequired全域授權者。Authorizer因此，使用時也必須至少指定其中一個屬性OverrideApiAuth。如需範例，請參閱 指定 DefinitionBody 為 AWS::Serverless::Api 時覆寫全域授權者。

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`ResourcePolicy`

在 API 上設定此路徑的資源策略。

類型:ResourcePolicyStatement

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

功能身份驗證

下列範例會指定函式層級的授權。

YAML

```
Auth:
  ApiKeyRequired: true
  Authorizer: NONE
```

覆寫 Amazon API Gateway REST API 的全球授權者

您可以為`AWS::Serverless::Api`資源指定全域授權者。以下是設定全域預設授權者的範例：

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyApiWithLambdaRequestAuth:
    Type: AWS::Serverless::Api
    Properties:
      ...
      Auth:
        Authorizers:
          MyLambdaRequestAuth:
            FunctionArn: !GetAtt MyAuthFn.Arn
        DefaultAuthorizer: MyLambdaRequestAuth
```

若要覆寫 AWS Lambda 函數的預設授權者，您可以指定Authorizer為NONE。以下是範例：

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  ...
  MyFn:
    Type: AWS::Serverless::Function
    Properties:
      ...
      Events:
        LambdaRequest:
          Type: Api
          Properties:
            RestApiId: !Ref MyApiWithLambdaRequestAuth
            Method: GET
            Auth:
              Authorizer: NONE
```

指定 DefinitionBody 為 AWS::Serverless::Api 時覆寫全域授權者

當使用DefinitionBody屬性來描述您的AWS::Serverless::Api資源時，先前的覆蓋方法不起作用。以下是針對AWS::Serverless::Api資源使用DefinitionBody屬性的範例：

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyApiWithLambdaRequestAuth:
    Type: AWS::Serverless::Api
    Properties:
      ...
      DefinitionBody:
        swagger: 2.0
        ...
        paths:
          /lambda-request:
            ...
      Auth:
        Authorizers:
          MyLambdaRequestAuth:
            FunctionArn: !GetAtt MyAuthFn.Arn
```

```
        DefaultAuthorizer: MyLambdaRequestAuth
```

若要覆寫全域授權者，請使用OverrideApiAuth屬性。下列範例會使用OverrideApiAuth提供的值覆寫全域授權者：Authorizer

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyApiWithLambdaRequestAuth:
    Type: AWS::Serverless::Api
    Properties:
      ...
      DefinitionBody:
        swagger: 2-0
        ...
        paths:
          /lambda-request:
            ...
      Auth:
        Authorizers:
          MyLambdaRequestAuth:
            FunctionArn: !GetAtt MyAuthFn.Arn
        DefaultAuthorizer: MyLambdaRequestAuth

    MyAuthFn:
      Type: AWS::Serverless::Function
      ...

    MyFn:
      Type: AWS::Serverless::Function
        Properties:
          ...
          Events:
            LambdaRequest:
              Type: Api
              Properties:
                RestApiId: !Ref MyApiWithLambdaRequestAuth
                Method: GET
                Auth:
                  Authorizer: NONE
                  OverrideApiAuth: true
                Path: /lambda-token
```

ResourcePolicyStatement

為 API 的所有方法和路徑配置資源策略。如需有關資源政策的詳細資訊，請參閱《API Gateway 開發人員指南》中的使用 API Gateway 資源政策控制 API 的存取。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

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

屬性

`AwsAccountBlacklist`

要封鎖的 AWS 帳戶。

類型：字串的清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`AwsAccountWhitelist`

要允許的 AWS 帳戶。如需此屬性的使用範例，請參閱本頁底部的「範例」一節。

類型：字串的清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`CustomStatements`

要套用至此 API 的自訂資源政策陳述式清單。如需此屬性的使用範例，請參閱本頁底部的「範例」一節。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`IntrinsicVpcBlacklist`

要封鎖的虛擬私有雲端 (VPC) 清單，其中每個 VPC 都指定為參考，例如動態參考或Ref內建函數。如需此屬性的使用範例，請參閱本頁底部的「範例」一節。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`IntrinsicVpcWhitelist`

要允許的 VPC 清單，其中每個 VPC 都被指定為參考，例如動態參考或Ref內建函數。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`IntrinsicVpceBlacklist`

要封鎖的 VPC 端點清單，其中每個 VPC 端點都指定為參考，例如動態參考或Ref內建函數。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`IntrinsicVpceWhitelist`

要允許的 VPC 端點清單，其中每個 VPC 端點都指定為參考，例如動態參考或Ref內建函數。如需此屬性的使用範例，請參閱本頁底部的「範例」一節。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`IpRangeBlacklist`

要封鎖的 IP 位址或位址範圍。如需此屬性的使用範例，請參閱本頁底部的「範例」一節。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`IpRangeWhitelist`

要允許的 IP 位址或位址範圍。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`SourceVpcBlacklist`

要封鎖的來源 VPC 或 VPC 端點。來源 VPC 名稱必須以開頭，`"vpc-"`且來源 VPC 端點名稱必須以開頭。`"vpce-"`如需此屬性的使用範例，請參閱本頁底部的「範例」一節。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`SourceVpcWhitelist`

要允許的來源 VPC 或 VPC 端點。來源 VPC 名稱必須以開頭，`"vpc-"`且來源 VPC 端點名稱必須以開頭。`"vpce-"`

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

資源策略範例

下列範例會封鎖兩個 IP 位址和一個來源 VPC，並允許 AWS 帳戶。

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

為特定的 API + 路徑 + 方法配置請求模型。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
Model: String
Required: Boolean
ValidateBody: Boolean
ValidateParameters: Boolean
```

屬性

`Model`

在的「模型」屬性中定義的模型名稱AWS::Serverless::Api。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Required`

在給定 API 端點的 OpenApi 定義的參數部分中添加`required`屬性。

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`ValidateBody`

指定 API Gateway 是否使用`Model`來驗證要求主體。如需詳細資訊，請參閱 API Gateway 開發人員指南中的在 API Gateway 中啟用要求驗證。

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`ValidateParameters`

指定 API Gateway 是否使用`Model`來驗證要求路徑參數、查詢字串和標頭。如需詳細資訊，請參閱 API Gateway 開發人員指南中的在 API Gateway 中啟用要求驗證。

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

請求型號

請求模型示例

YAML

```
RequestModel:
  Model: User
  Required: true
  ValidateBody: true
  ValidateParameters: true
```

RequestParameter

為特定的 API + 路徑 + 方法配置請求參數。

需要為請求參數指定Required或Caching屬性

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
Caching: Boolean
Required: Boolean
```

屬性

Caching

將cacheKeyParameters區段新增至 API Gateway OpenApi 定義

類型：布林值

必要：有條件

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

Required

此欄位指定是否需要參數

類型：布林值

必要：有條件

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

請求參數

設定要求參數的範例

YAML

```
RequestParameters:
```

```
  - method.request.header.Authorization:
      Required: true
      Caching: true
```

CloudWatchEvent

描述CloudWatchEvent事件來源類型的物件。

AWS Serverless Application Model (AWS SAM) 在設定此事件類型時產生AWS::Events::Rule資源。

重要注意事項：EventBridgeRule是要使用的首選事件源類型，而不是CloudWatchEvent。EventBridgeRule並CloudWatchEvent使用相同的基礎服務，API 和 AWS CloudFormation 資源。但是，只 AWS SAM 會將對新功能的支援增加到EventBridgeRule。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
Enabled: Boolean
EventBusName: String
Input: String
InputPath: String
Pattern: EventPattern
State: String
```

屬性

Enabled

指出系統是否已啟用規則。

若要停用規則，請將此屬性設定為false。

> **Note**
>
> 指定Enabled或State性質，但不能同時指定兩者。

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性類似於AWS::Events::Rule資源的State屬性。如果此屬性設定為，true則 AWS SAM 傳遞ENABLED，否則會傳遞DISABLED。

EventBusName

與此規則相關聯的事件匯流排。如果您省略此屬性， AWS SAM 會使用預設事件匯流排。

類型：字串

必要：否

預設值：預設事件匯流排

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Events::Rule資源的EventBusName屬性。

Input

傳遞到目標的有效 JSON 文字。如果您使用此屬性，事件文字本身不會有任何內容傳遞到目標。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Events::Rule Target資源的Input屬性。

InputPath

如果您不想將整個匹配的事件傳遞給目標，請使用該InputPath屬性來描述要傳遞的事件的哪個部分。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Events::Rule Target資源的InputPath屬性。

Pattern

說明哪些事件會路由到指定目標。如需詳細資訊，請參閱 Amazon EventBridge 使用者指南 EventBridge中的事件和事件模式。

類型:EventPattern

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Events::Rule資源的EventPattern屬性。

State

規則的狀態。

接受的值：DISABLED | ENABLED

> **Note**
>
> 指定Enabled或State性質，但不能同時指定兩者。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Events::Rule資源的State屬性。

範例

CloudWatchEvent

以下是CloudWatchEvent事件來源類型的範例。

YAML

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

CloudWatchLogs

描述CloudWatchLogs事件來源類型的物件。

此事件會產生AWS::Logs::SubscriptionFilter資源並指定訂閱篩選器，並將其與指定的記錄群組產生關聯。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
FilterPattern: String
LogGroupName: String
```

屬性

FilterPattern

限制傳遞至目標 AWS 資源的項目的篩選運算式。如需篩選條件模式語法的詳細資訊，請參閱篩選條件及模式語法。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Logs::SubscriptionFilter資源的FilterPattern屬性。

LogGroupName

要與訂閱篩選條件關聯的日誌群組。如果篩選器模式符合記錄事件，則會篩選所有上傳至此記錄群組的記錄事件，並傳送至指定的 AWS 資源。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Logs::SubscriptionFilter資源的LogGroupName屬性。

範例

訂閱過濾器

訂閱過濾器示例

YAML

```
CWLog:
  Type: CloudWatchLogs
  Properties:
    LogGroupName:
      Ref: CloudWatchLambdaLogsGroup
    FilterPattern: My pattern
```

Cognito

描述Cognito事件來源類型的物件。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
  Trigger: List
  UserPool: String
```

屬性

`Trigger`

新使用者集區的 Lambda 觸發組態資訊。

類型：清單

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Cognito::UserPool`資源的`LambdaConfig`屬性。

`UserPool`

在相同範本中 UserPool 定義的參照

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

Cognito 事件

Cognito 事件範例

YAML

```
CognitoUserPoolPreSignup:
  Type: Cognito
  Properties:
    UserPool:
      Ref: MyCognitoUserPool
    Trigger: PreSignUp
```

DocumentDB

描述DocumentDB事件來源類型的物件。如需詳細資訊，請參閱AWS Lambda 開發人員指南中的 AWS Lambda 與 Amazon DocumentDB 搭配使用。

語法

若要在 AWS SAM 範本中宣告此實體，請使用下列語法。

YAML

```
BatchSize: Integer
Cluster: String
CollectionName: String
DatabaseName: String
Enabled: Boolean
FilterCriteria: FilterCriteria
FullDocument: String
MaximumBatchingWindowInSeconds: Integer
```

```
SecretsManagerKmsKeyId: String
SourceAccessConfigurations: List
StartingPosition: String
StartingPositionTimestamp: Double
```

屬性

`BatchSize`

要在單一批次中擷取的最大項目數。

類型：整數

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::EventSourceMapping`資源的 `BatchSize`屬性。

`Cluster`

Amazon 文檔數據庫集群的亞馬遜資源名稱（ARN）。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::EventSourceMapping`資源的 `EventSourceArn`屬性。

`CollectionName`

要在資料庫內使用的集合名稱。如果您未指定集合，Lambda 會使用所有集合。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::EventSourceMappingDocumentDBEventSourceConfig`數據類型的`CollectionName`屬性。

`DatabaseName`

要在 Amazon DocumentDB 叢集內使用的資料庫名稱。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMappingDocumentDBEventSourceConfig數據類型的DatabaseName屬性。

Enabled

如果true，則事件來源對映處於作用中狀態。若要暫停輪詢和呼叫，請將設定為false。

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的 Enabled屬性。

FilterCriteria

定義決定 Lambda 是否應該處理事件之準則的物件。如需詳細資訊，請參閱AWS Lambda 開發人員指南中的 Lambda 事件篩選。

類型:FilterCriteria

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的 FilterCriteria屬性。

FullDocument

決定在文件更新操作期間，Amazon DocumentDB 會傳送到您的事件串流的內容。如果設定為UpdateLookup，Amazon DocumentDB 會傳送描述變更的增量，以及整份文件的副本。否則，Amazon DocumentDB 只發送包含更改的部分文檔。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMappingDocumentDBEventSourceConfig數據類型的FullDocument屬性。

`MaximumBatchingWindowInSeconds`

調用函式前收集記錄的最長時間 (單位為秒)。

類型：整數

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::EventSourceMapping`資源的 `MaximumBatchingWindowInSeconds`屬性。

`SecretsManagerKmsKeyId`

來自 AWS 秘密管理員的客戶管理金鑰的 AWS Key Management Service (AWS KMS) 金鑰識別碼。當您將來自秘密管理員的客戶管理金鑰與不包含`kms:Decrypt`權限的 Lambda 執行角色使用時，此金鑰為必要條件。

此屬性的值是 UUID。例如：`1abc23d4-567f-8ab9-cde0-1fab234c5d67`。

類型：字串

必要：有條件

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`SourceAccessConfigurations`

驗證通訊協定或虛擬主機的陣列。使用資SourceAccessConfigurations料類型指定此項目。

對於`DocumentDB`事件來源類型，唯一有效的組態類型為`BASIC_AUTH`。

- `BASIC_AUTH`— 存儲您的經紀人憑據的 Secrets Manager 秘密. 對於此類型，認證必須採用下列格`{"username": "your-username", "password": "your-password"}`式：只允許一個類型`BASIC_AUTH`的對象。

類型：清單

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::EventSourceMapping`資源的 `SourceAccessConfigurations`屬性。

`StartingPosition`

要從中開始讀取的串流位置。

- AT_TIMESTAMP— 指定開始讀取記錄的時間。
- LATEST— 只讀新記錄。
- TRIM_HORIZON— 處理所有可用記錄。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的 StartingPosition屬性。

StartingPositionTimestamp

開始讀取的時間，以 Unix 時間秒為單位。定義StartingPositionTimestamp何時StartingPosition被指定為AT_TIMESTAMP。

類型：Double

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的 StartingPositionTimestamp屬性。

範例

Amazon DocumentDB 事件源

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
    ...
      Events:
        MyDDBEvent:
          Type: DocumentDB
          Properties:
            Cluster: "arn:aws:rds:us-west-2:123456789012:cluster:docdb-2023-01-01"
            BatchSize: 10
            MaximumBatchingWindowInSeconds: 5
```

```
            DatabaseName: "db1"
            CollectionName: "collection1"
            FullDocument: "UpdateLookup"
            SourceAccessConfigurations:
              - Type: BASIC_AUTH
                URI: "arn:aws:secretsmanager:us-west-2:123456789012:secret:doc-db"
```

DynamoDB

描述DynamoDB事件來源類型的物件。如需詳細資訊，請參閱AWS Lambda 開發人員指南中的 AWS Lambda 與 Amazon DynamoDB 搭配使用。

AWS SAM 設置此事件類型時生成一個AWS::Lambda::EventSourceMapping資源。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
BatchSize: Integer
BisectBatchOnFunctionError: Boolean
DestinationConfig: DestinationConfig
Enabled: Boolean
FilterCriteria: FilterCriteria
FunctionResponseTypes: List
MaximumBatchingWindowInSeconds: Integer
MaximumRecordAgeInSeconds: Integer
MaximumRetryAttempts: Integer
ParallelizationFactor: Integer
StartingPosition: String
StartingPositionTimestamp: Double
Stream: String
TumblingWindowInSeconds: Integer
```

屬性

`BatchSize`

要在單一批次中擷取的最大項目數。

類型：整數

必要：否

預設值：100

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的BatchSize屬性。

下限：1

上限：1000

BisectBatchOnFunctionError

如果函數傳回錯誤，請將批次分成兩個，然後重試。

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的BisectBatchOnFunctionError屬性。

DestinationConfig

用於捨棄記錄的亞馬遜簡單佇列服務 (Amazon SQS) 佇列或亞馬遜 Simple Notification Service (Amazon SNS) 主題目的地。

類型:DestinationConfig

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的DestinationConfig屬性。

Enabled

停用事件來源映射以暫停輪詢和叫用。

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的Enabled屬性。

`FilterCriteria`

定義決定 Lambda 是否應該處理事件之準則的物件。如需詳細資訊，請參閱AWS Lambda 開發人員指南中的AWS Lambda 事件篩選。

類型:FilterCriteria

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::EventSourceMapping`資源的`FilterCriteria`屬性。

`FunctionResponseTypes`

目前套用至事件來源對應的回應類型清單。如需詳細資訊，請參閱《AWS Lambda 開發人員指南》中的報告批次項目失敗。

有效值：`ReportBatchItemFailures`

類型：列表

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::EventSourceMapping`資源的`FunctionResponseTypes`屬性。

`MaximumBatchingWindowInSeconds`

調用函式前收集記錄的最長時間 (單位為秒)。

類型：整數

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::EventSourceMapping`資源的`MaximumBatchingWindowInSeconds`屬性。

`MaximumRecordAgeInSeconds`

Lambda 傳送至函數進行處理的記錄保留時間上限。

類型：整數

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的MaximumRecordAgeInSeconds屬性。

MaximumRetryAttempts

當函數傳回錯誤時，重試的次數上限。

類型：整數

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的MaximumRetryAttempts屬性。

ParallelizationFactor

從每個碎片同時處理的批次數。

類型：整數

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的ParallelizationFactor屬性。

StartingPosition

要從中開始讀取的串流位置。

- AT_TIMESTAMP— 指定開始讀取記錄的時間。
- LATEST— 只讀新記錄。
- TRIM_HORIZON— 處理所有可用記錄。

有效值：AT_TIMESTAMP | LATEST | TRIM_HORIZON

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的StartingPosition屬性。

StartingPositionTimestamp

開始讀取的時間，以 Unix 時間秒為單位。定義StartingPositionTimestamp何時StartingPosition被指定為AT_TIMESTAMP。

類型：Double

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的StartingPositionTimestamp屬性。

Stream

DynamoDB 資源資料流的 Amazon 資源名稱 (ARN)。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的EventSourceArn屬性。

TumblingWindowInSeconds

處理視窗的持續時間 (以秒為單位)。有效範圍為 1 至 900 分鐘 (15 分鐘)。

如需詳細資訊，請參閱AWS Lambda 開發人員指南中的翻轉視窗。

類型：整數

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的TumblingWindowInSeconds屬性。

範例

現有動態資料表的事 DynamoDB 來源

帳戶中已存在之 DynamoDB 資料表的 DynamoDB 事件來源。 AWS

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
```

在範本中宣告的 DynamoDB 表格的事件

針對在相同範本檔案中宣告的 DynamoDB 資料表的 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事件來源類型的物件，可將無伺服器函數設定為 Amazon EventBridge 規則的目標。有關更多信息，請參閱什麼是 Amazon EventBridge？ 在 Amazon 用 EventBridge 戶指南。

AWS SAM 設置此事件類型時生成一個AWS::Events::Rule資源。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

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

```
  InputTransformer: InputTransformer
  Pattern: EventPattern
  RetryPolicy: RetryPolicy
  RuleName: String
  State: String
  Target: Target
```

屬性

`DeadLetterConfig`

設定 Amazon Simple Queue Service (Amazon SQS) 佇列，在目標叫用失敗後 EventBridge 傳送事件。例如，當將事件傳送至不存在的 Lambda 函數，或是沒 EventBridge 有足夠的權限無法呼叫 Lambda 函數時，呼叫可能會失敗。如需詳細資訊，請參閱 Amazon 使用 EventBridge 者指南中的事件重試政策和使用無效字母佇列。

> **Note**
>
> 資AWS::Serverless::Function源類型具有類似的資料類型`DeadLetterQueue`，可處理成功叫用目標 Lambda 函數之後發生的失敗。這些失敗類型的範例包括 Lambda 節流，或 Lambda 目標函數傳回的錯誤。如需有關函數`DeadLetterQueue`屬性的詳細資訊，請參閱AWS Lambda 開發人員指南中的無效字母佇列。

類型:DeadLetterConfig

必要：否

AWS CloudFormation 兼容性：此屬性類似於`AWS::Events::RuleTarget`數據類型的`DeadLetterConfig`屬性。此屬性的 AWS SAM 版本包括其他子屬性，以防您想 AWS SAM 要為您建立無效字母佇列。

`EventBusName`

與此規則相關聯的事件匯流排。如果省略此屬性， AWS SAM 會使用預設事件匯流排。

類型：字串

必要：否

預設值：預設事件匯流排

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Events::Rule資源的EventBusName屬性。

Input

傳遞到目標的有效 JSON 文字。如果您使用此屬性，事件文字本身不會有任何內容傳遞到目標。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Events::Rule Target資源的Input屬性。

InputPath

如果您不想將整個匹配的事件傳遞給目標，請使用該InputPath屬性來描述要傳遞的事件的哪個部分。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Events::Rule Target資源的InputPath屬性。

InputTransformer

此設定能讓您以特定事件資料為基礎，向目標提供自訂輸入。您可從事件擷取一或多組鍵/值對，然後使用該資料將自訂輸入傳送至目標。如需詳細資訊，請參閱 Amazon EventBridge 使用者指南中的 Amazon EventBridge 輸入轉換。

類型:InputTransformer

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Events::RuleTarget數據類型的InputTransformer屬性。

Pattern

說明哪些事件會路由到指定目標。如需詳細資訊，請參閱 Amazon EventBridge EventBridge 使用者指南中的 Amazon EventBridge 事件和事件模式。

類型:EventPattern

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Events::Rule資源的EventPattern屬性。

RetryPolicy

包含重試政策設定相關資訊的 RetryPolicy 物件。如需詳細資訊，請參閱 Amazon 使用 EventBridge 者指南中的事件重試政策和使用無效字母佇列。

類型:RetryPolicy

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Events::RuleTarget數據類型的RetryPolicy屬性。

RuleName

規則的名稱。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Events::Rule資源的Name屬性。

State

規則的狀態。

接受的值：DISABLED | ENABLED

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Events::Rule資源的State 屬性。

Target

觸發規則時 EventBridge 呼叫的 AWS 資源。您可以使用此屬性來指定目標的邏輯 ID。如果未指定此屬性，則 AWS SAM 會產生目標的邏輯 ID。

類型：Target

必要：否

AWS CloudFormation 兼容性：此屬性類似於AWS::Events::Rule資源的Targets屬性。此屬性的 AWS SAM 版本只允許您指定單一目標的邏輯 ID。

範例

EventBridgeRule

以下是EventBridgeRule事件來源類型的範例。

YAML

```
EBRule:
  Type: EventBridgeRule
  Properties:
    Input: '{"Key": "Value"}'
    Pattern:
      detail:
        state:
          - terminated
    RetryPolicy:
      MaximumRetryAttempts: 5
      MaximumEventAgeInSeconds: 900
    DeadLetterConfig:
      Type: SQS
      QueueLogicalId: EBRuleDLQ
    Target:
      Id: MyTarget
```

DeadLetterConfig

用於指定 Amazon Simple Queue Service (Amazon SQS) 佇列的物件，在目標叫用失敗後 EventBridge 傳送事件。例如，當將事件傳送至不存在的 Lambda 函數，或權限不足以叫用 Lambda 函數時，呼叫可能會失敗。如需詳細資訊，請參閱 Amazon 使用 EventBridge 者指南中的事件重試政策和使用無效字母佇列。

> **Note**
>
> 資AWS::Serverless::Function源類型具有類似的資料類型，DeadLetterQueue可處理成功叫用目標 Lambda 函數之後發生的失敗。此類失敗的範例包括 Lambda 節流，或 Lambda 目標

函數傳回的錯誤。如需有關函數DeadLetterQueue屬性的詳細資訊，請參閱AWS Lambda 開發人員指南中的無效字母佇列。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
Arn: String
QueueLogicalId: String
Type: String
```

屬性

`Arn`

Amazon SQS 佇列的亞馬遜資源名稱 (ARN) 指定為無效字母佇列的目標。

> **Note**
>
> 指定Type性質或Arn性質，但不能同時指定兩者。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Events::RuleDeadLetterConfig數據類型的Arn屬性。

`QueueLogicalId`

如果指定，則 AWS SAM Type創建無效字母隊列的自定義名稱。

> **Note**
>
> 如果未設定Type屬性，則會忽略此屬性。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

Type

佇列的類型。設定此屬性時， AWS SAM 會自動建立無效字母佇列，並附加必要的以資源為基礎的原則，以授與規則資源的權限，以便將事件傳送至佇列。

> **Note**
>
> 指定Type性質或Arn性質，但不能同時指定兩者。

有效值：SQS

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

DeadLetterConfig

DeadLetterConfig

YAML

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

Target

配置觸發規則時 EventBridge 調用的 AWS 資源。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
  Id: String
```

屬性

Id

目標的邏輯識別碼。

的值Id可以包括英數字元、句號 (.)、連字號 (-) 和底線 (_)。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Events::RuleTarget數據類型的Id屬性。

範例

目標

YAML

```
EBRule:
  Type: EventBridgeRule
  Properties:
    Target:
      Id: MyTarget
```

HttpApi

描述具有類型之事件來源的物件 HttpApi。

如果 API 上存在指定路徑和方法的定 OpenApi 義，SAM 將為您新增 Lambda 整合和安全性區段 (如果適用)。

如果 API 上沒有 OpenApi 指定路徑和方法的定義，SAM 將為您創建此定義。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
ApiId: String
Auth: HttpApiFunctionAuth
Method: String
Path: String
PayloadFormatVersion: String
RouteSettings: RouteSettings
TimeoutInMillis: Integer
```

屬性

`ApiId`

此範本中定義之AWS::Serverless::HttpApi資源的識別碼。

如果未定義，則使用生成的文檔創建一個調`ServerlessHttpApi`用的默認AWS::Serverless::HttpApi資源，該 OpenApi 文檔包含由此模板中定義的 Api 事件定義的所有路徑和方法的聯合，這些文檔不指定`ApiId`.

這不能引用另一個模板中定義的AWS::Serverless::HttpApi資源。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Auth`

此特定 API + 路徑 + 方法的身份驗證配置。

用於在未指定的情況下覆蓋 API `DefaultAuthorizer` 或在單個路徑上設置 auth 配置非常`DefaultAuthorizer`有用。

類型:HttpApiFunctionAuth

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Method`

呼叫此函數的 HTTP 方法。

如果沒`Method`有指定`Path`和，SAM 將建立預設 API 路徑，將任何未對應至不同端點的要求路由至此 Lambda 函數。每個 API 只能存在其中一個預設路徑。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Path`

呼叫此函式的 Uri 路徑。必須以開頭`/`。

如果沒`Method`有指定`Path`和，SAM 將建立預設 API 路徑，將任何未對應至不同端點的要求路由至此 Lambda 函數。每個 API 只能存在其中一個預設路徑。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`PayloadFormatVersion`

為傳送至整合的承載指定格式。

注意： PayloadFormatVersion 要求 SAM 修改您的 OpenAPI 定義，因此它僅適用於在`DefinitionBody`屬性中 OpenApi 定義的內聯定義。

類型：字串

必要：否

預設值：2.0

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`RouteSettings`

此 HTTP API 的每個路由設定。如需有關路由設定的詳細資訊，請參閱 API Gateway 開發人員指南AWS::ApiGatewayV2::Stage RouteSettings中的。

附註：如果 RouteSettings 同時在 HttpApi 資源和事件來源中指定，則會將它們與優先順序的事件來源屬性 AWS SAM 合併。

類型:RouteSettings

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::ApiGatewayV2::Stage`資源的`RouteSettings`屬性。

`TimeoutInMillis`

自訂介於 50 和 29,000 毫秒之間的逾時。

注意： TimeoutInMillis 要求 SAM 修改您的 OpenAPI 定義，因此它僅適用於在`DefinitionBody`屬性中 OpenApi 定義的內聯定義。

類型：整數

必要：否

預設值：

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

預設 HttpApi 事件

HttpApi 使用預設路徑的事件。此 API 上所有未映射的路徑和方法都將路由到此端點。

YAML

```
Events:
  HttpApiEvent:
    Type: HttpApi
```

HttpApi

HttpApi 使用特定路徑和方法的事件。

YAML

```
Events:
  HttpApiEvent:
    Type: HttpApi
    Properties:
      Path: /
      Method: GET
```

HttpApi 授權

HttpApi 使用授權者的事件。

YAML

```
Events:
  HttpApiEvent:
    Type: HttpApi
    Properties:
      Path: /authenticated
      Method: GET
      Auth:
        Authorizer: OpenIdAuth
        AuthorizationScopes:
          - scope1
          - scope2
```

HttpApiFunctionAuth

在事件層級設定授權。

為特定的 API + 路徑 + 方法配置身份驗證

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
  AuthorizationScopes: List
```

```
  Authorizer: String
```

屬性

AuthorizationScopes

要套用至此 API、路徑和方法的授權範圍。

此處列出的範圍將覆蓋由（DefaultAuthorizer如果存在）套用的任何範圍。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

Authorizer

對Authorizer於特定功能。若要使用 IAM 授權AWS_IAM，請在範本的Globals區段EnableIamAuthorizer中指定並指定 true for。

如果您已在 API 上指定了全局授權者，並希望公開特定功能，請Authorizer將設置為NONE覆蓋。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

功能身份驗證

在功能級別指定授權

YAML

```
Auth:
  Authorizer: OpenIdAuth
```

```
  AuthorizationScopes:
    - scope1
    - scope2
```

IAM 授權

在事件層級指定 IAM 授權。若要在事件層級使用AWS_IAM授權，您還必須EnableIamAuthorizer在範本的Globals區段中指定 true for。如需詳細資訊，請參閱 模板的全局部分 AWS SAM。

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事件來源類型的物件。

建立AWS::IoT::TopicRule資源以宣告 AWS IoT 規則。如需詳細資訊，請參閱AWS CloudFormation 文

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
  AwsIotSqlVersion: String
  Sql: String
```

屬性

`AwsIotSqlVersion`

評估規則時所用的 SQL 規則引擎版本。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::IoT::TopicRule TopicRulePayload`資源的`AwsIotSqlVersion`屬性。

`Sql`

用於查詢主題的 SQL 陳述式。如需詳細資訊，請參閱AWS IoT 開發人員指南中的AWS IoT SQL 參考。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::IoT::TopicRule TopicRulePayload`資源的`Sql`屬性。

範例

物聯網規則

物聯網規則示例

YAML

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

Kinesis

描述Kinesis事件來源類型的物件。如需詳細資訊，請參閱AWS Lambda 開發人員指南中的 AWS Lambda 與 Amazon Kinesis 搭配使用。

AWS SAM 設置此事件類型時生成一個AWS::Lambda::EventSourceMapping資源。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
BatchSize: Integer
BisectBatchOnFunctionError: Boolean
DestinationConfig: DestinationConfig
Enabled: Boolean
FilterCriteria: FilterCriteria
FunctionResponseTypes: List
MaximumBatchingWindowInSeconds: Integer
MaximumRecordAgeInSeconds: Integer
MaximumRetryAttempts: Integer
ParallelizationFactor: Integer
StartingPosition: String
StartingPositionTimestamp: Double
Stream: String
TumblingWindowInSeconds: Integer
```

屬性

BatchSize

要在單一批次中擷取的最大項目數。

類型：整數

必要：否

預設值：100

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的BatchSize屬性。

下限：1

上限：10000

`BisectBatchOnFunctionError`

如果函數傳回錯誤，請將批次分成兩個，然後重試。

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::EventSourceMapping`資源的`BisectBatchOnFunctionError`屬性。

`DestinationConfig`

用於捨棄記錄的亞馬遜簡單佇列服務 (Amazon SQS) 佇列或亞馬遜 Simple Notification Service (Amazon SNS) 主題目的地。

類型:DestinationConfig

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::EventSourceMapping`資源的`DestinationConfig`屬性。

`Enabled`

停用事件來源映射以暫停輪詢和叫用。

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::EventSourceMapping`資源的`Enabled`屬性。

`FilterCriteria`

定義決定 Lambda 是否應該處理事件之準則的物件。如需詳細資訊，請參閱AWS Lambda 開發人員指南中的AWS Lambda 事件篩選。

類型:FilterCriteria

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的FilterCriteria屬性。

FunctionResponseTypes

目前套用至事件來源對應的回應類型清單。如需詳細資訊，請參閱《AWS Lambda 開發人員指南》中的報告批次項目失敗。

有效值：ReportBatchItemFailures

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的FunctionResponseTypes屬性。

MaximumBatchingWindowInSeconds

調用函式前收集記錄的最長時間 (單位為秒)。

類型：整數

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的MaximumBatchingWindowInSeconds屬性。

MaximumRecordAgeInSeconds

Lambda 傳送至函數進行處理的記錄保留時間上限。

類型：整數

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的MaximumRecordAgeInSeconds屬性。

MaximumRetryAttempts

當函數傳回錯誤時，重試的次數上限。

類型：整數

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的MaximumRetryAttempts屬性。

ParallelizationFactor

從每個碎片同時處理的批次數。

類型：整數

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的ParallelizationFactor屬性。

StartingPosition

要從中開始讀取的串流位置。

- AT_TIMESTAMP— 指定開始讀取記錄的時間。
- LATEST— 只讀新記錄。
- TRIM_HORIZON— 處理所有可用記錄。

有效值：AT_TIMESTAMP | LATEST | TRIM_HORIZON

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的StartingPosition屬性。

StartingPositionTimestamp

開始讀取的時間，以 Unix 時間秒為單位。定義StartingPositionTimestamp何時StartingPosition被指定為AT_TIMESTAMP。

類型：Double

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的StartingPositionTimestamp屬性。

Stream

資料串流或串流取用者的 Amazon 資源名稱 (ARN)。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的EventSourceArn屬性。

TumblingWindowInSeconds

處理視窗的持續時間 (以秒為單位)。有效範圍為 1 至 900 分鐘 (15 分鐘)。

如需詳細資訊，請參閱AWS Lambda 開發人員指南中的翻轉視窗。

類型：整數

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的TumblingWindowInSeconds屬性。

範例

Kinesis 事件來源

以下是 Kinesis 事件來源的範例。

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事件來源類型的物件。如需詳細資訊，請參閱AWS Lambda 開發人員指南中的將 Lambda 與 Amazon MQ 搭配使用。

AWS Serverless Application Model (AWS SAM) 在設定此事件類型時產生AWS::Lambda::EventSourceMapping資源。

> **Note**
>
> 若要將 Amazon MQ 佇列放在連線至公用網路中的 Lambda 函數的虛擬私有雲端 (VPC) 中，函數的執行角色必須包含下列許可：
>
> - `ec2:CreateNetworkInterface`
> - `ec2:DeleteNetworkInterface`
> - `ec2:DescribeNetworkInterfaces`
> - `ec2:DescribeSecurityGroups`
> - `ec2:DescribeSubnets`
> - `ec2:DescribeVpcs`
>
> 如需詳細資訊，請參閱AWS Lambda 開發人員指南中的執行角色權限。

語法

若要在 AWS SAM 範本中宣告此實體，請使用下列語法。

YAML

```
BatchSize: Integer
Broker: String
DynamicPolicyName: Boolean
Enabled: Boolean
FilterCriteria: FilterCriteria
MaximumBatchingWindowInSeconds: Integer
Queues: List
SecretsManagerKmsKeyId: String
SourceAccessConfigurations: List
```

屬性

`BatchSize`

要在單一批次中擷取的最大項目數。

類型：整數

必要：否

預設值：100

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::EventSourceMapping`資源的`BatchSize`屬性。

下限：1

上限：`10000`

`Broker`

Amazon MQ 代理程式的 Amazon Resource Name (ARN)。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::EventSourceMapping`資源的`EventSourceArn`屬性。

`DynamicPolicyName`

根據預設， AWS Identity and Access Management (IAM) 政策名稱是`SamAutoGeneratedAMQPolicy`為了向後相容。指定`true`使用自動產生的 IAM 政策名稱。此名稱將包含 Amazon MQ 事件來源邏輯 ID。

> **Note**
>
> 使用多個 Amazon MQ 事件來源時，請指定`true`以避免重複 IAM 政策名稱。

類型：布林值

必要：否

預設：`false`

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Enabled`

如果`true`，則事件來源對映處於作用中狀態。若要暫停輪詢和呼叫，請將設定為`false`。

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::EventSourceMapping`資源的`Enabled`屬性。

`FilterCriteria`

定義決定 Lambda 是否應該處理事件之準則的物件。如需詳細資訊，請參閱AWS Lambda 開發人員指南中的AWS Lambda 事件篩選。

類型:FilterCriteria

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::EventSourceMapping`資源的`FilterCriteria`屬性。

`MaximumBatchingWindowInSeconds`

調用函式前收集記錄的最長時間 (單位為秒)。

類型：整數

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::EventSourceMapping`資源的`MaximumBatchingWindowInSeconds`屬性。

`Queues`

要使用的 Amazon MQ 代理程式目的地佇列的名稱。

類型：列表

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::EventSourceMapping`資源的`Queues`屬性。

`SecretsManagerKmsKeyId`

客戶管理金鑰的 AWS Key Management Service (AWS KMS) 來源金鑰識別碼 AWS Secrets Manager。當您使用來自秘密管理員的客戶管理金鑰與未包含`kms:Decrypt`權限的 Lambda 執行角色時，此選項為必要條件。

此屬性的值是 UUID。例如：`1abc23d4-567f-8ab9-cde0-1fab234c5d67`。

類型：字串

必要：有條件

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`SourceAccessConfigurations`

驗證通訊協定或重要主機的陣列。使用資SourceAccessConfigurations料類型指定此項目。

對於`MQ`事件來源類型，唯一有效的組態類型為`BASIC_AUTH`和`VIRTUAL_HOST`。

- **`BASIC_AUTH`**— 存儲您的經紀人憑據的 Secrets Manager 秘密. 對於此類型，認證必須採用下列格式：`{"username": "your-username", "password": "your-password"}`。只允許一個類型`BASIC_AUTH`的對象。
- **`VIRTUAL_HOST`**— 您的 RabbitMQ 代理程式中虛擬主機的名稱。Lambda 將使用這個兔子 MQ 的主機作為事件源。只允許一個類型`VIRTUAL_HOST`的對象。

類型：列表

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::EventSourceMapping`資源的SourceAccessConfigurations屬性。

範例

Amazon MQ 事件來源

以下是 Amazon MQ 代理程式的`MQ`事件來源類型範例。

YAML

```
Events:
```

```
  MQEvent:
    Type: MQ
    Properties:
      Broker: arn:aws:mq:us-
east-2:123456789012:broker:MyBroker:b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9
      Queues: List of queues
      SourceAccessConfigurations:
        - Type: BASIC_AUTH
          URI: arn:aws:secretsmanager:us-east-1:01234567890:secret:MyBrokerSecretName
      BatchSize: 200
      Enabled: true
```

MSK

描述MSK事件來源類型的物件。如需詳細資訊，請參閱 AWS Lambda 開AWS Lambda 發人員指南中的搭配 Amazon MSK 使用。

AWS Serverless Application Model (AWS SAM) 在設定此事件類型時產生AWS::Lambda::EventSourceMapping資源。

語法

若要在 AWS SAM 範本中宣告此實體，請使用下列語法。

YAML

```
  ConsumerGroupId: String
  DestinationConfig: DestinationConfig
  FilterCriteria: FilterCriteria
  MaximumBatchingWindowInSeconds: Integer
  SourceAccessConfigurations: SourceAccessConfigurations
  StartingPosition: String
  StartingPositionTimestamp: Double
  Stream: String
  Topics: List
```

屬性

ConsumerGroupId

設定如何從卡夫卡主題讀取事件的字串。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的AmazonManagedKafkaConfiguration屬性。

DestinationConfig

組態物件，指定在 Lambda 處理過後事件的目標。

使用此屬性可從 Amazon MSK 事件來源指定失敗呼叫的目的地。

類型:DestinationConfig

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的 DestinationConfig屬性。

FilterCriteria

定義決定 Lambda 是否應該處理事件之準則的物件。如需詳細資訊，請參閱AWS Lambda 開發人員指南中的AWS Lambda 事件篩選。

類型:FilterCriteria

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的FilterCriteria屬性。

MaximumBatchingWindowInSeconds

調用函式前收集記錄的最長時間 (單位為秒)。

類型：整數

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的MaximumBatchingWindowInSeconds屬性。

SourceAccessConfigurations

保護和定義事件來源的身分驗證協定、VPC 元件或虛擬主機。

有效值：CLIENT_CERTIFICATE_TLS_AUTH

類型：SourceAccessConfiguration 的清單

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的SourceAccessConfigurations屬性。

StartingPosition

要從中開始讀取的串流位置。

- AT_TIMESTAMP— 指定開始讀取記錄的時間。
- LATEST— 只讀新記錄。
- TRIM_HORIZON— 處理所有可用記錄。

有效值：AT_TIMESTAMP | LATEST | TRIM_HORIZON

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的StartingPosition屬性。

StartingPositionTimestamp

開始讀取的時間，以 Unix 時間秒為單位。定義StartingPositionTimestamp何時StartingPosition被指定為AT_TIMESTAMP。

類型：Double

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的StartingPositionTimestamp屬性。

Stream

資料串流或串流取用者的 Amazon 資源名稱 (ARN)。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的EventSourceArn屬性。

Topics

Kafka 主題名稱。

類型：清單

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的Topics屬性。

範例

現有叢集的 Amazon MSK 範例

以下是已存在於. MSK AWS 帳戶

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
```

在相同範本中宣告叢集的 Amazon MSK 範例

以下是在相同範本檔案中宣告之 Amazon MSK 叢集的MSK事件來源類型範例。

YAML

```
Events:
  MSKEvent:
    Type: MSK
```

```
    Properties:
      StartingPosition: LATEST
      Stream:
        Ref: MyMskCluster   # This must be the name of an MSK cluster declared in the
same template file
      Topics:
        - MyTopic
```

S3

描述S3事件來源類型的物件。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
Bucket: String
Events: String | List
Filter: NotificationFilter
```

屬性

`Bucket`

S3 儲存貯體名稱。此值區必須存在於相同的範本中。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性類似於`AWS::S3::Bucket`資源的`BucketName`屬性。這是 SAM 中的必填欄位。此欄位僅接受在此範本中建立之 S3 儲存貯體的參考

`Events`

要叫用 Lambda 函數的 Amazon S3 儲存貯體事件。如需有效值的清單，請參閱 Amazon S3 支援的事件類型。

類型：字符串 | 列表

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::S3::BucketLambdaConfiguration數據類型的Event屬性。

Filter

決定哪些 Amazon S3 物件叫用 Lambda 函數的篩選規則。如需 Amazon S3 金鑰名稱篩選的相關資訊，請參閱 Amazon 簡單儲存服務使用者指南中的設定 Amazon S3 事件通知。

類型:NotificationFilter

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::S3::BucketLambdaConfiguration數據類型的Filter屬性。

範例

中三-活動

S3 事件的範例。

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事件來源類型的物件，可將您的無伺服器函數設定為按排程觸發之 Amazon EventBridge 規則的目標。有關更多信息，請參閱什麼是 Amazon EventBridge？ 在 Amazon 用 EventBridge 戶指南。

AWS Serverless Application Model(AWS SAM) 在設定此事件類型時產生AWS::Events::Rule資源。

> **Note**
>
> EventBridge 現在提供了一個新的排程功能，Amazon EventBridge Scheduler。Amazon EventBridge Scheduler 是無伺服器排程器，可讓您從單一中央受管服務建立、執行和管理任務。 EventBridge Scheduler具有高度可自訂性，並提供比 EventBridge 排程規則更高的可擴展性，並具有更廣泛的目標 API 操作和AWS 服務.
> 我們建議您使用 EventBridge Scheduler來呼叫排程的目標。若要在AWS SAM範本中定義此事件來源類型，請參閱ScheduleV2。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
DeadLetterConfig: DeadLetterConfig
Description: String
Enabled: Boolean
Input: String
Name: String
RetryPolicy: RetryPolicy
Schedule: String
State: String
```

屬性

`DeadLetterConfig`

設定 Amazon Simple Queue Service (Amazon SQS) 佇列，在目標叫用失敗後 EventBridge 傳送事件。例如，當將事件傳送至不存在的 Lambda 函數，或是沒 EventBridge 有足夠的權限無法呼叫 Lambda 函數時，呼叫可能會失敗。如需詳細資訊，請參閱 Amazon 使用 EventBridge 者指南中的事件重試政策和使用無效字母佇列。

> **Note**
>
> 資AWS::Serverless::Function源類型具有類似的資料類型`DeadLetterQueue`，可處理成功叫用目標 Lambda 函數之後發生的失敗。這些失敗類型的範例包括 Lambda 節流，或

Lambda 目標函數傳回的錯誤。如需有關函數DeadLetterQueue屬性的詳細資訊，請參閱AWS Lambda開發人員指南中的無效字母佇列。

類型:DeadLetterConfig

必要：否

AWS CloudFormation兼容性：此屬性類似於AWS::Events::RuleTarget數據類型的DeadLetterConfig屬性。如果您想要AWS SAM為您建立無效字母佇列，則此屬性的AWS SAM版本包含其他子屬性。

`Description`

規則的描述。

類型：字串

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給AWS::Events::Rule資源的Description屬性。

`Enabled`

指出系統是否已啟用規則。

若要停用規則，請將此屬性設定為false。

> **Note**
>
> 指定Enabled或State性質，但不能同時指定兩者。

類型：布林值

必要：否

AWS CloudFormation兼容性：此屬性類似於AWS::Events::Rule資源的State屬性。如果此屬性設定為，true則AWS SAM傳遞ENABLED，否則會傳遞DISABLED。

`Input`

傳遞到目標的有效 JSON 文字。如果您使用此屬性，事件文字本身不會有任何內容傳遞到目標。

類型：字串

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給AWS::Events::Rule Target資源的Input屬性。

Name

規則的名稱。如果您未指定名稱，AWS CloudFormation 會產生唯一的實體 ID，並使用該 ID 做為規則名稱。

類型：字串

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給AWS::Events::Rule資源的Name屬性。

RetryPolicy

包含重試政策設定相關資訊的 RetryPolicy 物件。如需詳細資訊，請參閱 Amazon 使用 EventBridge 者指南中的事件重試政策和使用無效字母佇列。

類型:RetryPolicy

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給AWS::Events::RuleTarget數據類型的RetryPolicy屬性。

Schedule

判斷何時及執行規則頻率的排程表達式。如需詳細資訊，請參閱規則的排程運算式。

類型：字串

必要：是

AWS CloudFormation兼容性：此屬性直接傳遞給AWS::Events::Rule資源的ScheduleExpression屬性。

State

規則的狀態。

接受的值：`DISABLED | ENABLED`

> **Note**
>
> 指定`Enabled`或`State`性質，但不能同時指定兩者。

類型：字串

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給`AWS::Events::Rule`資源的`State`屬性。

範例

CloudWatch 排程活動

CloudWatch 排程事件範例

YAML

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

DeadLetterConfig

用於指定 Amazon Simple Queue Service (Amazon SQS) 佇列的物件，在目標叫用失敗後 EventBridge 傳送事件。例如，當將事件傳送至不存在的 Lambda 函數，或權限不足以叫用 Lambda 函數時，呼叫可能會失敗。如需詳細資訊，請參閱 Amazon 使用 EventBridge 者指南中的事件重試政策和使用無效字母佇列。

> **Note**
>
> 資AWS::Serverless::Function源類型具有類似的資料類型，`DeadLetterQueue`可處理成功叫用目標 Lambda 函數之後發生的失敗。此類失敗的範例包括 Lambda 節流，或 Lambda 目標

函數傳回的錯誤。如需有關函數DeadLetterQueue屬性的詳細資訊，請參閱AWS Lambda開發人員指南中的無效字母佇列。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
Arn: String
QueueLogicalId: String
Type: String
```

屬性

`Arn`

Amazon SQS 佇列的亞馬遜資源名稱 (ARN) 指定為無效字母佇列的目標。

> **Note**
>
> 指定Type性質或Arn性質，但不能同時指定兩者。

類型：字串

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給AWS::Events::RuleDeadLetterConfig數據類型的Arn屬性。

`QueueLogicalId`

如果指定，則AWS SAMType創建無效字母隊列的自定義名稱。

> **Note**
>
> 如果未設定Type屬性，則會忽略此屬性。

類型：字串

必要：否

AWS CloudFormation兼容性：此屬性是唯一的，AWS SAM並且沒有相AWS CloudFormation等的屬性。

Type

佇列的類型。設定此屬性時，AWS SAM會自動建立無效字母佇列，並附加必要的資源型政策，以授與規則資源的權限，以便將事件傳送至佇列。

> **Note**
>
> 指定Type性質或Arn性質，但不能同時指定兩者。

有效值：SQS

類型：字串

必要：否

AWS CloudFormation兼容性：此屬性是唯一的，AWS SAM並且沒有相AWS CloudFormation等的屬性。

範例

DeadLetterConfig

DeadLetterConfig

YAML

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

ScheduleV2

描述ScheduleV2事件來源類型的物件，可將您的無伺服器函數設定為按排程觸發之 Amazon EventBridge Scheduler 事件的目標。如需詳細資訊，請參閱什麼是 Amazon EventBridge 排程器？ 在「EventBridge 排程器使用指南」中。

AWS Serverless Application Model(AWS SAM) 在設定此事件類型時產生AWS::Scheduler::Schedule資源。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
DeadLetterConfig: DeadLetterConfig
Description: String
EndDate: String
FlexibleTimeWindow: FlexibleTimeWindow
GroupName: String
Input: String
KmsKeyArn: String
Name: String
OmitName: Boolean
PermissionsBoundary: String
RetryPolicy: RetryPolicy
RoleArn: String
ScheduleExpression: String
ScheduleExpressionTimezone: String
StartDate: String
State: String
```

屬性

`DeadLetterConfig`

設定 Amazon Simple Queue Service (Amazon SQS) 佇列，在目標叫用失敗後 EventBridge 傳送事件。例如，當將事件傳送至不存在的 Lambda 函數，或是沒 EventBridge 有足夠的權限無法呼叫 Lambda 函數時，呼叫可能會失敗。如需詳細資訊，請參閱《排程器使用指EventBridge 南》中的 < 設定 EventBridge 排程器的無效字母佇列 > 。

> **Note**
>
> 資AWS::Serverless::Function源類型具有類似的資料類型`DeadLetterQueue`，可處理成功叫用目標 Lambda 函數之後發生的失敗。這些失敗類型的範例包括 Lambda 節流，或 Lambda 目標函數傳回的錯誤。如需有關函數`DeadLetterQueue`屬性的詳細資訊，請參閱AWS Lambda開發人員指南中的無效字母佇列。

類型:DeadLetterConfig

必要：否

AWS CloudFormation兼容性：此屬性類似於AWS::Scheduler::ScheduleTarget數據類型的DeadLetterConfig屬性。如果您想要AWS SAM為您建立無效字母佇列，則此屬性的AWS SAM版本包含其他子屬性。

`Description`

排程的描述。

類型：字串

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給AWS::Scheduler::Schedule資源的Description屬性。

`EndDate`

UTC 日期，排程可在此日期之前叫用其目標。視排程的週期運算式而定，叫用可能會在您指定的 EndDate 當天或之前停止。

類型：字串

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給AWS::Scheduler::Schedule資源的EndDate屬性。

`FlexibleTimeWindow`

允許設定可在其中呼叫排程的視窗。

類型:FlexibleTimeWindow

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給AWS::Scheduler::Schedule資源的FlexibleTimeWindow屬性。

`GroupName`

要與此排程產生關聯的排程群組名稱。如果未定義，則使用預設群組。

類型：字串

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給AWS::Scheduler::Schedule資源的GroupName屬性。

Input

傳遞到目標的有效 JSON 文字。如果您使用此屬性，事件文字本身不會有任何內容傳遞到目標。

類型：字串

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給AWS::Scheduler::Schedule Target資源的Input屬性。

KmsKeyArn

將用於加密客戶資料的 KMS 金鑰的 ARN。

類型：字串

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給AWS::Scheduler::Schedule資源的KmsKeyArn屬性。

Name

排程的名稱。如果未指定名稱，則會以格式AWS SAM產生名稱，*Function-Logical-IDEvent-Source-Name*並使用該 ID 做為排程名稱。

類型：字串

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給AWS::Scheduler::Schedule資源的Name屬性。

OmitName

依預設，AWS SAM會產生並使用格式為 *< event-source-name >* <Function-logical-ID>的排程名稱。將此性質設定true為以AWS CloudFormation產生唯一的實體 ID，並將其用作明細表名稱。

類型：布林值

必要：否

預設：`false`

AWS CloudFormation兼容性：此屬性是唯一的，AWS SAM並且沒有相AWS CloudFormation等的屬性。

`PermissionsBoundary`

用來設定角色許可邊界的政策 ARN。

> **Note**
>
> 如果`PermissionsBoundary`已定義，則AWS SAM會將相同的界限套用至排程器排程的目標 IAM 角色。

類型：字串

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給`AWS::IAM::Role`資源的`PermissionsBoundary`屬性。

`RetryPolicy`

包含重試政策設定相關資訊的 RetryPolicy 物件。

類型:RetryPolicy

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給`AWS::Scheduler::ScheduleTarget`數據類型的`RetryPolicy`屬性。

`RoleArn`

呼叫排程時， EventBridge 排程器將用於目標的 IAM 角色 ARN。

類型:RoleArn

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給`AWS::Scheduler::ScheduleTarget`數據類型的`RoleArn`屬性。

`ScheduleExpression`

決定排程器排程事件執行的時間和頻率的排程運算式。

類型：字串

必要：是

AWS CloudFormation兼容性：此屬性直接傳遞給`AWS::Scheduler::Schedule`資源的`ScheduleExpression`屬性。

`ScheduleExpressionTimezone`

計算排程運算式所使用的時區。

類型：字串

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給`AWS::Scheduler::Schedule`資源的`ScheduleExpressionTimezone`屬性。

`StartDate`

日期 (以 UTC 為單位)，在此日期之後，排程就可以開始叫用目標。視排程的週期運算式而定，叫用可能會在您指定的 StartDate 當天或之後發生。

類型：字串

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給`AWS::Scheduler::Schedule`資源的`StartDate`屬性。

`State`

排程器排程的狀態。

接受的值：`DISABLED | ENABLED`

類型：字串

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給`AWS::Scheduler::Schedule`資源的`State`屬性。

範例

定義排程 2 資源的基本範例

```
Resources:
  Function:
    Properties:
      ...
      Events:
        ScheduleEvent:
          Type: ScheduleV2
          Properties:
            ScheduleExpression: "rate(1 minute)"
        ComplexScheduleEvent:
          Type: ScheduleV2
          Properties:
            ScheduleExpression: rate(1 minute)
            FlexibleTimeWindow:
              Mode: FLEXIBLE
              MaximumWindowInMinutes: 5
            StartDate: '2022-12-28T12:00:00.000Z'
            EndDate: '2023-01-28T12:00:00.000Z'
            ScheduleExpressionTimezone: UTC
            RetryPolicy:
              MaximumRetryAttempts: 5
              MaximumEventAgeInSeconds: 300
            DeadLetterConfig:
              Type: SQS
```

SelfManagedKafka

描述`SelfManagedKafka`事件來源類型的物件。如需詳細資訊，請參閱開發人員指AWS Lambda 南中的AWS Lambda 與自我管理的 Apache Kafka 搭配使用。

AWS Serverless Application Model (AWS SAM) 在設定此事件類型時產生AWS::Lambda::EventSourceMapping資源。

語法

若要在 AWS SAM 範本中宣告此實體，請使用下列語法。

YAML

```
  BatchSize: Integer
```

```
  ConsumerGroupId: String
  DestinationConfig: DestinationConfig
  Enabled: Boolean
  FilterCriteria: FilterCriteria
  KafkaBootstrapServers: List
  SourceAccessConfigurations: SourceAccessConfigurations
  StartingPosition: String
  StartingPositionTimestamp: Double
  Topics: List
```

屬性

`BatchSize`

Lambda 從串流中提取並傳送至函數的每個批次中的最大記錄數。

類型：整數

必要：否

預設值：100

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::EventSourceMapping`資源的`BatchSize`屬性。

下限：1

上限：`10000`

`ConsumerGroupId`

設定如何從卡夫卡主題讀取事件的字串。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::EventSourceMapping`資源的`SelfManagedKafkaConfiguration`屬性。

`DestinationConfig`

組態物件，指定在 Lambda 處理過後事件的目標。

使用此屬性可從自我管理的 Kafka 事件來源指定失敗叫用的目的地。

類型:DestinationConfig

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的 DestinationConfig屬性。

Enabled

停用事件來源映射以暫停輪詢和叫用。

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的Enabled屬性。

FilterCriteria

定義決定 Lambda 是否應該處理事件之準則的物件。如需詳細資訊，請參閱AWS Lambda 開發人員指南中的AWS Lambda 事件篩選。

類型:FilterCriteria

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的FilterCriteria屬性。

KafkaBootstrapServers

您的卡夫卡經紀人的引導服務器列表。包括端口，例如 broker.example.com:*xxxx*

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

SourceAccessConfigurations

保護和定義事件來源的身分驗證協定、VPC 元件或虛擬主機。

有效值：BASIC_AUTH | CLIENT_CERTIFICATE_TLS_AUTH | SASL_SCRAM_256_AUTH | SASL_SCRAM_512_AUTH | SERVER_ROOT_CA_CERTIFICATE

類型：SourceAccessConfiguration 的清單

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的 SourceAccessConfigurations屬性。

StartingPosition

要從中開始讀取的串流位置。

- AT_TIMESTAMP— 指定開始讀取記錄的時間。
- LATEST— 只讀新記錄。
- TRIM_HORIZON— 處理所有可用記錄。

有效值：AT_TIMESTAMP | LATEST | TRIM_HORIZON

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的StartingPosition屬性。

StartingPositionTimestamp

開始讀取的時間，以 Unix 時間秒為單位。定義StartingPositionTimestamp何時StartingPosition被指定為AT_TIMESTAMP。

類型：Double

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的StartingPositionTimestamp屬性。

Topics

Kafka 主題名稱。

類型：清單

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的Topics屬性。

範例

自我管理的卡夫卡事件來源

以下是SelfManagedKafka事件來源類型的範例。

YAML

```
Events:
  SelfManagedKafkaEvent:
    Type: SelfManagedKafka
    Properties:
      BatchSize: 1000
      Enabled: true
      KafkaBootstrapServers:
        - abc.xyz.com:xxxx
      SourceAccessConfigurations:
        -  Type: BASIC_AUTH
           URI: arn:aws:secretsmanager:us-west-2:123456789012:secret:my-path/my-secret-
name-1a2b3c
      Topics:
        - MyKafkaTopic
```

SNS

描述SNS事件來源類型的物件。

SAM 在設置此事件類型時生成AWS::SNS::Subscription資源

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
  FilterPolicy: SnsFilterPolicy
  FilterPolicyScope: String
```

```
  RedrivePolicy: Json
  Region: String
  SqsSubscription: Boolean | SqsSubscriptionObject
  Topic: String
```

屬性

`FilterPolicy`

指派給訂閱的篩選條件政策 JSON。如需詳細資訊，請參閱 Amazon 簡單通知服務 API 參考GetSubscriptionAttributes中的。

類型:SnsFilterPolicy

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::SNS::Subscription`資源的`FilterPolicy`屬性。

`FilterPolicyScope`

此屬性可讓您使用下列其中一種字串值類型來選擇篩選範圍：

- `MessageAttributes`— 篩選器會套用至郵件屬性。
- `MessageBody`— 篩選器會套用至郵件內文。

類型：字串

必要：否

預設：`MessageAttributes`

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::SNS::Subscription`資源的`FilterPolicyScope`屬性。

`RedrivePolicy`

如果指定，則會將無法傳遞的訊息傳送到指定的 Amazon SQS 無效信件佇列。由於用戶端錯誤 (例如當訂閱的端點無法連線時) 或伺服器錯誤 (例如提供訂閱端點的服務無法使用) 而無法傳遞的訊息，會保留在無效信件佇列，以供進一步分析或重新處理。

如需重新磁碟原則和無效字母佇列的詳細資訊，請參閱 Amazon SQS 無效字母佇列 (英文) 中的 Amazon SQS 無效字母佇列 (英文)。

類型：Json

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::SNS::Subscription資源的RedrivePolicy屬性。

Region

針對跨區域訂閱，為主題所在的區域。

如果未指定區域，則 CloudFormation 使用呼叫者的區域作為預設值。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::SNS::Subscription資源的Region屬性。

SqsSubscription

將此屬性設定為 true，或指定啟SqsSubscriptionObject用 SQS 佇列中的批次處理 SNS 主題通知。將此屬性設定為true建立新的 SQS 佇列，而指定會SqsSubscriptionObject使用現有的 SQS 佇列。

類型：布爾 | SqsSubscriptionObject

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

Topic

要訂閱的主題 ARN。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::SNS::Subscription資源的TopicArn屬性。

範例

SNS 事件來源範例

SNS 事件來源範例

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

指定 SNS 事件的現有 SQS 佇列選項

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
  BatchSize: String
  Enabled: Boolean
  QueueArn: String
  QueuePolicyLogicalId: String
  QueueUrl: String
```

屬性

`BatchSize`

SQS 佇列單一批次中要擷取的最大項目數。

類型：字串

必要：否

預設值：10

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Enabled`

停用 SQS 事件來源對映以暫停輪詢和呼叫。

類型：布林值

必要：否

預設值：真

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`QueueArn`

指定現有的 SQS 佇列範圍。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`QueuePolicyLogicalId`

為資源提供自定義邏輯 ID 名稱。AWS::SQS::QueuePolicy

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`QueueUrl`

指定與`QueueArn`屬性相關聯的佇列 URL。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

SNS 事件的現有 SQS

將用於子項目的現有 SQS 佇列新增至 SNS 主題的範例。

YAML

```
QueuePolicyLogicalId: CustomQueuePolicyLogicalId
QueueArn:
  Fn::GetAtt: MyCustomQueue.Arn
QueueUrl:
  Ref: MyCustomQueue
BatchSize: 5
```

SQS

描述SQS事件來源類型的物件。如需詳細資訊，請參閱 AWS Lambda 開AWS Lambda 發人員指南中的搭配 Amazon SQS 使用。

SAM 在設置此事件類型時生成AWS::Lambda::EventSourceMapping資源

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
  BatchSize: Integer
  Enabled: Boolean
```

```
  FilterCriteria: FilterCriteria
  FunctionResponseTypes: List
  MaximumBatchingWindowInSeconds: Integer
  Queue: String
  ScalingConfig: ScalingConfig
```

屬性

`BatchSize`

要在單一批次中擷取的最大項目數。

類型：整數

必要：否

預設值：10

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::EventSourceMapping`資源的`BatchSize`屬性。

下限：1

上限：`10000`

`Enabled`

停用事件來源映射以暫停輪詢和叫用。

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::EventSourceMapping`資源的`Enabled`屬性。

`FilterCriteria`

定義決定 Lambda 是否應該處理事件之準則的物件。如需詳細資訊，請參閱AWS Lambda 開發人員指南中的AWS Lambda 事件篩選。

類型:FilterCriteria

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的FilterCriteria屬性。

FunctionResponseTypes

目前套用至事件來源對應的回應類型清單。如需詳細資訊，請參閱AWS Lambda 開發人員指南中的報告批次項目失敗。

有效值：ReportBatchItemFailures

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的FunctionResponseTypes屬性。

MaximumBatchingWindowInSeconds

呼叫函式之前收集記錄的時間上限 (以秒為單位)。

類型：整數

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的MaximumBatchingWindowInSeconds屬性。

Queue

佇列的 ARN。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的EventSourceArn屬性。

ScalingConfig

調整 SQS 輪詢器的組態，以控制呼叫速率並設定最大並行呼叫。

Type (類型)：ScalingConfig

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::EventSourceMapping資源的 ScalingConfig屬性。

範例

基本 SQS 事件

```
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"]}'
```

設定 SQS 佇列的部分批次報告

```
Events:
  SQSEvent:
    Type: SQS
    Properties:
      Enabled: true
      FunctionResponseTypes:
        - ReportBatchItemFailures
      Queue: !GetAtt MySqsQueue.Arn
      BatchSize: 10
```

具有已設定擴展功能的 SQS 事件的 Lambda 函數

```
MyFunction:
  Type: AWS::Serverless::Function
  Properties:
    ...
    Events:
      MySQSEvent:
        Type: SQS
```

```
        Properties:
          ...
          ScalingConfig:
            MaximumConcurrency: 10
```

FunctionCode

Lambda 函數的部署套裝服務。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

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

屬性

`Bucket`

與您的功能位於相同 AWS 區域的 Amazon S3 儲存貯體。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::FunctionCode`數據類型的`S3Bucket`屬性。

`Key`

部署套件的 Amazon S3 金鑰。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::FunctionCode`數據類型的`S3Key`屬性。

`Version`

對於版本控制的物件，要使用的部署套件物件版本。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::FunctionCode`數據類型的`S3ObjectVersion`屬性。

範例

FunctionCode

`CodeUri`: 函數程式碼範例

YAML

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

FunctionUrlConfig

創建具有指定配置參數的 AWS Lambda 函數 URL。Lambda 函數網址是一個 HTTPS 端點，您可以使用它來叫用您的函數。

根據預設，您建立的函數 URL 會使用 Lambda 函數的`$LATEST`版本。如果您`AutoPublishAlias`為 Lambda 函數指定一個，則端點會連接到指定的函數別名。

如需詳細資訊，請參閱AWS Lambda 開發人員指南中的 Lambda 函數 URL。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
AuthType: String
Cors: Cors
```

```
InvokeMode: String
```

屬性

`AuthType`

函數 URL 的授權類型。若要使用 AWS Identity and Access Management (IAM) 授權請求，請將設定為`AWS_IAM`。對於開放取用，請設定為`NONE`。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::Url`資源的`AuthType`屬性。

`Cors`

函數 URL 的跨來源資源共享 (CORS) 設定。

類型：Cors

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::Url`資源的`Cors`屬性。

`InvokeMode`

您的函數 URL 將被調用的模式。要讓您的函數在調用完成後返回響應，請將設置為`BUFFERED`。若要讓您的函數串流回應，請將設定為`RESPONSE_STREAM`。預設值為 `BUFFERED`。

有效值：`BUFFERED` 或 `RESPONSE_STREAM`

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::Url`資源的`InvokeMode`屬性。

範例

功能網址

下列範例會建立含有函數 URL 的 Lambda 函數。函數 URL 使用 IAM 授權。

YAML

```
HelloWorldFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: hello_world/
    Handler: index.handler
    Runtime: nodejs20.x
    FunctionUrlConfig:
      AuthType: AWS_IAM
      InvokeMode: RESPONSE_STREAM

Outputs:
  MyFunctionUrlEndpoint:
      Description: "My Lambda Function URL Endpoint"
      Value:
        Fn::GetAtt: HelloWorldFunctionUrl.FunctionUrl
```

AWS::Serverless::GraphQLApi

使用 AWS Serverless Application Model (AWS SAM) `AWS::Serverless::GraphQLApi` 資源類型為無伺服器應用程式建立和設定 AWS AppSync GraphQL API。

若要深入瞭解 AWS AppSync，請參閱「什麼是 AWS AppSync？」 在AWS AppSync 開發人員指南中。

語法

YAML

```
LogicalId:
  Type: AWS::Serverless::GraphQLApi
  Properties:
    ApiKeys: ApiKeys
    Auth: Auth
    Cache: AWS::AppSync::ApiCache
    DataSources: DataSource
    DomainName: AWS::AppSync::DomainName
    Functions: Function
    Logging: LogConfig
    Name: String
    Resolvers: Resolver
    SchemaInline: String
```

```
    SchemaUri: String
    Tags:
    - Tag
    XrayEnabled: Boolean
```

屬性

`ApiKeys`

建立可用於執行需要 API 金鑰之GraphQL作業的唯一金鑰。

類型:ApiKeys

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Auth`

為您的 GraphQL API 配置身份驗證。

類型：驗證

必要：是

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Cache`

`CreateApiCache`作業的輸入。

類型:AWS::AppSync::ApiCache

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給資AWS::AppSync::ApiCache源。

`DataSources`

為中的函數建立資料來源 AWS AppSync 以進行連線。 AWS SAM 支援 Amazon DynamoDB 和 AWS Lambda 資料來源。

類型:DataSource

必要：是

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`DomainName`

GraphQLAPI 的自定義域名。

類型:AWS::AppSync::DomainName

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給資AWS::AppSync::DomainName源。 AWS SAM 會自動產生AWS::AppSync::DomainNameApiAssociation資源。

`Functions`

在 GraphQL API 中設定函數以執行特定作業。

類型：功能

必要：是

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Logging`

為您的 API 配置 Amazon CloudWatch 日誌記錄GraphQL。

如果您未指定此屬性， AWS SAM 將會產生`CloudWatchLogsRoleArn`並設定下列值：

- `ExcludeVerboseContent: true`
- `FieldLogLevel: ALL`

若要選擇退出記錄，請指定下列項目：

```
Logging: false
```

類型:LogConfig

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::GraphQLApi資源的LogConfig屬性。

LogicalId

GraphQLAPI 的唯一名稱。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::GraphQLApi資源的Name屬性。

Name

您的 GraphQL API 的名稱。指定此性質以取代LogicalId值。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::GraphQLApi資源的Name屬性。

Resolvers

設定 API 欄位的解析程式GraphQL。 AWS SAM 支援JavaScript管線解析器。

類型：解析器

必要：是

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

SchemaInline

SDL格式中GraphQL結構描述的文字表示。

類型：字串

必要：有條件限制。您必須指定SchemaInline或SchemaUri。

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::GraphQLSchema資源的Definition屬性。

`SchemaUri`

該模式的 Amazon Simple Storage Service (Amazon S3) 存儲桶 URI 或本地文件夾的路徑。

如果您指定本機資料夾的路徑，則 AWS CloudFormation 需要在部署之前先將檔案上傳到 Amazon S3。您可以使用 AWS SAMCLI來促進此程序。如需詳細資訊，請參閱 如何在部署時上傳本地文件 AWS SAMCLI。

類型：字串

必要：有條件限制。您必須指定`SchemaInline`或`SchemaUri`。

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::AppSync::GraphQLSchema`資源的`DefinitionS3Location`屬性。

`Tags`

此 GraphQL API 的標籤（鍵值對）。使用標籤來識別和分類資源。

類型：標籤的清單

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::AppSync::GraphQLApi`資源的`Tag`屬性。

`XrayEnabled`

指示是否對此資源使用 AWS X-Ray 追蹤。

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::AppSync::GraphQLApi`資源的`XrayEnabled`屬性。

範例

GraphQL API使用 DynamoDB 資料來源

在此範例中，我們建立了使用 DynamoDB 表作為資料來源的 GraphQL API。

模式

```
schema {
  query: Query
  mutation: Mutation
}

type Query {
  getPost(id: String!): Post
}

type Mutation {
  addPost(author: String!, title: String!, content: String!): Post!
}

type Post {
  id: String!
  author: String
  title: String
  content: String
  ups: Int!
  downs: Int!
  version: Int!
}
```

模板.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  DynamoDBPostsTable:
    Type: AWS::Serverless::SimpleTable

  MyGraphQLAPI:
    Type: AWS::Serverless::GraphQLApi
    Properties:
      SchemaUri: ./sam_graphql_api/schema.graphql
      Auth:
        Type: AWS_IAM
      DataSources:
        DynamoDb:
          PostsDataSource:
            TableName: !Ref DynamoDBPostsTable
            TableArn: !GetAtt DynamoDBPostsTable.Arn
```

```
      Functions:
        preprocessPostItem:
          Runtime:
            Name: APPSYNC_JS
            Version: 1.0.0
          DataSource: NONE
          CodeUri: ./sam_graphql_api/preprocessPostItem.js
        createPostItem:
          Runtime:
            Name: APPSYNC_JS
            Version: "1.0.0"
          DataSource: PostsDataSource
          CodeUri: ./sam_graphql_api/createPostItem.js
        getPostFromTable:
          Runtime:
            Name: APPSYNC_JS
            Version: "1.0.0"
          DataSource: PostsDataSource
          CodeUri: ./sam_graphql_api/getPostFromTable.js
      Resolvers:
        Mutation:
          addPost:
            Runtime:
              Name: APPSYNC_JS
              Version: "1.0.0"
            Pipeline:
            - preprocessPostItem
            - createPostItem
        Query:
          getPost:
            CodeUri: ./sam_graphql_api/getPost.js
            Runtime:
              Name: APPSYNC_JS
              Version: "1.0.0"
            Pipeline:
            - getPostFromTable
```

createPostItem.js

```
import { util } from "@aws-appsync/utils";

export function request(ctx) {
  const { key, values } = ctx.prev.result;
```

```
  return {
    operation: "PutItem",
    key: util.dynamodb.toMapValues(key),
    attributeValues: util.dynamodb.toMapValues(values),
  };
}

export function response(ctx) {
  return ctx.result;
}
```

getPostFromTable.js

```
import { util } from "@aws-appsync/utils";

export function request(ctx) {
  return dynamoDBGetItemRequest({ id: ctx.args.id });
}

export function response(ctx) {
  return ctx.result;
}

/**
 * A helper function to get a DynamoDB item
 */
function dynamoDBGetItemRequest(key) {
  return {
    operation: "GetItem",
    key: util.dynamodb.toMapValues(key),
  };
}
```

preprocessPostItem.js

```
import { util } from "@aws-appsync/utils";

export function request(ctx) {
  const id = util.autoId();
  const { ...values } = ctx.args;
  values.ups = 1;
  values.downs = 0;
  values.version = 1;
```

```
  return { payload: { key: { id }, values: values } };
}

export function response(ctx) {
  return ctx.result;
}
```

這是我們的解析器代碼：

getPost.js

```
export function request(ctx) {
  return {};
}

export function response(ctx) {
  return ctx.prev.result;
}
```

GraphQL以 Lambda 函數做為資料來源的 API

在此範例中，我們建立了使用 Lambda 函數做為資料來源的 GraphQL API。

模板.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyLambdaFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.handler
      Runtime: nodejs20.x
      CodeUri: ./lambda

  MyGraphQLAPI:
    Type: AWS::Serverless::GraphQLApi
    Properties:
      Name: MyApi
      SchemaUri: ./gql/schema.gql
      Auth:
        Type: API_KEY
```

```
      ApiKeys:
        MyApiKey:
          Description: my api key
      DataSources:
        Lambda:
          MyLambdaDataSource:
            FunctionArn: !GetAtt MyLambdaFunction.Arn
      Functions:
        lambdaInvoker:
          Runtime:
            Name: APPSYNC_JS
            Version: 1.0.0
          DataSource: MyLambdaDataSource
          CodeUri: ./gql/invoker.js
      Resolvers:
        Mutation:
          addPost:
            Runtime:
              Name: APPSYNC_JS
              Version: 1.0.0
            Pipeline:
            - lambdaInvoker
        Query:
          getPost:
            Runtime:
              Name: APPSYNC_JS
              Version: 1.0.0
            Pipeline:
            - lambdaInvoker

Outputs:
  MyGraphQLAPI:
    Description: AppSync API
    Value: !GetAtt MyGraphQLAPI.GraphQLUrl
  MyGraphQLAPIMyApiKey:
    Description: API Key for authentication
    Value: !GetAtt MyGraphQLAPIMyApiKey.ApiKey
```

模式

```
schema {
  query: Query
  mutation: Mutation
```

```
}
type Query {
  getPost(id: ID!): Post
}
type Mutation {
  addPost(id: ID!, author: String!, title: String, content: String): Post!
}
type Post {
  id: ID!
  author: String!
  title: String
  content: String
  ups: Int
  downs: Int
}
```

以下是我們的功能：

lambda/index.js

```
exports.handler = async (event) => {
  console.log("Received event {}", JSON.stringify(event, 3));

  const posts = {
    1: {
      id: "1",
      title: "First book",
      author: "Author1",
      content: "Book 1 has this content",
      ups: "100",
      downs: "10",
    },
  };

  console.log("Got an Invoke Request.");
  let result;
  switch (event.field) {
    case "getPost":
      return posts[event.arguments.id];
    case "addPost":
      // return the arguments back
      return event.arguments;
    default:
      throw new Error("Unknown field, unable to resolve " + event.field);
```

```
  }
};
```

invoker.js

```
import { util } from "@aws-appsync/utils";

export function request(ctx) {
  const { source, args } = ctx;
  return {
    operation: "Invoke",
    payload: { field: ctx.info.fieldName, arguments: args, source },
  };
}

export function response(ctx) {
  return ctx.result;
}
```

ApiKeys

建立可用於執行需要 API 金鑰之GraphQL作業的唯一金鑰。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
LogicalId:
  ApiKeyId: String
  Description: String
  ExpiresOn: Double
```

屬性

`ApiKeyId`

API 金鑰的唯一名稱。指定以取代`LogicalId`值。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::ApiKey資源的ApiKeyId屬性。

`Description`

您的 API 密鑰的描述。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::ApiKey資源的Description屬性。

`ExpiresOn`

API 金鑰的到期時間。日期以自 epoch 以來的秒數表示，四捨五入至最接近的整點。

類型：Double

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::ApiKey資源的Expires屬性。

`LogicalId`

API 金鑰的唯一名稱。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::ApiKey資源的ApiKeyId屬性。

Auth

為您的 GraphQL API 設定授權。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
Additional:
- AuthProvider
LambdaAuthorizer: LambdaAuthorizerConfig
OpenIDConnect: OpenIDConnectConfig
Type: String
UserPool: UserPoolConfig
```

屬性

`Additional`

GraphQLAPI 的其他授權類型清單。

類型：清單 AuthProvider

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`LambdaAuthorizer`

為您的 Lambda 函數授權者指定選用的授權組態。當指定為時`Type`，您可以配置此可選屬性`AWS_LAMBDA`。

類型:LambdaAuthorizerConfig

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::AppSync::GraphQLApi`資源的`LambdaAuthorizerConfig`屬性。

`OpenIDConnect`

指定OpenID Connect合規服務的選用授權組態。當指定為時`Type`，您可以配置此可選屬性`OPENID_CONNECT`。

類型：OpenID ConnectConfig

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::AppSync::GraphQLApi`資源的`OpenIDConnectConfig`屬性。

`Type`

應用程式和 AWS AppSync GraphQL API 之間的預設授權類型。

如需允許值的清單和說明，請參閱AWS AppSync 開發人員指南中的授權和驗證。

當您指定 Lambda 授權者 (`AWS_LAMBDA`) 時， AWS SAM 會建立 AWS Identity and Access Management (IAM) 政策以在 GraphQL API 和 Lambda 函數之間佈建許可。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::AppSync::GraphQLApi`資源的`AuthenticationType`屬性。

`UserPool`

指定使用 Amazon Cognito 使用者集區的選用授權組態。當指定為時`Type`，您可以配置此可選屬性`AMAZON_COGNITO_USER_POOLS`。

類型:UserPoolConfig

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::AppSync::GraphQLApi`資源的`UserPoolConfig`屬性。

範例

設定預設和其他授權類型

在此範例中，我們首先將 Lambda 授權者設定為 GraphQL API 的預設授權類型。

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyGraphQLAPI:
    Type: AWS::Serverless::GraphQLApi
    Properties:
      Auth:
```

```
        Type: AWS_LAMBDA
        LambdaAuthorizer:
          AuthorizerUri: !GetAtt Authorizer1.Arn
          AuthorizerResultTtlInSeconds: 10
          IdentityValidationExpression: hello
```

接下來，我們通過將以下內容添加到 AWS SAM 模板中來為 GraphQL API 配置其他授權類型：

```
        Additional:
        - Type: AWS_IAM
        - Type: API_KEY
        - Type: OPENID_CONNECT
          OpenIDConnect:
            AuthTTL: 10
            ClientId: myId
            IatTTL: 10
            Issuer: prod
```

這會產生下列 AWS SAM 範本：

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyGraphQLAPI:
    Type: AWS::Serverless::GraphQLApi
    Properties:
      Auth:
        Type: AWS_LAMBDA
        LambdaAuthorizer:
          AuthorizerUri: !GetAtt Authorizer1.Arn
          AuthorizerResultTtlInSeconds: 10
          IdentityValidationExpression: hello
        Additional:
        - Type: AWS_IAM
        - Type: API_KEY
        - Type: OPENID_CONNECT
          OpenIDConnect:
            AuthTTL: 10
            ClientId: myId
            IatTTL: 10
            Issuer: prod
```

AuthProvider

其他 GraphQL API 授權類型的可選授權配置。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
LambdaAuthorizer: LambdaAuthorizerConfig
OpenIDConnect: OpenIDConnectConfig
Type: String
UserPool: UserPoolConfig
```

屬性

`LambdaAuthorizer`

為您的 AWS Lambda 函數授權者指定選用的授權組態。當指定為時`Type`，您可以配置此可選屬性`AWS_LAMBDA`。

類型: LambdaAuthorizerConfig

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::AppSync::GraphQLApi` `AdditionalAuthenticationProvider`對象的 `LambdaAuthorizerConfig`屬性。

`OpenIDConnect`

指定OpenID Connect合規服務的選用授權組態。當指定為時`Type`，您可以配置此可選屬性`OPENID_CONNECT`。

類型：OpenID ConnectConfig

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::AppSync::GraphQLApi` `AdditionalAuthenticationProvider`對象的 `OpenIDConnectConfig`屬性。

`Type`

應用程式和 AWS AppSync GraphQL API 之間的預設授權類型。

如需允許值的清單和說明，請參閱AWS AppSync 開發人員指南中的授權和驗證。

當您指定 Lambda 授權者 (AWS_LAMBDA) 時， AWS SAM 會建立 AWS Identity and Access Management (IAM) 政策以在 GraphQL API 和 Lambda 函數之間佈建許可。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::GraphQLApi AdditionalAuthenticationProvider對象的 AuthenticationType屬性。

UserPool

指定使用 Amazon Cognito 使用者集區的選用授權組態。當指定為時Type，您可以配置此可選屬性AMAZON_COGNITO_USER_POOLS。

類型: UserPoolConfig

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::GraphQLApi AdditionalAuthenticationProvider對象的 UserPoolConfig屬性。

DataSource

設定 GraphQL API 解析器可連線的資料來源。您可以使用 AWS Serverless Application Model (AWS SAM) 範本來設定與下列資料來源的連線：

- Amazon DynamoDB
- AWS Lambda

若要深入瞭解資料來源，請參閱AWS AppSync 開發人員指南中的附加資料來源。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
DynamoDb: DynamoDb
```

```
Lambda: Lambda
```

屬性

DynamoDb

將 DynamoDB 表格設定為您的 GraphQL API 解析器的資料來源。

類型:DynamoDb

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

Lambda

將 Lambda 函數設定為 GraphQL API 解析器的資料來源。

類型：Lambda

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

DynamoDb

將 Amazon DynamoDB 表格設定為您的 GraphQL API 解析器的資料來源。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
LogicalId:
  DeltaSync: DeltaSyncConfig
  Description: String
  Name: String
  Permissions: List
  Region: String
  ServiceRoleArn: String
```

```
  TableArn: String
  TableName: String
  UseCallerCredentials: Boolean
  Versioned: Boolean
```

屬性

`DeltaSync`

描述差異同步組態。

類型:DeltaSyncConfig

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::AppSync::DataSource` `DynamoDBConfig`對象的`DeltaSyncConfig`屬性。

`Description`

資料來源的說明。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::AppSync::DataSource`資源的`Description`屬性。

`LogicalId`

資料來源的唯一名稱。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::AppSync::DataSource`資源的`Name`屬性。

`Name`

資料來源的名稱。指定此性質以取代`LogicalId`值。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::DataSource資源的Name屬性。

`Permissions`

使用將權限佈建到資料來源AWS SAM 連接器。您可以在清單中提供下列任何值：

- `Read`— 允許您的解析器讀取您的數據源。
- `Write`— 允許您的解析器寫入數據源。

AWS SAM 使用在部署時轉換的AWS::Serverless::Connector資源來佈建您的權限。若要瞭解產生的資源，請參閱AWS CloudFormation指定時產生的資源 AWS::Serverless::Connector。

> **Note**
>
> 您可以指定 Permissions 或 ServiceRoleArn，但不能同時指定兩者。如果兩者都未指定， AWS SAM 將會產生Read和的預設值Write。若要撤銷對資料來源的存取權，請從範本中移除 DynamoDB 物件。 AWS SAM

類型：列表

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。它類似於資AWS::Serverless::Connector源的Permissions屬性。

`Region`

您 AWS 區域 的 DynamoDB 料表。如果未指定，請 AWS SAM 使用AWS::Region。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::DataSource DynamoDBConfig對象的AwsRegion屬性。

`ServiceRoleArn`

資料來源的 AWS Identity and Access Management (IAM) 服務角色 ARN。系統會在存取資料來源時取得此角色。

您可以指定 Permissions 或 ServiceRoleArn，但不能同時指定兩者。

類型：字串

需要：否。 如果未指定，會 AWS SAM 套用的預設值Permissions。

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::DataSource資源的ServiceRoleArn屬性。

TableArn

DynamoDB 資料表的 ARN。

類型：字串

必要：有條件限制。如果未指定ServiceRoleArn，則TableArn為必填項。

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

TableName

資料表名稱。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::DataSource DynamoDBConfig對象的TableName屬性。

UseCallerCredentials

設定true為搭配此資料來源使用 IAM。

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::DataSource DynamoDBConfig對象的UseCallerCredentials屬性。

Versioned

設定為使true用「衝突偵測」、「衝突解決」和「同步處理」與此資料來源。

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::AppSync::DataSource` `DynamoDBConfig`對象的`Versioned`屬性。

Lambda

將 AWS Lambda 函數設定為 GraphQL API 解析器的資料來源。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
LogicalId:
  Description: String
  FunctionArn: String
  Name: String
  ServiceRoleArn: String
```

屬性

`Description`

資料來源的說明。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::AppSync::DataSource`資源的`Description`屬性。

`FunctionArn`

Lambda 函數的 ARN。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::DataSource LambdaConfig對象的LambdaFunctionArn屬性。

LogicalId

資料來源的唯一名稱。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::DataSource資源的Name屬性。

Name

資料來源的名稱。指定此性質以取代LogicalId值。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::DataSource資源的Name屬性。

ServiceRoleArn

資料來源的 AWS Identity and Access Management (IAM) 服務角色 ARN。系統會在存取資料來源時取得此角色。

> **Note**
> 若要撤銷對資料來源的存取權，請從 AWS SAM 範本中移除 Lambda 物件。

類型：字串

需要：否 如果未指定， AWS SAM 將使用佈建Write權限AWS SAM 連接器。

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::DataSource資源的ServiceRoleArn屬性。

函式

在 GraphQL API 中設定函數以執行特定作業。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
LogicalId:
  CodeUri: String
  DataSource: String
  Description: String
  Id: String
  InlineCode: String
  MaxBatchSize: Integer
  Name: String
  Runtime: Runtime
  Sync: SyncConfig
```

屬性

`CodeUri`

函數代碼的 Amazon Simple Storage Service (Amazon S3) URI 或本地文件夾的路徑。

如果您指定本機資料夾的路徑，則 AWS CloudFormation 需要在部署之前先將檔案上傳到 Amazon S3。您可以使用 AWS SAMCLI來促進此程序。如需詳細資訊，請參閱 如何在部署時上傳本地文件 AWS SAMCLI。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞
給`AWS::AppSync::FunctionConfiguration`資源的`CodeS3Location`屬性。

`DataSource`

此函數將附加到的數據源的名稱。

- 若要參照資源中的`AWS::Serverless::GraphQLApi`資料來源，請指定其邏輯 ID。

- 若要參考資源外部的資料來AWS::Serverless::GraphQLApi源，請使用Fn::GetAtt內建函數提供其Name屬性。例如 !GetAtt MyLambdaDataSource.Name。
- 若要參考來自不同堆疊的資料來源，請使用Fn::ImportValue。

如果指定[NONE | None | none]的變數， AWS SAM 將會產生AWS::AppSync::DataSourceType物件的None值。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::FunctionConfiguration資源的DataSourceName屬性。

`Description`

你的函數的描述。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::FunctionConfiguration資源的Description屬性。

`Id`

位於資源外部之函數的函數 AWS::Serverless::GraphQLApi ID。

- 要引用同一 AWS SAM 模板中的函數，請使用Fn::GetAtt內部函數。例如：Id: !GetAtt createPostItemFunc.FunctionId。
- 要引用來自不同堆棧的函數，請使用Fn::ImportValue.

使用時Id，不允許所有其他屬性。 AWS SAM 將自動傳遞引用函數的函數 ID。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`InlineCode`

包含請求和響應函數的函數代碼。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::FunctionConfiguration資源的Code屬性。

`LogicalId`

函數的唯一名稱。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::FunctionConfiguration資源的Name屬性。

`MaxBatchSize`

解析程式請求輸入的數量上限，輸入將傳送到 BatchInvoke 操作中的單一 AWS Lambda 函數。

類型：整數

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::FunctionConfiguration資源的MaxBatchSize屬性。

`Name`

函數的名稱。指定以取代LogicalId值。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::FunctionConfiguration資源的Name屬性。

`Runtime`

描述 AWS AppSync 管線解析程式或 AWS AppSync 函數使用的執行階段。指定要使用的執行階段名稱和版本。

類型：運行時

必要：是

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。它類似於資AWS::AppSync::FunctionConfiguration源的Runtime屬性。

Sync

描述函數的同步配置。

指定呼叫函數時要使用的衝突偵測策略與解決策略。

類型:SyncConfig

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::FunctionConfiguration資源的SyncConfig屬性。

執行期

管線解析程式或函數的執行階段。指定要使用的名稱和版本。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
Name: String
Version: String
```

屬性

Name

要使用的執行階段名稱。目前，唯一允許的值為 APPSYNC_JS。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::FunctionConfiguration AppSyncRuntime對象的Name屬性。

`Version`

要使用的執行階段版本。目前唯一允許的版本為 `1.0.0`。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::FunctionConfiguration AppSyncRuntime對象的RuntimeVersion屬性。

解析程式

設定 API 欄位的解析程式GraphQL。 AWS Serverless Application Model (AWS SAM) 支援JavaScript管線解析器。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
OperationType:
  LogicalId:
    Caching: CachingConfig
    CodeUri: String
    FieldName: String
    InlineCode: String
    MaxBatchSize: Integer
    Pipeline: List
    Runtime: Runtime
    Sync: SyncConfig
```

屬性

`Caching`

已啟動快取之解析程式的快取組態。

類型:CachingConfig

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::Resolver資源的CachingConfig屬性。

CodeUri

解析器函數代碼的 Amazon Simple Storage Service (Amazon S3) URI 或本地文件夾的路徑。

如果您指定本機資料夾的路徑，則 AWS CloudFormation 需要在部署之前先將檔案上傳到 Amazon S3。您可以使用 AWS SAMCLI來促進此程序。如需詳細資訊，請參閱 如何在部署時上傳本地文件 AWS SAMCLI。

如果兩CodeUri者InlineCode都沒有提供或者， AWS SAM 將生成InlineCode將請求重定向到第一個管道函數，並從最後一個管道函數接收響應。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::Resolver資源的CodeS3Location屬性。

FieldName

您的解析器的名稱。指定此性質以取代LogicalId值。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::Resolver資源的FieldName屬性。

InlineCode

包含請求和響應函數的解析器代碼。

如果兩CodeUri者InlineCode都沒有提供或者， AWS SAM 將生成InlineCode將請求重定向到第一個管道函數，並從最後一個管道函數接收響應。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::Resolver資源的Code屬性。

LogicalId

解析程式的唯一名稱。在GraphQL結構描述中，您的解析器名稱應與其用於的欄位名稱相符。使用相同的欄位名稱LogicalId。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

MaxBatchSize

解析程式請求輸入的數量上限，輸入將傳送到 BatchInvoke 操作中的單一 AWS Lambda 函數。

類型：整數

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::Resolver資源的MaxBatchSize屬性。

OperationType

與解析器相關聯的GraphQL操作類型。例如，Query、Mutation 或 Subscription。您可以LogicalId在單個解析器中嵌套多個解析器。OperationType

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::Resolver資源的TypeName屬性。

Pipeline

與解析程式連結的函數。通過列表中的邏輯 ID 指定函數。

類型：清單

必要：是

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。它類似於資AWS::AppSync::Resolver源的PipelineConfig屬性。

Runtime

管線解析程式或函數的執行階段。指定要使用的名稱和版本。

類型：運行時

必要：是

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。它類似於資AWS::AppSync::Resolver源的Runtime屬性。

Sync

描述解析程式的同步組態。

指定叫用解析程式時，要使用的衝突偵測策略和解決方案策略。

類型:SyncConfig

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::AppSync::Resolver資源的SyncConfig屬性。

範例

使用 AWS SAM 生成的解析器函數代碼並將字段另存為變量

以下是我們範例的GraphQL結構描述：

```
schema {
  query: Query
  mutation: Mutation
}

type Query {
  getPost(id: ID!): Post
}

type Mutation {
```

```
  addPost(author: String!, title: String!, content: String!): Post!
}

type Post {
  id: ID!
  author: String
  title: String
  content: String
}
```

這是我們的 AWS SAM 模板的一個片段：

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyGraphQLApi:
    Type: AWS::Serverless::GraphQLApi
    Properties:
      ...
      Functions:
        preprocessPostItem:
          ...
        createPostItem:
          ...
      Resolvers:
        Mutation:
          addPost:
            Runtime:
              Name: APPSYNC_JS
              Version: 1.0.0
            Pipeline:
            - preprocessPostItem
            - createPostItem
```

在我們的 AWS SAM 範本中，我們不指定`CodeUri`或`InlineCode`。在部署時， AWS SAM 自動為我們的解析器生成以下內聯代碼：

```
export function request(ctx) {
  return {};
}

export function response(ctx) {
```

```
  return ctx.prev.result;
}
```

此預設解析程式碼會將要求重新導向至第一個管線函式，並接收來自最後一個管線函式的回應。

在我們的第一個管道函數中，我們可以使用提供的`args`字段來解析請求對象並創建我們的變量。然後，我們可以在我們的函數中使用這些變量。下面是我們的`preprocessPostItem`功能的一個例子：

```
import { util } from "@aws-appsync/utils";

export function request(ctx) {
  const author = ctx.args.author;
  const title = ctx.args.title;
  const content = ctx.args.content;

  // Use variables to process data

}

export function response(ctx) {
  return ctx.result;
}
```

執行期

管線解析程式或函數的執行階段。指定要使用的名稱和版本。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
Name: String
Version: String
```

屬性

`Name`

要使用的執行階段名稱。目前，唯一允許的值為 `APPSYNC_JS`。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::AppSync::Resolver` `AppSyncRuntime`對象的`Name`屬性。

`Version`

要使用的執行階段版本。目前唯一允許的版本為 `1.0.0`。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::AppSync::Resolver` `AppSyncRuntime`對象的`RuntimeVersion`屬性。

AWS::Serverless::HttpApi

建立 Amazon API Gateway HTTP API，與其他 API 相比，您可以使用更低的延遲和更低的成本來建立 REST 風格 API。如需詳細資訊，請參閱 API Gateway 開發人員指南中的使用 HTTP API。

我們建議您使用 AWS CloudFormation 勾點或 IAM 政策來驗證 API Gateway 資源是否已附加授權人，以控制對其存取權限。

如需有關使用 AWS CloudFormation 勾點的詳細資訊，請參閱 AWS CloudFormation CLI 使用手冊和apigw-enforce-authorizer GitHub 存放庫中的註冊勾點。

如需使用 IAM 政策的詳細資訊，請參閱 API Gateway 開發人員指南中的「要求 API 路由具有授權」。

> **Note**
>
> 當您部署到時 AWS CloudFormation，將您的 AWS SAM 資源 AWS SAM 轉換為 AWS CloudFormation 資源。如需詳細資訊，請參閱 產生的 AWS CloudFormation 資源。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
Type: AWS::Serverless::HttpApi
Properties:
  AccessLogSettings: AccessLogSettings
  Auth: HttpApiAuth
  CorsConfiguration: String | HttpApiCorsConfiguration
  DefaultRouteSettings: RouteSettings
  DefinitionBody: JSON
  DefinitionUri: String | HttpApiDefinition
  Description: String
  DisableExecuteApiEndpoint: Boolean
  Domain: HttpApiDomainConfiguration
  FailOnWarnings: Boolean
  Name: String
  PropagateTags: Boolean
  RouteSettings: RouteSettings
  StageName: String
  StageVariables: Json
  Tags: Map
```

屬性

`AccessLogSettings`

階段中存取記錄的設定。

類型:AccessLogSettings

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::ApiGatewayV2::Stage`資源的`AccessLogSettings`屬性。

`Auth`

設定授權以控制 API Gateway HTTP API 的存取權。

如需詳細資訊，請參閱《API Gateway 開發人員指南》中的使用 JWT 授權方控制對 HTTP API 的存取。

類型:HttpApiAuth

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`CorsConfiguration`

管理所有 API Gateway HTTP API 的跨來源資源共用 (CORS)。將允許的網域指定為字串，或指定`HttpApiCorsConfiguration`物件。請注意，CORS 需 AWS SAM 要修改您的 OpenAPI 定義，因此 CORS 僅在指定`DefinitionBody`屬性時才起作用。

如需詳細資訊，請參閱 API Gateway 開發人員指南中的針對 HTTP API 設定 CORS。

> **Note**
>
> 如果`CorsConfiguration`同時在 OpenAPI 定義和屬性層級設定，則會將這兩個組態來源與優先順序 AWS SAM 合併。如果將此屬性設定為`true`，則允許所有原點。

類型：字符串 | HttpApiCorsConfiguration

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`DefaultRouteSettings`

此 HTTP API 的預設路由設定。除非被某些佈線的`RouteSettings`性質取代，否則這些設定適用於所有佈線。

類型:RouteSettings

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::ApiGatewayV2::Stage`資源的`RouteSettings`屬性。

`DefinitionBody`

用於描述您的 HTTP API 的開發 API 定義。如果您未指定`DefinitionUri`或`DefinitionBody`，請根據範本設定`DefinitionBody`為您 AWS SAM 產生一個。

類型：JSON

必要：否

AWS CloudFormation 兼容性：此屬性類似於AWS::ApiGatewayV2::Api資源的Body屬性。如果提供了某些屬性， AWS SAM 可以在內容傳遞到之DefinitionBody前將內容插入或修改 AWS CloudFormation。屬性包括對應AWS::Serverless::Function資源EventSource HttpApi 的類型Auth和類型。

`DefinitionUri`

Amazon Simple Storage Service (Amazon S3) URI、本機檔案路徑或定義 HTTP API 定義的位置物件。此屬性參考的 Amazon S3 物件必須是有效的 OpenAPI 定義檔案。如果您未指定DefinitionUri或指定，DefinitionBody則 AWS SAM 會根據範本設定DefinitionBody為您產生一個。

如果您提供本機檔案路徑，則範本必須經過包含sam deploy或sam package指令的工作流程，以便正確轉換定義。

您參考的外部 OpenApi 定義檔案不支援內建函數。DefinitionUri若要將定 OpenApi 義匯入範本，請搭配 In clude 轉換使用DefinitionBody屬性。

類型：字符串 | HttpApiDefinition

必要：否

AWS CloudFormation 兼容性：此屬性類似於AWS::ApiGatewayV2::Api資源的BodyS3Location屬性。巢狀 Amazon S3 屬性的名稱不同。

`Description`

HTTP API 資源的說明。

當您指定時Description， AWS SAM 會透過設定description欄位來修改 HTTP API 資源的定OpenApi 義。以下情況將導致錯誤：

- DefinitionBody屬性是使用 Open API 定義中設定的description欄位來指定 — 這會導致 AWS SAM 無法解決的description欄位衝突。
- 已指定DefinitionUri屬性 — AWS SAM 不會修改從 Amazon S3 擷取的開放式 API 定義。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`DisableExecuteApiEndpoint`

指定用戶端是否可以使用預設`execute-api`端點叫用您的 HTTP API `https://{api_id}.execute-api.{region}.amazonaws.com`。根據預設，用戶端可以使用預設 端點叫用您的 API。若要求用戶端僅使用自訂網域名稱來叫用您的 API，請停用預設端點。

若要使用此屬性，您必須指定`DefinitionBody`屬性而非`DefinitionUri`屬性，或在 OpenAPI 定義`disableExecuteApiEndpoint`中使`x-amazon-apigateway-endpoint-configuration`用定義。

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性類似於`AWS::ApiGatewayV2::Api`資源的 `DisableExecuteApiEndpoint`屬性。它會直接傳遞至擴充功能的`disableExecuteApiEndpoint`屬性，而 `x-amazon-apigateway-endpoint-configuration`擴充功能會新增至`AWS::ApiGatewayV2::Api`資源的 `Body`屬性。

`Domain`

設定此 API Gateway HTTP API 的自訂網域。

類型:HttpApiDomainConfiguration

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`FailOnWarnings`

指定遇到警告時是否回滾 HTTP API 創建（`true`或`false`）或不（ ）。預設值為 `false`。

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::ApiGatewayV2::Api`資源的`FailOnWarnings`屬性。

`Name`

HTTP API 資源的名稱。

當您指定時`Name`， AWS SAM 會透過設定`title`欄位來修改 HTTP API 資源的 OpenAPI 定義。以下情況將導致錯誤：

- `DefinitionBody`屬性是使用 Open API 定義中設定的`title`欄位來指定 — 這會導致 AWS SAM 無法解決的`title`欄位衝突。
- 已指定`DefinitionUri`屬性 — AWS SAM 不會修改從 Amazon S3 擷取的開放式 API 定義。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`PropagateTags`

指出是否要將標籤從`Tags`屬性傳遞至您AWS::Serverless::HttpApi產生的資源。指`True`定在產生的資源中傳播標籤。

類型：布林值

必要：否

預設：`False`

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`RouteSettings`

此 HTTP API 的每個路由的路由設定。如需詳細資訊，請參閱 API Gateway 開發人員指南中的使用 HTTP API 的路由。

類型:RouteSettings

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::ApiGatewayV2::Stage`資源的`RouteSettings`屬性。

`StageName`

API 階段的名稱。如果未指定名稱， AWS SAM 會使用來自 API Gateway 的`$default`階段。

類型：字串

必要：否

默認值：$ 默認

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::ApiGatewayV2::Stage`資源的`StageName`屬性。

`StageVariables`

定義階段變數的映射。變數名稱可以包含字母數字和底線字元。這些值必須符合 [A-ZA-Z0-9-._ ~:/?#&=,] +.

類型：Json

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::ApiGatewayV2::Stage`資源的`StageVariables`屬性。

`Tags`

映射（字符串到字符串），指定要添加到此 API Gateway 階段的標籤。金鑰的長度可以是 1 到 128 個 Unicode 字元，且不能包含前置字元`aws:`。您可以使用以下任何字元：Unicode 字母、數字、空格、_、 .、 /、 =、 + 和 - 的組合。值的長度可以是 1 到 256 個萬國碼字元。

類型：地圖

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

其他附註：`Tags`屬性需 AWS SAM 要修改 OpenAPI 定義，因此只有在指定屬性時才會新增標籤 — 如果指定`DefinitionBody`屬性，則不會新增標籤。`DefinitionUri` AWS SAM 會自動加入`httpapi:createdBy:SAM`標籤。標籤也會新增至資`AWS::ApiGatewayV2::Stage`源和資`AWS::ApiGatewayV2::DomainName`源 (如果`DomainName`已指定)。

傳回值

Ref

當您將此資源的邏輯 ID 傳遞給內建Ref函數時，會Ref傳回基礎AWS::ApiGatewayV2::Api資源的 API ID，例如。a1bcdef2gh

若要取得有關使用Ref功能的更多資訊，請參閱《使AWS CloudFormation 用指南》*Ref*中的〈〉

範例

簡單 HttpApi

下列範例顯示設定由 Lambda 函數支援的 HTTP API 端點所需的最低限度。此範例使用 AWS SAM 建立的預設 HTTP API。

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Description: AWS SAM template with a simple API definition
Resources:
  ApiFunction:
    Type: AWS::Serverless::Function
    Properties:
      Events:
        ApiEvent:
          Type: HttpApi
      Handler: index.handler
      InlineCode: |
        def handler(event, context):
            return {'body': 'Hello World!', 'statusCode': 200}
      Runtime: python3.7
Transform: AWS::Serverless-2016-10-31
```

HttpApi 與身份驗證

下列範例顯示如何在 HTTP API 端點上設定授權。

YAML

```
Properties:
  FailOnWarnings: true
```

```
  Auth:
    DefaultAuthorizer: OAuth2
    Authorizers:
      OAuth2:
        AuthorizationScopes:
          - scope4
        JwtConfiguration:
          issuer: "https://www.example.com/v1/connect/oauth2"
          audience:
            - MyApi
        IdentitySource: "$request.querystring.param"
```

HttpApi使用 OpenAPI 定義

下列範例會示範如何將 OpenAPI 定義新增至範本。

請注意，針對參考此 HTTP API 的 HttpApi 事件， AWS SAM 填入任何遺失的 Lambda 整合。 AWS SAM 也會新增 HttpApi 事件參照的任何遺失路徑。

YAML

```
Properties:
  FailOnWarnings: true
  DefinitionBody:
    info:
      version: '1.0'
      title:
        Ref: AWS::StackName
    paths:
      "/":
        get:
          security:
          - OpenIdAuth:
            - scope1
            - scope2
          responses: {}
    openapi: 3.0.1
    securitySchemes:
      OpenIdAuth:
        type: openIdConnect
        x-amazon-apigateway-authorizer:
          identitySource: "$request.querystring.param"
          type: jwt
```

```
          jwtConfiguration:
            audience:
            - MyApi
            issuer: https://www.example.com/v1/connect/oidc
          openIdConnectUrl: https://www.example.com/v1/connect/oidc/.well-known/openid-
configuration
```

HttpApi 與配置設置

下列範例會示範如何將 HTTP API 和階段設定新增至範本。

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

設定授權以控制對 Amazon API Gateway HTTP API 的存取。

如需設定 HTTP API 存取權的詳細資訊，請參閱《API Gateway 開發人員指南》中的「在 API Gateway 中控制和管理 HTTP API 的存取」。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
Authorizers: OAuth2Authorizer | LambdaAuthorizer
DefaultAuthorizer: String
EnableIamAuthorizer: Boolean
```

屬性

`Authorizers`

用來控制 API Gateway API 存取權的授權者。

類型：授權者 | LambdaAuthorizer

必要：否

預設：無

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

其他注意事項： AWS SAM 將授權者新增至 OpenAPI 定義中。

`DefaultAuthorizer`

指定用於授權 API Gateway API 呼叫的預設授權者。如`EnableIamAuthorizer`果設定`AWS_IAM`為，您可以指定為預設授權者。`true`否則，請指定您在`Authorizers`中定義的授權者。

類型：字串

必要：否

預設：無

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`EnableIamAuthorizer`

指定是否對 API 路由使用 IAM 授權。

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

驗證 2.0 授權者

驗證 2.0 授權者示例

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 授權者

IAM 授權者範例

YAML

```
Auth:
  EnableIamAuthorizer: true
  DefaultAuthorizer: AWS_IAM
```

LambdaAuthorizer

設定 Lambda 授權器，以透過某個 AWS Lambda 函數控制對 Amazon API Gateway HTTP API 的存取。

如需詳細資訊和範例，請參閱 API Gateway 開發人員指南中的使用 HTTP API 的 AWS Lambda 授權人員。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
AuthorizerPayloadFormatVersion: String
EnableFunctionDefaultPermissions: Boolean
EnableSimpleResponses: Boolean
FunctionArn: String
FunctionInvokeRole: String
Identity: LambdaAuthorizationIdentity
```

屬性

`AuthorizerPayloadFormatVersion`

指定傳送至 HTTP API Lambda 授權方的承載格式。HTTP API Lambda 授權方的必要項目。

這會傳遞至 OpenAPI 定義一`securitySchemes`節`x-amazon-apigateway-authorizer`中的一節。`authorizerPayloadFormatVersion`

有效值：`1.0` 或 `2.0`

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`EnableFunctionDefaultPermissions`

依預設，HTTP API 資源不會被授與叫用 Lambda 授權者的權限。將此屬性指定`true`為可在 HTTP API 資源和 Lambda 授權者之間自動建立權限。

類型：布林值

必要：否

預設值：`false`

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`EnableSimpleResponses`

指定 Lambda 授權方是否以簡單格式傳回回應。根據預設，Lambda 授權者必須傳回 AWS Identity and Access Management (IAM) 政策。如果啟用，Lambda 授權方會傳回布林值，而不是 IAM 政策。

這會傳遞至 OpenAPI 定義一`securitySchemes`節`x-amazon-apigateway-authorizer`中的一節。`enableSimpleResponses`

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`FunctionArn`

提供 API 授權的 Lambda 函數的 Amazon 資源名稱 (ARN)。

這會傳遞至 OpenAPI 定義一`securitySchemes`節`x-amazon-apigateway-authorizer`中的一節。`authorizerUri`

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`FunctionInvokeRole`

IAM 角色的 ARN，具有 API Gateway 呼叫授權者函數所需的登入資料。如果函數的資源型政策未授與 API Gateway `lambda:InvokeFunction` 權限，請指定此參數。

這會傳遞至 OpenAPI 定義一`securitySchemes`節`x-amazon-apigateway-authorizer`中的一節。`authorizerCredentials`

如需詳細資訊，請參閱 API Gateway 開發人員指南中的建立 Lambda 授權者。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Identity`

指定一個`IdentitySource`在授權者的傳入請求。

這會傳遞至 OpenAPI 定義一`securitySchemes`節`x-amazon-apigateway-authorizer`中的一節。`identitySource`

類型:LambdaAuthorizationIdentity

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

LambdaAuthorizer

LambdaAuthorizer 例子

YAML

```
Auth:
  Authorizers:
    MyLambdaAuthorizer:
      AuthorizerPayloadFormatVersion: 2.0
      FunctionArn:
        Fn::GetAtt:
          - MyAuthFunction
          - Arn
      FunctionInvokeRole:
        Fn::GetAtt:
          - LambdaAuthInvokeRole
          - Arn
      Identity:
        Headers:
          - Authorization
```

LambdaAuthorizationIdentity

使用屬性可用於 IdentitySource 在 Lambda 授權者的傳入請求中指定。如需身分識別來源的詳細資訊，請參閱 API Gateway 開發人員指南中的身分識別來源。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

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

屬性

`Context`

將指定的上下文字串轉換為格式的對應運算式清單`$context.contextString`。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Headers`

將標題轉換為格式的對映運算式清單`$request.header.name`。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`QueryStrings`

將指定的查詢字串轉換為格式的對應運算式清單`$request.querystring.queryString`。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`ReauthorizeEvery`

指定 API Gateway 快取授權者結果的時間 time-to-live (TTL) 期間 (以秒為單位)。如果您指定的值大於 0，則 API Gateway 會快取授權方回應。值的上限為 3600 (1 小時)。

類型：整數

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`StageVariables`

將指定的階段變數轉換為格式中的對應運算式清單`$stageVariables.stageVariable`。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

LambdaRequestIdentity

請 Lambda 身分識別範例

YAML

```
Identity:
  QueryStrings:
    - auth
  Headers:
    - Authorization
```

```
  StageVariables:
    - VARIABLE
  Context:
    - authcontext
  ReauthorizeEvery: 100
```

OAuth2Authorizer

OAuth 2.0 授權者的定義，也稱為 JSON 網絡令牌（JWT）授權者。

如需詳細資訊，請參閱《API Gateway 開發人員指南》中的使用 JWT 授權方控制對 HTTP API 的存取。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
AuthorizationScopes: List
IdentitySource: String
JwtConfiguration: Map
```

屬性

`AuthorizationScopes`

此授權者的授權範圍清單。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`IdentitySource`

此授權者的識別來源運算式。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`JwtConfiguration`

此授權者的 JWT 設定。

這會傳遞至 OpenAPI 定義一`securitySchemes`節`x-amazon-apigateway-authorizer`中的一節。`jwtConfiguration`

> **Note**
>
> 屬性`issuer`和`audience`不區分大小寫，可以在 OpenAPI 中使用小寫字母或大寫字母`Issuer`和`Audience`。AWS::ApiGatewayV2::Authorizer

類型：地圖

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

驗證 2.0 授權者

驗證 2.0 授權者示例

YAML

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

HttpApiCorsConfiguration

管理 HTTP API 的跨來源資源共用 (CORS)。指定允許作為字符串的域或指定帶有其他 CRS 配置的字典。注意：科爾斯要求 SAM 修改您的 OpenAPI 定義，因此它僅適用於在屬性中 OpenApi 定義的內聯定義。DefinitionBody

如需有關 CORS 的詳細資訊，請參閱 API Gateway 開發人員指南中的針對 HTTP API 設定 CO RS。

注意：如果 HttpApiCorsConfiguration 同時在 OpenAPI 和屬性層級設定，則會將它們與 AWS SAM 優先順序的屬性合併。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
AllowCredentials: Boolean
AllowHeaders: List
AllowMethods: List
AllowOrigins: List
ExposeHeaders: List
MaxAge: Integer
```

屬性

`AllowCredentials`

指定 CORS 請求中是否包含憑證。

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`AllowHeaders`

代表允許標頭的集合。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`AllowMethods`

代表允許的 HTTP 方法集合。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`AllowOrigins`

代表允許的來源集合。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`ExposeHeaders`

代表公開標頭的集合。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`MaxAge`

瀏覽器應快取預檢請求結果的秒數。

類型：整數

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

HttpApiCorsConfiguration

HTTP API 協議配置示例。

YAML

```
CorsConfiguration:
  AllowOrigins:
    - "https://example.com"
  AllowHeaders:
    - x-apigateway-header
  AllowMethods:
    - GET
  MaxAge: 600
  AllowCredentials: true
```

HttpApiDefinition

一個 OpenAPI 文檔定義的 API。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

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

屬性

`Bucket`

存放 OpenAPI 檔案所在的 Amazon S3 儲存貯體的名稱。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::ApiGatewayV2::ApiBodyS3Location`數據類型的`Bucket`屬性。

Key

OpenAPI 文件的 Amazon S3 密鑰。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::ApiGatewayV2::ApiBodyS3Location數據類型的Key屬性。

Version

對於已建立版本的物件，則為 OpenAPI 檔案的版本。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::ApiGatewayV2::ApiBodyS3Location數據類型的Version屬性。

範例

定義 Uri 範例

API 定義範例

YAML

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

HttpApiDomainConfiguration

設定 API 的自訂網域。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
  BasePath: List
  CertificateArn: String
  DomainName: String
  EndpointConfiguration: String
  MutualTlsAuthentication: MutualTlsAuthentication
  OwnershipVerificationCertificateArn: String
  Route53: Route53Configuration
  SecurityPolicy: String
```

屬性

`BasePath`

要使用 Amazon API Gateway 網域名稱設定的基本路徑清單。

類型：清單

必要：否

預設值：/

AWS CloudFormation兼容性：此屬性類似於`AWS::ApiGatewayV2::ApiMapping`資源的`ApiMappingKey`屬性。 AWS SAM會建立多個`AWS::ApiGatewayV2::ApiMapping`資源，每個在此屬性中指定的值一個資源。

`CertificateArn`

此網域名稱端點的AWS受管憑證的 Amazon 資源名稱 (ARN)。 AWS Certificate Manager是唯一受支援的來源。

類型：字串

必要：是

AWS CloudFormation兼容性：此屬性直接傳遞給`AWS::ApiGateway2::DomainName` `DomainNameConfiguration`資源的`CertificateArn`屬性。

`DomainName`

API Gateway API 的自訂網域名稱。不支援大寫字母。

AWS SAM設置此屬性時生成一個AWS::ApiGatewayV2::DomainName資源。如需有關此案例的資訊，請參閱DomainName屬性已指定。如需有關已產生AWS CloudFormation資源的資訊，請參閱產生的 AWS CloudFormation 資源。

類型：字串

必要：是

AWS CloudFormation兼容性：此屬性直接傳遞給AWS::ApiGateway2::DomainName資源的DomainName屬性。

`EndpointConfiguration`

定義要對應至自訂網域的 API Gateway 端點類型。此屬性的值決定了對應CertificateArn性質的方式AWS CloudFormation。

HTTP API 的唯一有效值是REGIONAL。

類型：字串

必要：否

預設：REGIONAL

AWS CloudFormation兼容性：此屬性是唯一的，AWS SAM並且沒有相AWS CloudFormation等的屬性。

`MutualTlsAuthentication`

自訂網域名稱的相互傳輸層安全性 (TLS) 驗證組態。

類型:MutualTlsAuthentication

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給AWS::ApiGatewayV2::DomainName資源的MutualTlsAuthentication屬性。

`OwnershipVerificationCertificateArn`

ACM 核發之公有憑證的 ARN，用於驗證您的自訂網域的擁有權。只有當您設定相互 TLS，並指定 ACM 匯入或私有 CA 憑證 ARN 給. CertificateArn

類型：字串

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給AWS::ApiGatewayV2::DomainNameDomainNameConfiguration數據類型的OwnershipVerificationCertificateArn屬性。

Route53

定義 Amazon 路線 53 配置。

類型：路線 53 配置

必要：否

AWS CloudFormation兼容性：此屬性是唯一的，AWS SAM並且沒有相AWS CloudFormation等的屬性。

SecurityPolicy

此網域名稱的安全性原則的 TLS 版本。

HTTP API 的唯一有效值是TLS_1_2。

類型：字串

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給AWS::ApiGatewayV2::DomainNameDomainNameConfiguration數據類型的SecurityPolicy屬性。

範例

DomainName

DomainName 例子

YAML

```
Domain:
  DomainName: www.example.com
  CertificateArn: arn-example
  EndpointConfiguration: REGIONAL
  Route53:
    HostedZoneId: Z1PA6795UKMFR9
  BasePath:
```

```
    - foo
    - bar
```

Route53Configuration

設定 API 的路由 53 記錄集。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

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

屬性

`DistributionDomainName`

設定 API 自訂網域名稱的自訂分發。

類型：字串

必要：否

預設值：使用 API Gateway 分發。

AWS CloudFormation兼容性：此屬性直接傳遞給`AWS::Route53::RecordSetGroup` `AliasTarget`資源的DNSName屬性。

其他注意事項：CloudFront發行版的網域名稱。

`EvaluateTargetHealth`

當 EvaluateTargetHealth 為 true 時，別名記錄會繼承參考AWS資源的健全狀況，例如 Elastic Load Balancing 負載平衡器或託管區域中的其他記錄。

類型：布林值

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給AWS::Route53::RecordSetGroup AliasTarget資源的EvaluateTargetHealth屬性。

其他注意事項：當別名目標是 CloudFront 發佈時，您無法設定 EvaluateTargetHealth 為 true。

HostedZoneId

託管區域的 ID，您要在其中建立記錄。

請指定 HostedZoneName 或 HostedZoneId 其中之一。若有多個託管區域的網域名稱相同，則您必須使用 HostedZoneId 以明確指定託管區域。

類型：字串

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給AWS::Route53::RecordSetGroup RecordSet資源的HostedZoneId屬性。

HostedZoneName

您要在其中建立記錄的託管區域名稱。您必須包含結尾點 (例如 www.example.com.) 作為 HostedZoneName 的一部分。

請指定 HostedZoneName 或 HostedZoneId 其中之一。若有多個託管區域的網域名稱相同，則您必須使用 HostedZoneId 以明確指定託管區域。

類型：字串

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給AWS::Route53::RecordSetGroup RecordSet資源的HostedZoneName屬性。

IpV6

設定此屬性後，AWS SAM會為提供的建立AWS::Route53::RecordSet資源並將「類型」設定AAAA為 HostedZone。

類型：布林值

必要：否

AWS CloudFormation兼容性：此屬性是唯一的，AWS SAM並且沒有相AWS CloudFormation等的屬性。

`Region`

僅限延遲型資源記錄集：您建立資源的 Amazon EC2 區域，這是此資源記錄集指向的資源。資源一般是 AWS 資源 (例如 EC2 執行個體或 ELB 負載平衡器)，且以 IP 地址或 DNS 網域名稱表示 (視記錄類型而定)。

當 Amazon Route 53 收到網域名稱和類型的 DNS 查詢，而您已建立其延遲資源記錄集時，Route 53 會選取最低延遲介於最終使用者與相關聯 Amazon EC2 區域之間的延遲資源記錄集。Route 53 接著會傳回與所選資源記錄集相關聯的值。

注意下列事項：

- 每個延遲資源記錄集只能指定一個 `ResourceRecord`。
- 每個 Amazon EC2 區域都只能建立一個延遲資源記錄集。
- 您不必為所有的 Amazon EC2 區域建立延遲資源記錄集。Route 53 會從您建立延遲資源記錄集的區域中，選擇具有最佳延遲的區域。
- 您無法建立和延遲資源記錄集的 `Name` 和 `Type` 元素具有相同值的非延遲資源記錄集。

類型：字串

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給`AWS::Route53::RecordSetGroupRecordSet`數據類型的 `Region`屬性。

`SetIdentifier`

沒有簡單路由政策的資源記錄集：在具有相同名稱和類型組合的多個資源記錄集中用以區隔的識別碼，例如，多個加權資源記錄集名為 acme.example.com 且類型為 A。在一組具有相同名稱和類型的資源記錄集中，每個資源記錄集的 `SetIdentifier` 值都必須是唯一的。

如需有關路由政策的資訊，請參閱 Amazon Route 53 開發人員指南中的選擇路由政策。

類型：字串

必要：否

AWS CloudFormation兼容性：此屬性直接傳遞給`AWS::Route53::RecordSetGroupRecordSet`數據類型的 `SetIdentifier`屬性。

範例

Route 53 組態範例

此範例顯示如何設定路由 53。

YAML

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

AWS::Serverless::LayerVersion

建立 Lambda LayerVersion ，其中包含 Lambda 函數所需的程式庫或執行階段程式碼。

該AWS::Serverless::LayerVersion資源還支持 `Metadata` resource 屬性，因此您可以指示 AWS SAM 構建包含在應用程序中的層。如需建置圖層的更多資訊，請參閱〈〉建置 Lambda 層。

重要注意事項：自中的UpdateReplacePolicy資源屬性發行以來 AWS CloudFormation，AWS::Lambda::LayerVersion(建議) 提供與中相同的優點AWS::Serverless::LayerVersion。

轉換無伺服器 LayerVersion 時，SAM 也會轉換資源的邏輯 ID，以便 LayerVersions 在資源更新 CloudFormation 時不會自動刪除舊的 ID。

> **Note**
>
> 當您部署到時 AWS CloudFormation，將您的 AWS SAM 資源 AWS SAM 轉換為 AWS CloudFormation 資源。如需詳細資訊，請參閱 產生的 AWS CloudFormation 資源。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
Type: AWS::Serverless::LayerVersion
Properties:
  CompatibleArchitectures: List
  CompatibleRuntimes: List
  ContentUri: String | LayerContent
  Description: String
  LayerName: String
  LicenseInfo: String
  RetentionPolicy: String
```

屬性

`CompatibleArchitectures`

指定圖層版本支援的指令集架構。

如需有關此屬性的詳細資訊，請參閱AWS Lambda 開發人員指南中的 Lambda 指令集架構。

有效值：`x86_64`、`arm64`

類型：列表

必要：否

預設：`x86_64`

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::LayerVersion`資源的`CompatibleArchitectures`屬性。

`CompatibleRuntimes`

與此 LayerVersion相容的執行階段清單。

類型：列表

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::LayerVersion`資源的`CompatibleRuntimes`屬性。

`ContentUri`

Amazon S3 Uri，路徑到本地文件夾，或層代碼的 LayerContent 對象。

如果提供了 Amazon S3 Uri 或 LayerContent 物件，則所參照的 Amazon S3 物件必須是包含 Lambda 層內容的有效 ZIP 存檔。

如果提供了本機資料夾的路徑，要正確轉換內容，範本必須經過包含sam deploy或之sam build後的工作流程sam package。依預設，相對路徑會根據 AWS SAM 樣板的位置進行解析。

類型：字符串 | LayerContent

必要：是

AWS CloudFormation 兼容性：此屬性類似於AWS::Lambda::LayerVersion資源的Content屬性。巢狀 Amazon S3 屬性的名稱不同。

`Description`

此圖層的描述。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::LayerVersion資源的Description屬性。

`LayerName`

layer 的名稱或 Amazon Resource Name (ARN)。

類型：字串

必要：否

預設值：資源邏輯 ID

AWS CloudFormation 兼容性：此屬性類似於AWS::Lambda::LayerVersion資源的LayerName屬性。如果您未指定名稱，則會使用資源的邏輯 ID 做為名稱。

`LicenseInfo`

有關此授權的資訊 LayerVersion。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Lambda::LayerVersion資源的LicenseInfo屬性。

RetentionPolicy

此屬性指定刪除資源時，LayerVersion是否保留或刪除舊版本。如果您在更新或取代資源LayerVersion時需要保留舊版本的，則必須啟用UpdateReplacePolicy屬性。如需執行此操作的資訊，請參閱《AWS CloudFormation 使用指南》中的「UpdateReplacePolicy屬性」。

有效值：Retain 或 Delete

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

其他附註：當您指定時Retain，會 AWS SAM 將支援的資源屬性 AWS SAM的新增DeletionPolicy: Retain至已轉換的AWS::Lambda::LayerVersion資源。

傳回值

Ref

將此資源的邏輯 ID 提供給Ref內建函數時，它會傳回基礎 Lambda 的資源 ARN。 LayerVersion

若要取得有關使用Ref功能的更多資訊，請參閱《使AWS CloudFormation 用指南》*Ref*中的〈〉

範例

LayerVersionExample

一個例子 LayerVersion

YAML

```
Properties:
  LayerName: MyLayer
  Description: Layer description
  ContentUri: 's3://my-bucket/my-layer.zip'
  CompatibleRuntimes:
    - nodejs10.x
```

```
    - nodejs12.x
  LicenseInfo: 'Available under the MIT-0 license.'
  RetentionPolicy: Retain
```

LayerContent

包含 Lambda 層內容的 ZIP 封存。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

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

屬性

`Bucket`

層封存的 Amazon S3 儲存貯體。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::LayerVersionContent`數據類型的`S3Bucket`屬性。

`Key`

層封存的 Amazon S3 金鑰。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::LayerVersionContent`數據類型的`S3Key`屬性。

`Version`

對於版本控制的物件，要使用的層封存物件版本。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Lambda::LayerVersionContent`數據類型的`S3ObjectVersion`屬性。

範例

LayerContent

圖層內容範例

YAML

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

AWS::Serverless::SimpleTable

建立具有單一屬性主索引鍵的 DynamoDB 表格。當數據只需要通過主鍵訪問時，這很有用。

若要使用 DynamoDB 更進階的功能，請改用AWS::DynamoDB::Table資源。

> **Note**
> 當您部署到時 AWS CloudFormation，將您的 AWS SAM 資源 AWS SAM 轉換為 AWS CloudFormation 資源。如需詳細資訊，請參閱 產生的 AWS CloudFormation 資源。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
Type: AWS::Serverless::SimpleTable
Properties:
```

```
  PointInTimeRecoverySpecification: PointInTimeRecoverySpecification
  PrimaryKey: PrimaryKeyObject
  ProvisionedThroughput: ProvisionedThroughput
  SSESpecification: SSESpecification
  TableName: String
  Tags: Map
```

屬性

`PointInTimeRecoverySpecification`

用於啟用時間點復原恢復的設定。

類型:PointInTimeRecoverySpecification

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::DynamoDB::Table`資源的`PointInTimeRecoverySpecification`屬性。

`PrimaryKey`

屬性名稱和類型被用作表的主鍵。如果沒有提供，主鍵將是`String`一個值為`id`。

> **Note**
>
> 建立此資源之後，就無法修改此屬性的值。

類型:PrimaryKeyObject

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`ProvisionedThroughput`

讀取和寫入輸送量佈建資訊。

如果`ProvisionedThroughput`未指定`BillingMode`將被指定為`PAY_PER_REQUEST`。

類型:ProvisionedThroughput

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::DynamoDB::Table資源的ProvisionedThroughput屬性。

SSESpecification

指定此屬性來啟用伺服器端加密。

類型：SSESpecification

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::DynamoDB::Table資源的SSESpecification屬性。

TableName

動態資料表的名稱。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::DynamoDB::Table資源的TableName屬性。

Tags

一個映射（字符串到字符串），指定要添加到此標籤 SimpleTable。如需有關標籤的有效鍵和值的詳細資訊，請參閱《AWS CloudFormation 使用指南》中的〈Resource 標籤〉。

類型：地圖

必要：否

AWS CloudFormation 兼容性：此屬性類似於AWS::DynamoDB::Table資源的Tags屬性。SAM 中的標籤屬性由鍵：值對組成; CloudFormation 其中包含標籤對象的列表。

傳回值

Ref

將此資源的邏輯 ID 提供給 Ref 內建函數時，它會傳回基礎 DynamoDB 表格的資源名稱。

若要取得有關使用Ref功能的更多資訊，請參閱《使AWS CloudFormation 用指南》*Ref*中的〈〉

範例

SimpleTableExample

一個例子 SimpleTable

YAML

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

PrimaryKeyObject

描述主鍵屬性的對象。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
  Name: String
  Type: String
```

屬性

Name

主鍵的屬性名稱。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::DynamoDB::TableAttributeDefinition數據類型的AttributeName屬性。

其他注意事項：此屬性也會傳遞給AWS::DynamoDB::Table KeySchema資料類型的AttributeName屬性。

Type

主索引鍵的資料類型。

有效值：String、Number、Binary

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::DynamoDB::TableAttributeDefinition數據類型的AttributeType屬性。

範例

PrimaryKey

主鍵示例。

YAML

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

AWS::Serverless::StateMachine

建立 AWS Step Functions 狀態機器，您可以使用它來協調 AWS Lambda 功能和其他 AWS 資源，以形成複雜且強大的工作流程。

如需 Step Functions 的詳細資訊，請參閱《AWS Step Functions 開發人員指南》。

> **Note**
>
> 當您部署到時 AWS CloudFormation，將您的 AWS SAM 資源 AWS SAM 轉換為 AWS CloudFormation 資源。如需詳細資訊，請參閱 產生的 AWS CloudFormation 資源。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
Type: AWS::Serverless::StateMachine
Properties:
  AutoPublishAlias: String
  Definition: Map
  DefinitionSubstitutions: Map
  DefinitionUri: String | S3Location
  DeploymentPreference: DeploymentPreference
  Events: EventSource
  Logging: LoggingConfiguration
  Name: String
  PermissionsBoundary: String
  Policies: String | List | Map
  PropagateTags: Boolean
  RolePath: String
  Role: String
  Tags: Map
  Tracing: TracingConfiguration
  Type: String
```

屬性

`AutoPublishAlias`

狀態機器別名的名稱。若要深入了解如何使用 Step Functions 狀態機器別名，請參閱AWS Step Functions 開發人員指南中的使用版本和別名管理持續部署。

用於設`DeploymentPreference`定別名的部署偏好設定。如果您未指定`DeploymentPreference`，則 AWS SAM 會將流量設定為一次全部轉移到較新的狀態機版本。

AWS SAM 默認情況下將版本`UpdateReplacePolicy`的`DeletionPolicy`和`Retain`設置為。以前的版本將不會自動刪除。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::StepFunctions::StateMachineAlias`資源的 `Name`屬性。

`Definition`

狀態機器定義是一個對象，其中對象的格式與 AWS SAM 模板文件的格式匹配，例如 JSON 或 YAML。狀態機器定義遵循 Amazon 狀態語言。

如需內嵌狀態機定義的範例，請參閱範例。

您必須提供一個`Definition`或`DefinitionUri`.

類型：地圖

必要：有條件

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`DefinitionSubstitutions`

一種 string-to-string 映射，指定狀態機定義中佔位符變量的映射。這可讓您將在執行階段取得的值 (例如，從內建函式) 插入狀態機器定義中。

類型：地圖

必要：否

AWS CloudFormation 兼容性：此屬性類似於`AWS::StepFunctions::StateMachine`資源的`DefinitionSubstitutions`屬性。如果在內嵌狀態機器定義中指定了任何內建函數，則會將項目 AWS SAM 新增至此屬性，以將它們插入到狀態機器定義中。

`DefinitionUri`

Amazon 簡單儲存服務 (Amazon S3) URI 或以亞馬遜州語言撰寫的狀態機器定義的本機檔案路徑。

如果您提供本機檔案路徑，則範本必須經過包含`sam deploy`或`sam package`指令的工作流程，才能正確轉換定義。若要這麼做，您必須使用 AWS SAM CLI 的 0.52.0 版或更新版本。

您必須提供一個`Definition`或`DefinitionUri`.

類型：字符串 | 第 3 位置

必要：有條件

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::StepFunctions::StateMachine`資源的`DefinitionS3Location`屬性。

`DeploymentPreference`

啟用和設定漸進狀態機器部署的設定。若要深入了解 Step Functions 逐步部署，請參閱AWS Step Functions 開發人員指南中的使用版本和別名管理持續部署。

`AutoPublishAlias`在配置此屬性之前指定。您的`DeploymentPreference`設定將套用至使用指定的別名`AutoPublishAlias`。

當您指定時`DeploymentPreference`， AWS SAM 會自動產生`StateMachineVersionArn`子性質值。

類型:DeploymentPreference

必要：否

AWS CloudFormation 兼容性： AWS SAM 生成`StateMachineVersionArn`屬性值並將其附加到資源的屬性，`DeploymentPreference`並將其傳遞`DeploymentPreference`給`AWS::StepFunctions::StateMachineAlias`資源的`DeploymentPreference`屬性。

`Events`

指定觸發此狀態機器的事件。事件包含一個類型和一組依賴於類型的屬性。

類型:EventSource

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Logging`

定義要記錄哪些執行歷程記錄事件以及記錄位置。

類型:LoggingConfiguration

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::StepFunctions::StateMachine`資源的`LoggingConfiguration`屬性。

`Name`

狀態機器的名稱。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::StepFunctions::StateMachine`資源的`StateMachineName`屬性。

`PermissionsBoundary`

權限界限的 ARN，用於此狀態機器的執行角色。只有在為您產生角色時，此屬性才有效。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::IAM::Role`資源的`PermissionsBoundary`屬性。

`Policies`

此狀態機的權限原則。政策將附加到狀態機器的預設 AWS Identity and Access Management (IAM) 執行角色。

此屬性接受單一值或值清單。允許數值包括：

- AWS SAM策略範本.
- AWS 受管理策略或客戶管理策略ARN的。
- 下列清單中 AWS 受管理策略的名稱。
- 格式化YAML為地圖的內嵌 IAM 政策。

> **Note**
>
> 如果您設定`Role`屬性，則會忽略此屬性。

類型：字串 | 清單 | 地圖

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`PropagateTags`

指出是否要將標籤從`Tags`屬性傳遞至您AWS::Serverless::StateMachine產生的資源。指`True`定在產生的資源中傳播標籤。

類型：布林值

必要：否

預設：`False`

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Role`

IAM 角色的 ARN，用作此狀態機器的執行角色。

類型：字串

必要：有條件

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::StepFunctions::StateMachine`資源的 `RoleArn`屬性。

`RolePath`

狀態機器 IAM 執行角色的路徑。

為您產生角色時，請使用此屬性。當角色與`Role`屬性一起指定時，請勿使用。

類型：字串

必要：有條件

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::IAM::Role`資源的`Path`屬性。

`Tags`

string-to-string 指定新增至狀態機器的標籤以及對應執行角色的對應。有關標籤的有效鍵和值的詳細資訊，請參閱AWS::StepFunctions::StateMachine資源的 Tags 屬性。

類型：地圖

必要：否

AWS CloudFormation 兼容性：此屬性類似於AWS::StepFunctions::StateMachine資源的Tags屬性。 AWS SAM 會自動將stateMachine:createdBy:SAM標籤新增至此資源，以及為其產生的預設角色。

Tracing

選擇 AWS X-Ray 是否為狀態機啟用。如需有關將 X-Ray 與 Step Functions 搭配使用的詳細資訊 AWS X-Ray ，請參閱AWS Step Functions 開發人員指南中的和 Step Functions

類型:TracingConfiguration

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::StepFunctions::StateMachine資源的TracingConfiguration屬性。

Type

狀態機的類型。

有效值：STANDARD 或 EXPRESS

類型：字串

必要：否

預設：STANDARD

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::StepFunctions::StateMachine資源的StateMachineType屬性。

傳回值

Ref

當您將此資源的邏輯 ID 提供給 Ref 內建函數時，Ref 會傳回基礎資源的 Amazon 資源名稱 (ARN)。AWS::StepFunctions::StateMachine

若要取得有關使用Ref功能的更多資訊，請參閱《使AWS CloudFormation 用指南》*Ref*中的〈〉

Fn: GetAtt

Fn::GetAtt 會傳回此類型之指定屬性的值。以下為可用屬性及傳回值的範例。

若要取得有關使用的更多資訊Fn::GetAtt，請參閱使AWS CloudFormation 用指南*Fn::GetAtt*中的〈〉

Name

返回狀態機的名稱，例如HelloWorld-StateMachine。

範例

狀態機定義文件

以下是允許 lambda 函數調用狀態機器的內聯狀態機定義的示例。請注意，此範例會預期內Role容設定適當的原則以允許呼叫。檔my_state_machine.asl.json案必須以 Amazon 州語言撰寫。

在此範例中，DefinitionSubstitution項目允許狀態機器包含 AWS SAM 範本檔案中宣告的資源。

YAML

```
MySampleStateMachine:
  Type: AWS::Serverless::StateMachine
  Properties:
    DefinitionUri: statemachine/my_state_machine.asl.json
    Role: arn:aws:iam::123456123456:role/service-role/my-sample-role
    Tracing:
      Enabled: true
    DefinitionSubstitutions:
      MyFunctionArn: !GetAtt MyFunction.Arn
      MyDDBTable: !Ref TransactionTable
```

內聯狀態機定義

以下是內嵌狀態機定義的範例。

在此範例中，範 AWS SAM 本檔案是以 YAML 撰寫，因此狀態機器定義也在 YAML 中。若要在 JSON 中宣告內嵌狀態機器定義，請以 JSON 撰寫 AWS SAM 範本檔案。

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

描述觸發狀態機器之事件來源的物件。每個事件都包含一個類型和一組依賴於該類型的屬性。如需有關每個事件來源屬性的詳細資訊，請參閱與該類型對應的副主題。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
Properties: Schedule | ScheduleV2 | CloudWatchEvent | EventBridgeRule | Api
Type: String
```

屬性

`Properties`

描述此事件對應屬性的物件。屬性集必須符合定義的`Type`。

類型：排程 | 排程 2 | CloudWatchEvent| API EventBridgeRule

必要：是

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Type`

事件類型。

有效值：`Api`、`Schedule`、`ScheduleV2`、`CloudWatchEvent`、`EventBridgeRule`

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

API

以下是`API`類型事件的範例。

YAML

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

Api

描述`Api`事件來源類型的物件。如果已定義AWS::Serverless::Api資源，路徑和方法值必須對應至 API OpenAPI 定義中的作業。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
  Auth: ApiStateMachineAuth
  Method: String
  Path: String
  RestApiId: String
  UnescapeMappingTemplate: Boolean
```

屬性

`Auth`

此 API、路徑和方法的授權設定。

如果未指定，請使用此屬性覆寫個別路徑`DefaultAuthorizer`的 API 設定，或覆寫預設`ApiKeyRequired`設定。`DefaultAuthorizer`

類型:ApiStateMachineAuth

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Method`

呼叫此函數的 HTTP 方法。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Path`

呼叫此函式的 URI 路徑。值必須以開頭`/`。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`RestApiId`

`RestApi`資源的標識符，其中必須包含具有給定路徑和方法的操作。一般而言，這會設定為參照此範本中定義的AWS::Serverless::Api資源。

如果您未定義此屬性，請使用產生的`OpenApi`文件 AWS SAM 建立預設AWS::Serverless::Api資源。該資源包含由不指定的相同範本中的`Api`事件所定義的所有路徑和方法的聯集`RestApiId`。

此屬性無法參考在其他範本中定義的AWS::Serverless::Api資源。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`UnescapeMappingTemplate`

通過替換\'為，在傳遞給狀態機的輸入上不轉義單引號。'當您的輸入包含單引號時使用。

> **Note**
>
> 如果設定為，`False`且您的輸入包含單引號，就會發生錯誤。

類型：布林值

必要：否

預設值：False

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

ApiEvent

以下是`Api`類型事件的範例。

YAML

```
Events:
  ApiEvent:
    Type: Api
    Properties:
      Path: /path
      Method: get
```

ApiStateMachineAuth

在事件層級設定特定 API、路徑和方法的授權。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
  ApiKeyRequired: Boolean
  AuthorizationScopes: List
  Authorizer: String
  ResourcePolicy: ResourcePolicyStatement
```

屬性

`ApiKeyRequired`

需要此 API、路徑和方法的 API 金鑰。

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`AuthorizationScopes`

要套用至此 API、路徑和方法的授權範圍。

如果您已指定屬性套用的任何範圍，您指定的範圍將覆寫該`DefaultAuthorizer`屬性套用的任何範圍。

類型：列表

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`Authorizer`

特`Authorizer`定狀態機器的。

如果您已為 API 指定全域授權者，並且想要將此狀態機器設為公用，請將設Authorizer定為覆寫全域授權者。NONE

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`ResourcePolicy`

設定此 API 和路徑的資源策略。

類型:ResourcePolicyStatement

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

StateMachine-身份驗證

下列範例會指定狀態機器層級的授權。

YAML

```
Auth:
  ApiKeyRequired: true
  Authorizer: NONE
```

ResourcePolicyStatement

為 API 的所有方法和路徑配置資源策略。如需有關資源政策的詳細資訊，請參閱《API Gateway 開發人員指南》中的使用 API Gateway 資源政策控制 API 的存取。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

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

屬性

`AwsAccountBlacklist`

要封鎖的 AWS 帳戶。

類型：字串的清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`AwsAccountWhitelist`

要允許的 AWS 帳戶。如需此屬性的使用範例，請參閱本頁底部的「範例」一節。

類型：字串的清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`CustomStatements`

要套用至此 API 的自訂資源政策陳述式清單。如需此屬性的使用範例，請參閱本頁底部的「範例」一節。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`IntrinsicVpcBlacklist`

要封鎖的虛擬私有雲端 (VPC) 清單，其中每個 VPC 都指定為參考，例如動態參考或Ref內建函數。如需此屬性的使用範例，請參閱本頁底部的「範例」一節。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`IntrinsicVpcWhitelist`

要允許的 VPC 清單，其中每個 VPC 都被指定為參考，例如動態參考或Ref內建函數。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`IntrinsicVpceBlacklist`

要封鎖的 VPC 端點清單，其中每個 VPC 端點都指定為參考，例如動態參考或Ref內建函數。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`IntrinsicVpceWhitelist`

要允許的 VPC 端點清單，其中每個 VPC 端點都指定為參考，例如動態參考或Ref內建函數。如需此屬性的使用範例，請參閱本頁底部的「範例」一節。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`IpRangeBlacklist`

要封鎖的 IP 位址或位址範圍。如需此屬性的使用範例，請參閱本頁底部的「範例」一節。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`IpRangeWhitelist`

要允許的 IP 位址或位址範圍。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`SourceVpcBlacklist`

要封鎖的來源 VPC 或 VPC 端點。來源 VPC 名稱必須以開頭，`"vpc-"`且來源 VPC 端點名稱必須以開頭。`"vpce-"`如需此屬性的使用範例，請參閱本頁底部的「範例」一節。

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`SourceVpcWhitelist`

要允許的來源 VPC 或 VPC 端點。來源 VPC 名稱必須以開頭，`"vpc-"`且來源 VPC 端點名稱必須以開頭。`"vpce-"`

類型：清單

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

資源策略範例

下列範例會封鎖兩個 IP 位址和一個來源 VPC，並允許 AWS 帳戶。

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`事件來源類型的物件。

AWS Serverless Application Model (AWS SAM) 在設定此事件類型時產生AWS::Events::Rule資源。

重要注意事項：EventBridgeRule是要使用的首選事件源類型，而不是CloudWatchEvent。EventBridgeRule並CloudWatchEvent使用相同的基礎服務，API 和 AWS CloudFormation 資源。但是，只 AWS SAM 會將對新功能的支援增加到EventBridgeRule。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
EventBusName: String
Input: String
InputPath: String
Pattern: EventPattern
```

屬性

EventBusName

與此規則相關聯的事件匯流排。如果您省略此屬性， AWS SAM 會使用預設事件匯流排。

類型：字串

必要：否

預設值：預設事件匯流排

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Events::Rule資源的EventBusName屬性。

Input

傳遞到目標的有效 JSON 文字。如果您使用此屬性，事件文字本身不會有任何內容傳遞到目標。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Events::Rule Target資源的Input屬性。

InputPath

如果您不想將整個匹配的事件傳遞給目標，請使用該InputPath屬性來描述要傳遞的事件的哪個部分。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Events::Rule Target`資源的`InputPath`屬性。

`Pattern`

說明哪些事件會路由到指定目標。如需詳細資訊，請參閱 Amazon EventBridge 使用者指南 EventBridge中的事件和事件模式。

類型:EventPattern

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Events::Rule`資源的`EventPattern`屬性。

範例

CloudWatchEvent

以下是`CloudWatchEvent`事件來源類型的範例。

YAML

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

EventBridgeRule

描述`EventBridgeRule`事件來源類型的物件，可將狀態機器設定為 Amazon EventBridge 規則的目標。有關更多信息，請參閱什麼是 Amazon EventBridge？ 在 Amazon 用 EventBridge 戶指南。

AWS SAM 設置此事件類型時生成一個AWS::Events::Rule資源。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
DeadLetterConfig: DeadLetterConfig
EventBusName: String
Input: String
InputPath: String
InputTransformer: InputTransformer
Pattern: EventPattern
RetryPolicy: RetryPolicy
RuleName: String
State: String
Target: Target
```

屬性

`DeadLetterConfig`

設定 Amazon Simple Queue Service (Amazon SQS) 佇列，在目標叫用失敗後 EventBridge 傳送事件。例如，當將事件傳送至不存在的 Lambda 函數，或是沒 EventBridge 有足夠的權限無法呼叫 Lambda 函數時，呼叫可能會失敗。如需詳細資訊，請參閱 Amazon 使用 EventBridge 者指南中的事件重試政策和使用無效字母佇列。

類型:DeadLetterConfig

必要：否

AWS CloudFormation 兼容性：此屬性類似於`AWS::Events::RuleTarget`數據類型的`DeadLetterConfig`屬性。如果您想要 AWS SAM 為您建立無效字母佇列，則此屬性的 AWS SAM 版本包含其他子屬性。

`EventBusName`

與此規則相關聯的事件匯流排。如果您省略此屬性， AWS SAM 會使用預設事件匯流排。

類型：字串

必要：否

預設值：預設事件匯流排

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Events::Rule資源的EventBusName屬性。

Input

傳遞到目標的有效 JSON 文字。如果您使用此屬性，事件文字本身不會有任何內容傳遞到目標。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Events::Rule Target資源的Input屬性。

InputPath

如果您不想將整個匹配的事件傳遞給目標，請使用該InputPath屬性來描述要傳遞的事件的哪個部分。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Events::Rule Target資源的InputPath屬性。

InputTransformer

此設定能讓您以特定事件資料為基礎，向目標提供自訂輸入。您可從事件擷取一或多組鍵/值對，然後使用該資料將自訂輸入傳送至目標。如需詳細資訊，請參閱 Amazon EventBridge 使用者指南中的 Amazon EventBridge 輸入轉換。

類型:InputTransformer

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Events::RuleTarget數據類型的InputTransformer 屬性。

Pattern

說明哪些事件會路由到指定目標。如需詳細資訊，請參閱 Amazon EventBridge 使用者指南 EventBridge中的事件和事件模式。

類型:EventPattern

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Events::Rule資源的EventPattern屬性。

RetryPolicy

包含重試政策設定相關資訊的 RetryPolicy 物件。如需詳細資訊，請參閱 Amazon 使用 EventBridge 者指南中的事件重試政策和使用無效字母佇列。

類型:RetryPolicy

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Events::RuleTarget數據類型的RetryPolicy屬性。

RuleName

規則的名稱。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Events::Rule資源的Name屬性。

State

規則的狀態。

有效值：[DISABLED | ENABLED]

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Events::Rule資源的State屬性。

Target

觸發規則時 EventBridge 呼叫的 AWS 資源。您可以使用此屬性來指定目標的邏輯 ID。如果未指定此屬性，則 AWS SAM 會產生目標的邏輯 ID。

類型：Target

必要：否

AWS CloudFormation 兼容性：此屬性類似於AWS::Events::Rule資源的Targets屬性。此屬性的 AWS SAM 版本只允許您指定單一目標的邏輯 ID。

範例

EventBridgeRule

以下是EventBridgeRule事件來源類型的範例。

YAML

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

DeadLetterConfig

用於指定 Amazon Simple Queue Service (Amazon SQS) 佇列的物件，在目標叫用失敗後 EventBridge 傳送事件。例如，當將事件傳送至不存在的狀態機器時，或是呼叫狀態機器的權限不足時，呼叫可能會失敗。如需詳細資訊，請參閱 Amazon 使用 EventBridge 者指南中的事件重試政策和使用無效字母佇列。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
  Arn: String
  QueueLogicalId: String
  Type: String
```

屬性

Arn

Amazon SQS 佇列的亞馬遜資源名稱 (ARN) 指定為無效字母佇列的目標。

> **Note**
>
> 指定Type性質或Arn性質，但不能同時指定兩者。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Events::RuleDeadLetterConfig數據類型的Arn屬性。

QueueLogicalId

如果指定，則 AWS SAM Type創建無效字母隊列的自定義名稱。

> **Note**
>
> 如果未設定Type屬性，則會忽略此屬性。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

Type

佇列的類型。設定此屬性時， AWS SAM 會自動建立無效字母佇列，並附加必要的資源型政策，以授與規則資源的權限，以便將事件傳送至佇列。

> **Note**
>
> 指定Type性質或Arn性質，但不能同時指定兩者。

有效值：SQS

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

DeadLetterConfig

DeadLetterConfig

YAML

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

Target

配置觸發規則時 EventBridge 調用的 AWS 資源。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
  Id: String
```

屬性

`Id`

目標的邏輯識別碼。

的值`Id`可以包括英數字元、句號 (`.`)、連字號 (`-`) 和底線 (`_`)。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Events::RuleTarget`數據類型的`Id`屬性。

範例

目標

YAML

```
EBRule:
  Type: EventBridgeRule
  Properties:
    Target:
      Id: MyTarget
```

Schedule

描述Schedule事件來源類型的物件，可將您的狀態機器設定為依排程觸發之 EventBridge 規則的目標。有關更多信息，請參閱什麼是 Amazon EventBridge？ 在 Amazon 用 EventBridge 戶指南。

AWS Serverless Application Model (AWS SAM) 在設定此事件類型時產生AWS::Events::Rule資源。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
  DeadLetterConfig: DeadLetterConfig
  Description: String
  Enabled: Boolean
  Input: String
  Name: String
  RetryPolicy: RetryPolicy
  RoleArn: String
  Schedule: String
  State: String
  Target: Target
```

屬性

`DeadLetterConfig`

設定 Amazon Simple Queue Service (Amazon SQS) 佇列，在目標叫用失敗後 EventBridge 傳送事件。例如，當將事件傳送至不存在的 Lambda 函數，或是沒 EventBridge 有足夠的權限無法呼叫

Lambda 函數時，呼叫可能會失敗。如需詳細資訊，請參閱 Amazon 使用 EventBridge 者指南中的事件重試政策和使用無效字母佇列。

類型:DeadLetterConfig

必要：否

AWS CloudFormation 兼容性：此屬性類似於AWS::Events::RuleTarget數據類型的DeadLetterConfig屬性。如果您想要 AWS SAM 為您建立無效字母佇列，則此屬性的 AWS SAM 版本包含其他子屬性。

`Description`

規則的描述。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Events::Rule資源的Description屬性。

`Enabled`

指出系統是否已啟用規則。

若要停用規則，請將此屬性設定為false。

> **Note**
>
> 指定Enabled或State性質，但不能同時指定兩者。

類型：布林值

必要：否

AWS CloudFormation 兼容性：此屬性類似於AWS::Events::Rule資源的State屬性。如果此屬性設定為，true則 AWS SAM 傳遞ENABLED，否則會傳遞DISABLED。

`Input`

傳遞到目標的有效 JSON 文字。如果您使用此屬性，事件文字本身不會有任何內容傳遞到目標。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Events::Rule Target資源的Input屬性。

Name

規則的名稱。如果未指定名稱， AWS CloudFormation 會產生唯一的實體 ID，並使用該 ID 做為規則名稱。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Events::Rule資源的Name屬性。

RetryPolicy

包含重試政策設定相關資訊的 RetryPolicy 物件。如需詳細資訊，請參閱 Amazon 使用 EventBridge 者指南中的事件重試政策和使用無效字母佇列。

類型:RetryPolicy

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Events::RuleTarget數據類型的RetryPolicy屬性。

RoleArn

呼叫排程時， EventBridge 排程器將用於目標的 IAM 角色 ARN。

類型:RoleArn

需要：否。 如果未提供，則會建立並使用新角色。

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Scheduler::ScheduleTarget數據類型的RoleArn屬性。

Schedule

判斷何時及執行規則頻率的排程表達式。如需詳細資訊，請參閱規則的排程運算式。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Events::Rule資源的ScheduleExpression屬性。

State

規則的狀態。

接受的值：DISABLED | ENABLED

> **Note**
>
> 指定Enabled或State性質，但不能同時指定兩者。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Events::Rule資源的State屬性。

Target

觸發規則時 EventBridge 呼叫的 AWS 資源。您可以使用此屬性來指定目標的邏輯 ID。如果未指定此屬性，則 AWS SAM 會產生目標的邏輯 ID。

類型：Target

必要：否

AWS CloudFormation 兼容性：此屬性類似於AWS::Events::Rule資源的Targets屬性。此屬性的 AWS SAM 版本只允許您指定單一目標的邏輯 ID。

範例

CloudWatch 排程活動

CloudWatch 排程事件範例

YAML

```
CWSchedule:
  Type: Schedule
  Properties:
```

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

DeadLetterConfig

用於指定 Amazon Simple Queue Service (Amazon SQS) 佇列的物件，在目標叫用失敗後 EventBridge 傳送事件。例如，當將事件傳送至不存在的狀態機器時，或是呼叫狀態機器的權限不足時，呼叫可能會失敗。如需詳細資訊，請參閱 Amazon 使用 EventBridge 者指南中的事件重試政策和使用無效字母佇列。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
Arn: String
QueueLogicalId: String
Type: String
```

屬性

`Arn`

Amazon SQS 佇列的亞馬遜資源名稱 (ARN) 指定為無效字母佇列的目標。

> **Note**
>
> 指定`Type`性質或`Arn`性質，但不能同時指定兩者。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Events::RuleDeadLetterConfig`數據類型的`Arn`屬性。

`QueueLogicalId`

如果指定，則 AWS SAM `Type`創建無效字母隊列的自定義名稱。

> **Note**
>
> 如果未設定Type屬性，則會忽略此屬性。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

Type

佇列的類型。設定此屬性時， AWS SAM 會自動建立無效字母佇列，並附加必要的資源型政策，以授與規則資源的權限，以便將事件傳送至佇列。

> **Note**
>
> 指定Type性質或Arn性質，但不能同時指定兩者。

有效值：SQS

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

範例

DeadLetterConfig

DeadLetterConfig

YAML

```
DeadLetterConfig:
```

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

Target

設定觸發規則時 EventBridge 呼叫的 AWS 資源。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
  Id: String
```

屬性

Id

目標的邏輯識別碼。

的值Id可以包括英數字元、句號 (.)、連字號 (-) 和底線 (_)。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Events::RuleTarget數據類型的Id屬性。

範例

目標

YAML

```
EBRule:
  Type: Schedule
  Properties:
    Target:
      Id: MyTarget
```

ScheduleV2

描述ScheduleV2事件來源類型的物件，可將狀態機器設定為按排程觸發之 Amazon EventBridge Scheduler 事件的目標。如需詳細資訊，請參閱什麼是 Amazon EventBridge 排程器？ 在「EventBridge 排程器使用指南」中。

AWS Serverless Application Model (AWS SAM) 在設定此事件類型時產生AWS::Scheduler::Schedule資源。

語法

若要在 AWS Serverless Application Model (AWS SAM) 範本中宣告此實體，請使用下列語法。

YAML

```
DeadLetterConfig: DeadLetterConfig
Description: String
EndDate: String
FlexibleTimeWindow: FlexibleTimeWindow
GroupName: String
Input: String
KmsKeyArn: String
Name: String
OmitName: Boolean
PermissionsBoundary: String
RetryPolicy: RetryPolicy
RoleArn: String
ScheduleExpression: String
ScheduleExpressionTimezone: String
StartDate: String
State: String
```

屬性

`DeadLetterConfig`

設定 Amazon Simple Queue Service (Amazon SQS) 佇列，在目標叫用失敗後 EventBridge 傳送事件。例如，當將事件傳送至不存在的 Lambda 函數，或是沒 EventBridge 有足夠的權限無法呼叫 Lambda 函數時，呼叫可能會失敗。如需詳細資訊，請參閱《排程器使用指EventBridge 南》中的 < 設定 EventBridge 排程器的無效字母佇列 > 。

類型:DeadLetterConfig

必要：否

AWS CloudFormation 兼容性：此屬性類似於AWS::Scheduler::ScheduleTarget數據類型的DeadLetterConfig屬性。如果您想要 AWS SAM 為您建立無效字母佇列，則此屬性的 AWS SAM 版本包含其他子屬性。

Description

排程的描述。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Scheduler::Schedule資源的Description屬性。

EndDate

UTC 日期，排程可在此日期之前叫用其目標。視排程的週期運算式而定，叫用可能會在您指定的 EndDate 當天或之前停止。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Scheduler::Schedule資源的EndDate屬性。

FlexibleTimeWindow

允許設定可在其中呼叫排程的視窗。

類型:FlexibleTimeWindow

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Scheduler::Schedule資源的FlexibleTimeWindow屬性。

GroupName

要與此排程產生關聯的排程群組名稱。如果未定義，則使用預設群組。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Scheduler::Schedule資源的GroupName屬性。

Input

傳遞到目標的有效 JSON 文字。如果您使用此屬性，事件文字本身不會有任何內容傳遞到目標。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Scheduler::Schedule Target資源的Input屬性。

KmsKeyArn

將用於加密客戶資料的 KMS 金鑰的 ARN。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Scheduler::Schedule資源的KmsKeyArn屬性。

Name

排程的名稱。如果未指定名稱，則會以格式 AWS SAM 產生名稱，*StateMachine-Logical-IDEvent-Source-Name*並使用該 ID 做為排程名稱。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Scheduler::Schedule資源的Name屬性。

OmitName

依預設， AWS SAM 會產生並使用格式為 *<S tate-machine-logical-ID event-source-name >< >* 的排程名稱。將此性質設定true為以 AWS CloudFormation 產生唯一的實體 ID，並將其用作明細表名稱。

類型：布林值

必要：否

預設：`false`

AWS CloudFormation 兼容性：此屬性是唯一的， AWS SAM 並且沒有相 AWS CloudFormation 等的屬性。

`PermissionsBoundary`

用來設定角色許可邊界的政策 ARN。

> **Note**
>
> 如果`PermissionsBoundary`已定義，則 AWS SAM 會將相同的界限套用至排程器排程的目標 IAM 角色。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::IAM::Role`資源的`PermissionsBoundary`屬性。

`RetryPolicy`

包含重試政策設定相關資訊的 `RetryPolicy` 物件。

類型:RetryPolicy

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給`AWS::Scheduler::ScheduleTarget`數據類型的`RetryPolicy`屬性。

`RoleArn`

呼叫排程時， EventBridge 排程器將用於目標的 IAM 角色 ARN。

類型:RoleArn

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Scheduler::ScheduleTarget數據類型的RoleArn屬性。

ScheduleExpression

決定排程執行時間和頻率的排程運算式。

類型：字串

必要：是

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Scheduler::Schedule資源的ScheduleExpression屬性。

ScheduleExpressionTimezone

計算排程運算式所使用的時區。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Scheduler::Schedule資源的ScheduleExpressionTimezone屬性。

StartDate

日期 (以 UTC 為單位)，在此日期之後，排程就可以開始叫用目標。視排程的週期運算式而定，叫用可能會在您指定的 StartDate 當天或之後發生。

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Scheduler::Schedule資源的StartDate屬性。

State

排程的狀態。

接受的值：DISABLED | ENABLED

類型：字串

必要：否

AWS CloudFormation 兼容性：此屬性直接傳遞給AWS::Scheduler::Schedule資源的State屬性。

範例

定義排程 2 資源的基本範例

```
StateMachine:
  Type: AWS::Serverless::StateMachine
  Properties:
    Name: MyStateMachine
    Events:
      ScheduleEvent:
        Type: ScheduleV2
        Properties:
          ScheduleExpression: "rate(1 minute)"
      ComplexScheduleEvent:
        Type: ScheduleV2
        Properties:
          ScheduleExpression: rate(1 minute)
          FlexibleTimeWindow:
            Mode: FLEXIBLE
            MaximumWindowInMinutes: 5
          StartDate: '2022-12-28T12:00:00.000Z'
          EndDate: '2023-01-28T12:00:00.000Z'
          ScheduleExpressionTimezone: UTC
          RetryPolicy:
            MaximumRetryAttempts: 5
            MaximumEventAgeInSeconds: 300
          DeadLetterConfig:
            Type: SQS
    DefinitionUri:
      Bucket: sam-demo-bucket
      Key: my-state-machine.asl.json
      Version: 3
    Policies:
      - LambdaInvokePolicy:
          FunctionName: !Ref MyFunction
```

產生的 AWS CloudFormation 資源

本節提供 AWS SAM 處理 AWS 範本時所建立之 AWS CloudFormation 資源的詳細資訊。根據您指定的案例， AWS SAM 產生的 AWS CloudFormation 資源集會有所不同。案例是模板文件中指定的 AWS SAM 資源和屬性的組合。您可以在模板文件中的其他位置引用生成 AWS CloudFormation 的資源，類似於引用在模板文件中明確聲明的資源的方式。

例如，如果您在 AWS SAM 範本檔案中指定`AWS::Serverless::Function`資源，則 AWS SAM 永遠會產生`AWS::Lambda::Function`基底資源。如果您還指定了可選`AutoPublishAlias`屬性，則 AWS SAM 另外生成`AWS::Lambda::Alias`和`AWS::Lambda::Version`資源。

本節列出案例及其產生的 AWS CloudFormation 資源，並顯示如何參考 AWS SAM 範本檔案中產生的 AWS CloudFormation 資源。

引用生成的 AWS CloudFormation 資源

您有兩個選項可用來參考 AWS SAM 範本檔案中產生的 AWS CloudFormation 資源，依照`LogicalId`或可參照的屬性。

引用生成的 AWS CloudFormation 資源 LogicalId

每個 AWS CloudFormation 資源都 AWS SAM 有一個`LogicalId`，它是一個字母數字（A-Z，a-z，0-9）標識符，在模板文件中是唯一的。 AWS SAM 使用`LogicalIds`範本檔案中的 AWS SAM 資源來建構其產生`LogicalIds`的 AWS CloudFormation 資源。您可以使用產生`LogicalId`的 AWS CloudFormation 資源在範本檔案中存取該資源的屬性，就像您對已明確宣告的 AWS CloudFormation 資源一樣。如需`LogicalIds`中 AWS CloudFormation 和 AWS SAM 範本的詳細資訊，請參閱AWS CloudFormation 使用指南中的資源。

> **Note**
>
> 某些產生`LogicalIds`的資源包含唯一的雜湊值，以避免命名空間衝突。建立堆疊時會衍生這些資源。`LogicalIds`您只能在使用 AWS Management Console、 AWS CLI或其中一個 AWS SDK 建立堆疊之後擷取它們。我們不建議您參考這些資源，`LogicalId`因為雜湊值可能會變更。

通過可引用屬性引用生成的 AWS CloudFormation 資源

對於某些產生的資源， AWS SAM 提供資源的可參照屬性。 AWS SAM 您可以使用此屬性來參考 AWS SAM 範本檔案中產生的 AWS CloudFormation 資源及其屬性。

> **Note**
>
> 並非所有產生的 AWS CloudFormation 資源都具有可參照的屬性。對於這些資源，您必須使用`LogicalId`.

產生的 AWS CloudFormation 資源案例

下表摘要說明組成產生 AWS SAM 資源之案例的 AWS CloudFormation 資源和屬性。「案例」欄中的主題提供針對該案例所 AWS SAM 產生之其他 AWS CloudFormation 資源的詳細資訊。

AWS SAM 資源	基本 AWS CloudFormation 資源	案例
`AWS::Serverless::Api`	`AWS::ApiGateway::RestApi`	• DomainName屬性已指定 • UsagePlan屬性已指定
`AWS::Serverless::Application`	`AWS::CloudFormation::Stack`	• 除了產生基礎 AWS CloudFormation 資源之外，此無伺服器資源沒有其他案例。
`AWS::Serverless::Function`	`AWS::Lambda::Function`	• AutoPublishAlias 屬性已指定 • 未指定角色屬性 • DeploymentPreference 屬性已指定 • 已指定 Api 事件來源 • 已指定 HttpApi事件來源 • 已指定串流事件來源 • 已指定事件橋接器 (或事件匯流排) 事件來源

AWS SAM 資源	基本 AWS CloudFormation 資源	案例
		• 已指定 IotRule事件來源 • OnSuccess(或 OnFailure) 屬性已針對 Amazon SNS 事件指定 • OnSuccess(或 OnFailure) 屬性是針對 Amazon SQS 事件指定的
AWS::Serverless::HttpApi	AWS::ApiGatewayV2::Api	• StageName屬性已指定 • StageName未指定屬性 • DomainName屬性已指定
AWS::Serverless::LayerVersion	AWS::Lambda::LayerVersion	• 除了產生基礎 AWS CloudFormation 資源之外，此無伺服器資源沒有其他案例。
AWS::Serverless::SimpleTable	AWS::DynamoDB::Table	• 除了產生基礎 AWS CloudFormation 資源之外，此無伺服器資源沒有其他案例。
AWS::Serverless::StateMachine	AWS::StepFunctions::StateMachine	• 未指定角色屬性 • 已指定 API 事件來源 • 已指定事件橋接器 (或事件匯流排) 事件來源

主題

- AWS CloudFormation 指定時產生AWS::Serverless::Api的資源

- AWS CloudFormation 指定時產生AWS::Serverless::Application的資源
- AWS CloudFormation指定時產生的資源 AWS::Serverless::Connector
- AWS CloudFormation 指定時產生AWS::Serverless::Function的資源
- AWS CloudFormation 指定時產生AWS::Serverless::GraphQLApi的資源
- AWS CloudFormation 指定時產生 AWS::Serverless::HttpApi 的資源
- AWS CloudFormation 指定時產生AWS::Serverless::LayerVersion的資源
- AWS CloudFormation 指定時產生AWS::Serverless::SimpleTable的資源
- AWS CloudFormation 指定時產生AWS::Serverless::StateMachine的資源

AWS CloudFormation 指定時產生AWS::Serverless::Api的資源

指定`AWS::Serverless::Api`時， AWS Serverless Application Model (AWS SAM) 一律會產生`AWS::ApiGateway::RestApi`基底 AWS CloudFormation 資源。此外，它還始終生成一個`AWS::ApiGateway::Stage`和一個`AWS::ApiGateway::Deployment`資源。

`AWS::ApiGateway::RestApi`

`LogicalId`: *`<api-LogicalId>`*

可參考屬性：N/A (您必須使用*`LogicalId`*來參照此 AWS CloudFormation 資源)

`AWS::ApiGateway::Stage`

`LogicalId`: *`<api-LogicalId><stage-name>`*`Stage`

*`<stage-name>`*是屬`StageName`性設定的字串。例如，如果您設`StageName`定為`Gamma`，則`LogicalId`是*`MyRestApiGamma`*`Stage`。

可引用的屬性：*`<api-LogicalId>`*`.Stage`

`AWS::ApiGateway::Deployment`

`LogicalId`: *`<api-LogicalId>`*`Deployment`*`<sha>`*

*`<sha>`*是建立堆疊時產生的唯一雜湊值。例如 *`MyRestApi`*`Deployment`*`926eeb5ff1`*。

可引用的屬性：*`<api-LogicalId>`*`.Deployment`

除了這些資 AWS CloudFormation 源之外，如果`AWS::Serverless::Api`有指定，則會針對下列案例 AWS SAM 產生其他 AWS CloudFormation 資源。

案例

- DomainName屬性已指定
- UsagePlan屬性已指定

DomainName屬性已指定

當指定的DomainName屬Domain性的屬性AWS::Serverless::Api時， AWS SAM 會產生資AWS::ApiGateway::DomainName AWS CloudFormation 源。

AWS::ApiGateway::DomainName

LogicalId: ApiGatewayDomainName*<sha>*

*<sha>*是建立堆疊時產生的唯一雜湊值。例如：ApiGatewayDomainName*926eeb5ff1*。

可引用的屬性：*<api-LogicalId>*.DomainName

UsagePlan屬性已指定

指定UsagePlan屬性的Auth屬性時AWS::Serverless::Api， AWS SAM 會產生下列 AWS CloudFormation 資源：AWS::ApiGateway::UsagePlanAWS::ApiGateway::UsagePlanKey、和AWS::ApiGateway::ApiKey。

AWS::ApiGateway::UsagePlan

LogicalId: *<api-LogicalId>*UsagePlan

可引用的屬性：*<api-LogicalId>*.UsagePlan

AWS::ApiGateway::UsagePlanKey

LogicalId: *<api-LogicalId>*UsagePlanKey

可引用的屬性：*<api-LogicalId>*.UsagePlanKey

AWS::ApiGateway::ApiKey

LogicalId: *<api-LogicalId>*ApiKey

可引用的屬性：*<api-LogicalId>*.ApiKey

AWS CloudFormation 指定時產生AWS::Serverless::Application的資源

指定`AWS::Serverless::Application`時， AWS Serverless Application Model (AWS SAM) 會產生`AWS::CloudFormation::Stack`基底 AWS CloudFormation 資源。

`AWS::CloudFormation::Stack`

`LogicalId`: *`<application-LogicalId>`*

可參考屬性：N/A (您必須使用*`LogicalId`*來參照此 AWS CloudFormation 資源)

AWS CloudFormation指定時產生的資源 AWS::Serverless::Connector

> **Note**
>
> 當您透過內嵌`Connectors`屬性定義連接器時，會先將其轉換為`AWS::Serverless::Connector`資源，然後再產生這些資源。

當您在AWS SAM範本中指定`AWS::Serverless::Connector`資源時，AWS SAM會視需要產生下列AWS CloudFormation資源。

`AWS::IAM::ManagedPolicy`

`LogicalId`:*`<connector-LogicalId>`*`Policy`

可參考屬性：N/A (若要參照此AWS CloudFormation資源，您必須使用.) `LogicalId`

`AWS::SNS::TopicPolicy`

`LogicalId`:*`<connector-LogicalId>`*`TopicPolicy`

可參考屬性：N/A (若要參照此AWS CloudFormation資源，您必須使用.) `LogicalId`

`AWS::SQS::QueuePolicy`

`LogicalId`:*`<connector-LogicalId>`*`QueuePolicy`

可參考屬性：N/A (若要參照此AWS CloudFormation資源，您必須使用.) `LogicalId`

`AWS::Lambda::Permission`

`LogicalId`:*`<connector-LogicalId><permission>`*`LambdaPermission`

*<permission>*是屬性指定的Permissions權限。例如 Write。

可參考屬性：N/A (若要參照此AWS CloudFormation資源，您必須使用.) LogicalId

AWS CloudFormation 指定時產生AWS::Serverless::Function的資源

指定AWS::Serverless::Function時， AWS Serverless Application Model (AWS SAM) 一律會建立AWS::Lambda::Function基底 AWS CloudFormation 資源。

AWS::Lambda::Function

LogicalId: *<function-LogicalId>*

可參考屬性：N/A (您必須使用*LogicalId*來參照此 AWS CloudFormation 資源)

除了此資 AWS CloudFormation 源之外，如果AWS::Serverless::Function有指定， AWS SAM 也會針對下列案例產生 AWS CloudFormation 資源。

案例

- AutoPublishAlias 屬性已指定
- 未指定角色屬性
- DeploymentPreference 屬性已指定
- 已指定 Api 事件來源
- 已指定 HttpApi事件來源
- 已指定串流事件來源
- 已指定事件橋接器 (或事件匯流排) 事件來源
- 已指定 IotRule事件來源
- OnSuccess(或 OnFailure) 屬性已針對 Amazon SNS 事件指定
- OnSuccess(或 OnFailure) 屬性是針對 Amazon SQS 事件指定的

AutoPublishAlias 屬性已指定

指定的AutoPublishAlias屬性時， AWS SAM 會產生下列 AWS CloudFormation 資源：AWS::Lambda::Alias和AWS::Lambda::Version。AWS::Serverless::Function

AWS::Lambda::Alias

LogicalId: *<function-LogicalId>*Alias*<alias-name>*

*<alias-name>*是設定為的字串。AutoPublishAlias例如，如果您設定AutoPublishAlias為live，則LogicalId為：*##MyFunction*別名。

可引用的屬性：*<function-LogicalId>*.Alias

AWS::Lambda::Version

LogicalId: *<function-LogicalId>*Version*<sha>*

*<sha>*是建立堆疊時產生的唯一雜湊值。例如，*MyFunction*版本 *926* 電子版本 1。

可引用的屬性：*<function-LogicalId>*.Version

未指定角色屬性

未指定的Role屬AWS::Serverless::Function性時， AWS SAM 會產生資AWS::IAM::Role AWS CloudFormation 源。

AWS::IAM::Role

LogicalId: *<function-LogicalId>*Role

可參考屬性：N/A (您必須使用*LogicalId*來參照此 AWS CloudFormation 資源)

DeploymentPreference 屬性已指定

指定的DeploymentPreference屬性時， AWS SAM 會產生下列資源 AWS CloudFormation 資源：AWS::CodeDeploy::Application和AWS::CodeDeploy::DeploymentGroup。AWS::Serverles外，如果未指定DeploymentPreference物件的Role屬性， AWS SAM 也會產生資AWS::IAM::Role AWS CloudFormation 源。

AWS::CodeDeploy::Application

LogicalId: ServerlessDeploymentApplication

可參考屬性：N/A (您必須使用*LogicalId*來參照此 AWS CloudFormation 資源)

AWS::CodeDeploy::DeploymentGroup

LogicalId: *<function-LogicalId>*DeploymentGroup

可參考屬性：N/A (您必須使用*LogicalId*來參照此 AWS CloudFormation 資源)

AWS::IAM::Role

LogicalId: CodeDeployServiceRole

可參考屬性：N/A (您必須使用*LogicalId*來參照此 AWS CloudFormation 資源)

已指定 Api 事件來源

當 a 的Event屬性設定為Api，但未指定RestApiId屬性時， AWS SAM 會產生資AWS::ApiGateway::RestApi AWS CloudFormation 源。AWS::Serverless::Function

AWS::ApiGateway::RestApi

LogicalId: ServerlessRestApi

可參考屬性：N/A (您必須使用*LogicalId*來參照此 AWS CloudFormation 資源)

已指定 HttpApi事件來源

當 a 的Event屬性設定為HttpApi，但未指定ApiId屬性時， AWS SAM 會產生資AWS::ApiGatewayV2::Api AWS CloudFormation 源。AWS::Serverless::Function

AWS::ApiGatewayV2::Api

LogicalId: ServerlessHttpApi

可參考屬性：N/A (您必須使用*LogicalId*來參照此 AWS CloudFormation 資源)

已指定串流事件來源

將 a 的Event屬性設定AWS::Serverless::Function為其中一個串流類型時， AWS SAM 會產生資AWS::Lambda::EventSourceMapping AWS CloudFormation 源。這適用於下列類型：DynamoDBKinesis、MQ、MSK、和SQS。

AWS::Lambda::EventSourceMapping

LogicalId: *<function-LogicalId><event-LogicalId>*

可參考屬性：N/A (您必須使用*LogicalId*來參照此 AWS CloudFormation 資源)

已指定事件橋接器 (或事件匯流排) 事件來源

將 a 的Event屬性設定AWS::Serverless::Function為其中一個事件橋接器 (或事件匯流排) 類型時， AWS SAM 會產生AWS::Events::Rule AWS CloudFormation 資源。這適用於下列類型：EventBridgeRuleSchedule、和CloudWatchEvents。

AWS::Events::Rule

LogicalId: *<function-LogicalId><event-LogicalId>*

可參考屬性：N/A (您必須使用*LogicalId*來參照此 AWS CloudFormation 資源)

已指定 IotRule事件來源

將的Event屬性設定AWS::Serverless::Function為 IOTrule 時， AWS SAM 會產生資源。AWS::IoT::TopicRule AWS CloudFormation

AWS::IoT::TopicRule

LogicalId: *<function-LogicalId><event-LogicalId>*

可參考屬性：N/A (您必須使用*LogicalId*來參照此 AWS CloudFormation 資源)

OnSuccess(或 OnFailure) 屬性已針對 Amazon SNS 事件指定

如果指定了屬性的 OnSuccess DestinationConfig (或OnFailure) EventInvokeConfig 屬性，且目的地類型為SNS但未指定目的地 ARN，則 AWS SAM 會產生下列 AWS CloudFormation 資源：AWS::Lambda::EventInvokeConfig和AWS::SNS::Topic。AWS::Serverless::Function

AWS::Lambda::EventInvokeConfig

LogicalId: *<function-LogicalId>*EventInvokeConfig

可參考屬性：N/A (您必須使用*LogicalId*來參照此 AWS CloudFormation 資源)

AWS::SNS::Topic

LogicalId：*<function-LogicalId>*OnSuccessTopic (或*<function-LogicalId>*OnFailureT

可引用的屬性：*<function-LogicalId>*.DestinationTopic

如果為 Amazon SNS 事件指定OnSuccess和OnFailure，為了區分產生的資源，您必須使用LogicalId.

OnSuccess(或 OnFailure) 屬性是針對 Amazon SQS 事件指定的

如果指定了屬性的 OnSuccess DestinationConfig (或OnFailure) EventInvokeConfig 屬性，且目的地類型為SQS但未指定目的地 ARN，則 AWS SAM 會產生下列 AWS CloudFormation 資源：AWS::Lambda::EventInvokeConfig和AWS::SQS::Queue。AWS::Serverless::Function

AWS::Lambda::EventInvokeConfig

LogicalId: *<function-LogicalId>*EventInvokeConfig

可參考屬性：N/A (您必須使用*LogicalId*來參照此 AWS CloudFormation 資源)

AWS::SQS::Queue

LogicalId：*<function-LogicalId>*OnSuccessQueue (或*<function-LogicalId>*OnFailureQ

可引用的屬性：*<function-LogicalId>*.DestinationQueue

如果為 Amazon SQS 事件指定OnSuccess和OnFailure，為了區分產生的資源，您必須使用LogicalId.

AWS CloudFormation 指定時產生AWS::Serverless::GraphQLApi的資源

當您在 AWS Serverless Application Model (AWS SAM) 範本中指定AWS::Serverless::GraphQLApi資源時，一 AWS SAM 律會建立下列基礎 AWS CloudFormation 資源。

AWS::AppSync::DataSource

LogicalId: *<graphqlapi-LogicalId><datasource-RelativeId><datasource-Type>*DataSource

可參考屬性：N/A (您必須使用*LogicalId*來參照此 AWS CloudFormation 資源)

AWS::AppSync::FunctionConfiguration

LogicalId: *<graphqlapi-LogicalId><function-RelativeId>*

可參考屬性：N/A (您必須使用*LogicalId*來參照此 AWS CloudFormation 資源)

AWS::AppSync::GraphQLApi

LogicalId: *<graphqlapi-LogicalId>*

可參考屬性：N/A (您必須使用*LogicalId*來參照此 AWS CloudFormation 資源)

AWS::AppSync::GraphQLSchema

LogicalId: *<graphqlapi-LogicalId>*Schema

可參考屬性：N/A (您必須使用*LogicalId*來參照此 AWS CloudFormation 資源)

AWS::AppSync::Resolver

LogicalId: *<graphqlapi-LogicalId><OperationType><resolver-RelativeId>*

可參考屬性：N/A (您必須使用*LogicalId*來參照此 AWS CloudFormation 資源)

除了這些資 AWS CloudFormation 源之外，如果AWS::Serverless::GraphQLApi有指定，也 AWS SAM 可能會產生下列 AWS CloudFormation 資源。

AWS::AppSync::ApiCache

LogicalId: *<graphqlapi-LogicalId>*ApiCache

可參考屬性：N/A (您必須使用*LogicalId*來參照此 AWS CloudFormation 資源)

AWS::AppSync::ApiKey

LogicalId: *<graphqlapi-LogicalId><apikey-RelativeId>*

可參考屬性：N/A (您必須使用*LogicalId*來參照此 AWS CloudFormation 資源)

AWS::AppSync::DomainName

LogicalId: *<graphqlapi-LogicalId>*DomainName

可參考屬性：N/A (您必須使用*LogicalId*來參照此 AWS CloudFormation 資源)

AWS::AppSync::DomainNameApiAssociation

LogicalId: *<graphqlapi-LogicalId>*DomainNameApiAssociation

可參考屬性：N/A (您必須使用*LogicalId*來參照此 AWS CloudFormation 資源)

AWS SAM 也可以使用資AWS::Serverless::Connector源來提供權限。如需更多詳細資訊，請參閱 AWS CloudFormation指定時產生的資源 AWS::Serverless::Connector。

AWS CloudFormation 指定時產生 AWS::Serverless::HttpApi 的資源

指定AWS::Serverless::HttpApi時， AWS Serverless Application Model (AWS SAM) 會產生AWS::ApiGatewayV2::Api基底 AWS CloudFormation 資源。

AWS::ApiGatewayV2::Api

LogicalId: *<httpapi-LogicalId>*

可參考屬性：N/A (您必須使用*LogicalId*來參照此 AWS CloudFormation 資源)

除了此資 AWS CloudFormation 源之外，如果AWS::Serverless::HttpApi有指定， AWS SAM 也會針對下列案例產生 AWS CloudFormation 資源：

案例

- StageName屬性已指定
- StageName未指定屬性
- DomainName屬性已指定

StageName屬性已指定

指定的StageName屬性時， AWS SAM 會產生資AWS::ApiGatewayV2::Stage AWS CloudFormation 源。AWS::Serverless::HttpApi

AWS::ApiGatewayV2::Stage

LogicalId: *<httpapi-LogicalId><stage-name>*Stage

*<stage-name>*是屬StageName性設定的字串。例如，如果您設定StageName為Gamma，則LogicalId為：*MyHttpApiGamma*舞台。

可引用的屬性：*<httpapi-LogicalId>*.Stage

StageName未指定屬性

未指定的StageName屬AWS::Serverless::HttpApi性時， AWS SAM 會產生資AWS::ApiGatewayV2::Stage AWS CloudFormation 源。

AWS::ApiGatewayV2::Stage

LogicalId: *<httpapi-LogicalId>*ApiGatewayDefaultStage

可引用的屬性：*<httpapi-LogicalId>*.Stage

DomainName屬性已指定

當指定的DomainName屬Domain性的屬性AWS::Serverless::HttpApi時， AWS SAM 會產生資AWS::ApiGatewayV2::DomainName AWS CloudFormation 源。

AWS::ApiGatewayV2::DomainName

LogicalId: ApiGatewayDomainNameV2*<sha>*

*<sha>*是建立堆疊時產生的唯一雜湊值。例如，ApiGatewayDomainNameV2*926* 電子郵件 1。

可引用的屬性：*<httpapi-LogicalId>*.DomainName

AWS CloudFormation 指定時產生AWS::Serverless::LayerVersion的資源

指定AWS::Serverless::LayerVersion時， AWS Serverless Application Model (AWS SAM) 會產生AWS::Lambda::LayerVersion基底 AWS CloudFormation 資源。

AWS::Lambda::LayerVersion

LogicalId: *<layerversion-LogicalId>*

可參考屬性：N/A (您必須使用*LogicalId*來參照此 AWS CloudFormation 資源)

AWS CloudFormation 指定時產生AWS::Serverless::SimpleTable的資源

指定AWS::Serverless::SimpleTable時， AWS Serverless Application Model (AWS SAM) 會產生AWS::DynamoDB::Table基底 AWS CloudFormation 資源。

AWS::DynamoDB::Table

LogicalId: *<simpletable-LogicalId>*

可參考屬性：N/A (您必須使用*LogicalId*來參照此 AWS CloudFormation 資源)

AWS CloudFormation 指定時產生AWS::Serverless::StateMachine的資源

指定`AWS::Serverless::StateMachine`時， AWS Serverless Application Model (AWS SAM) 會產生`AWS::StepFunctions::StateMachine`基底 AWS CloudFormation 資源。

`AWS::StepFunctions::StateMachine`

`LogicalId`: *`<statemachine-LogicalId>`*

可參考屬性：N/A (您必須使用*`LogicalId`*來參照此 AWS CloudFormation 資源)

除了此資 AWS CloudFormation 源之外，如果`AWS::Serverless::StateMachine`有指定， AWS SAM 也會針對下列案例產生 AWS CloudFormation 資源：

案例

- 未指定角色屬性
- 已指定 API 事件來源
- 已指定事件橋接器 (或事件匯流排) 事件來源

未指定角色屬性

未指定的`Role`屬`AWS::Serverless::StateMachine`性時， AWS SAM 會產生資`AWS::IAM::Role` AWS CloudFormation 源。

`AWS::IAM::Role`

`LogicalId`: *`<statemachine-LogicalId>`*`Role`

可參考屬性：N/A (您必須使用*`LogicalId`*來參照此 AWS CloudFormation 資源)

已指定 API 事件來源

當 a 的`Event`屬性設定為`Api`，但未指定`RestApiId`屬性時， AWS SAM 會產生資`AWS::ApiGateway::RestApi` AWS CloudFormation 源。`AWS::Serverless::StateMachine`

`AWS::ApiGateway::RestApi`

`LogicalId`: `ServerlessRestApi`

可參考屬性：N/A (您必須使用*LogicalId*來參照此 AWS CloudFormation 資源)

已指定事件橋接器 (或事件匯流排) 事件來源

將 a 的Event屬性設定AWS::Serverless::StateMachine為其中一個事件橋接器 (或事件匯流排) 類型時， AWS SAM 會產生AWS::Events::Rule AWS CloudFormation 資源。這適用於下列類型：EventBridgeRuleSchedule、和CloudWatchEvents。

AWS::Events::Rule

LogicalId: *<statemachine-LogicalId><event-LogicalId>*

可參考屬性：N/A (您必須使用*LogicalId*來參照此 AWS CloudFormation 資源)

支援的資源屬性 AWS SAM

資源屬性是您可以加入的屬性，以 AWS SAM 及用來控制其他行為和關係的 AWS CloudFormation 資源。如需有關資源屬性的詳細資訊，請參閱AWS CloudFormation 使用指南中的資源屬性參考。

AWS SAM 支援由定義的資源屬性子集 AWS CloudFormation。在支援的資源屬性中，有些只會複製到對應 AWS CloudFormation 資源的基礎產生 AWS SAM 資源，而有些則複製到所有產生的 AWS CloudFormation 資源，從對應的 AWS SAM 資源產生。如需有關從對應 AWS CloudFormation 資源產生之資源的詳細 AWS SAM 資訊，請參閱產生的 AWS CloudFormation 資源。

下表摘要說明資源屬性支援 AWS SAM，視以下例外狀況列出項目為準。

資源屬性	目標產生的資源
DependsOn 中繼資料 [1, 2]	僅基礎 AWS CloudFormation 產生的資源。如需有關資 AWS SAM 源與基底資源之間對映的 AWS CloudFormation 資訊，請參閱產生的 AWS CloudFormation 資源案例。
條件 DeletionPolicy UpdateReplacePolicy	來自對應 AWS CloudFormation 資源的所有產生 AWS SAM 資源。如需已產生 AWS CloudFormation 資源案例的相關資訊，請參閱產生的 AWS CloudFormation 資源案例。

備註：

1. 如需將 Metadata resource 屬性與資源類型搭配使用的詳細AWS::Serverless::Function資訊，請參閱使用自訂執行階段建置 Lambda 函數。
2. 如需將 Metadata resource 屬性與資源類型搭配使用的詳細AWS::Serverless::LayerVersion資訊，請參閱建置 Lambda 層。

例外狀況

先前描述的資源屬性規則有許多例外狀況：

- 對於AWS::Lambda::LayerVersion， AWS SAM唯一的自訂欄位會RetentionPolicyDeletionPolicy為所產生的 AWS CloudFormation 資源設定。這具有比DeletionPolicy自己更高的優先級。如果兩者都不設定，則依預設DeletionPolicy會設定為Retain。
- 對於AWS::Lambda::Version，如果DeletionPolicy未指定，則預設值為Retain。
- 針對為無伺服器函數指定的案例，資源屬性不會複製到下列產生的 AWS CloudFormation 資源：DeploymentPreferences
  - AWS::CodeDeploy::Application
  - AWS::CodeDeploy::DeploymentGroup
  - 為此案例建立的AWS::IAM::Role具CodeDeployServiceRole名
- 如果您的 AWS SAM 範本包含多個具有隱含建立之 API 事件來源的函數，則這些函數將共用產生的AWS::ApiGateway::RestApi資源。在此案例中，如果函數具有不同的資源屬性，則針對產生的AWS::ApiGateway::RestApi資源，根據下列優先順序清單 AWS SAM 複製資源屬性：
  - UpdateReplacePolicy:
    1. Retain
    2. Snapshot
    3. Delete
  - DeletionPolicy:
    1. Retain
    2. Delete

API Gateway 擴充功能

API Gateway 延伸功能專為設計 AWS，提供設計和管理 API 的額外自訂和功能。這些是 OpenAPI 規範的擴充功能，支援 API Gateway AWS特定的特定授權和 API 整合。

API Gateway 延伸模組是 OpenAPI 規格的延伸模組，可支援 AWS特定授權和 API 閘道特定的 API 整合。如需 API Gateway 延伸模組的詳細資訊，請參閱 OpenAPI 的 API Gateway 延伸模組。

AWS SAM 支援 API Gateway 延伸模組的子集。若要查看支援哪些 API Gateway 延伸模組 AWS SAM，請參閱下表。

API Gateway 擴充功能	支持 AWS SAM
x-amazon-apigateway-any-方法對象	是
x-amazon-apigateway-api-鍵源屬性	否
x-amazon-apigateway-auth 物件	是
x-amazon-apigateway-authorizer 物件	是
x-amazon-apigateway-authtype 屬性	是
x-amazon-apigateway-binary-媒體類型屬性	是
x-amazon-apigateway-documentation 物件	否
x-amazon-apigateway-endpoint-配置對象	否
x-amazon-apigateway-gateway-響應對象	是
x-amazon-apigateway-gateway-響應. 網關響應對象	是
x-amazon-apigateway-gateway-回應. 回應參數物件	是
x-amazon-apigateway-gateway-響應. 響應模板對象	是
x-amazon-apigateway-integration 物件	是
x-amazon-apigateway-integration。requestTemplates 對象	是
x-amazon-apigateway-integrationrequestParameters 對象	否

x-amazon-apigateway-integration. 回應對象	是
x-amazon-apigateway-integration. 響應對象	是
x-amazon-apigateway-integration. responseTemplates 物件	是
x-amazon-apigateway-integration. responseParameters 對象	是
x-amazon-apigateway-request-驗證屬性	否
x-amazon-apigateway-request-驗證對象	否
x-amazon-apigateway-request-驗證器. 請求驗證器對象	否

內部函數

內建函式是內建函式，可讓您將值指派給只能在執行階段使用的屬性。 AWS SAM 對某些內在函數屬性的支持有限，因此無法解決某些內在函數。因此，我們建議您新增`AWS::LanguageExtensions`轉換以解決此問題。這`AWS::LanguageExtensions`是一個由主控的巨集 AWS CloudFormation ，可讓您使用內建函式和其他未包含在中的預設功能。 AWS CloudFormation

```
Transform:
  - AWS::LanguageExtensions
  - AWS::Serverless-2016-10-31
```

> **Note**
>
> 注意：如果在 CodeUri 屬性中使用內置函數， AWS SAM 將無法正確解析值。請考慮`AWS::LanguageExtensions`改用轉換。
> 如需詳細資訊，請參閱的〈屬性〉一節 AWS::Serverless::Function。

如需有關內建函數的詳細資訊，請參閱《使用指南》中的內建函數參考。AWS CloudFormation

開發您的無伺服器應用程式 AWS SAM

本節包含有關驗證 AWS SAM 範本以及使用相依性建置應用程式的主題。它也包含用於特定使用案 AWS SAM 例的主題，例如使用 Lambda 層、使用巢狀應用程式、控制 API Gateway API 的存取、使用 Step Functions 協調 AWS 資源，以及程式碼簽章應用程式。下面列出了開發應用程序所需要完成的三個主要里程碑。

主題

- 使用sam init指令建立您的應用程式
- 定義您的基礎架構 AWS SAM
- 建置您的應用程式 AWS SAM

使用sam init指令建立您的應用程式

完成入門和閱讀後如何使用 AWS Serverless Application Model （ AWS SAM ），您將準備在開發人員環境中創建一個 AWS SAM 項目。您的 AWS SAM 專案將成為編寫無伺服器應用程式的起點。如需指 AWS SAMCLI`sam init`令選項的清單，請參閱sam init。

「指 AWS Serverless Application Model 令行介面」(AWS SAMCLI) `sam init` 指令提供用於初始化新的無伺服器應用程式的選項，該應用程式包括：

- 定義基礎結構程式碼的 AWS SAM 範本。
- 組織應用程式的資料夾結構。
- 為您的 AWS Lambda 功能配置。

若要建立 AWS SAM 專案，請參閱本節中的主題。

主題

- 初始化新的無伺服器應用程式
- 山姆初始化的選項
- 故障診斷
- 範例
- 進一步了解
- 後續步驟

初始化新的無伺服器應用程式

使用初始化新的無伺服器應用程式 AWS SAMCLI

1. cd到一個起始目錄。
2. 在命令行中運行以下命令：

```
$ sam init
```

3. AWS SAMCLI將引導您完成互動式流程，以建立新的無伺服器應用程式。

> **Note**
>
> 如中所述教學課程：部署 Hello World 應用程式，此命令會初始化您的無伺服器應用程式，並建立專案目錄。該目錄將包含多個文件和文件夾。最重要的文件是`template.yaml`. 這是您的 AWS SAM 範本。您的 python 版本必須與命sam init令創建的`template.yaml`文件中列出的 python 版本匹配。

選擇起始範本

範本包含下列項目：

1. 基礎架構程式碼的 AWS SAM 範本。
2. 組織專案檔案的起始專案目錄。例如，這可能包括：
   a. Lambda 函數程式碼及其相依性的結構。
   b. 包含用於本機測試之測試事件的資`events`料夾。
   c. 支援單元測試的`tests`資料夾。
   d. 用於規劃專`samconfig.toml`案設定的檔案。
   e. `ReadMe`檔案和其他基本的起始專案檔案。

   以下是起始專案目錄的範例：

```
sam-app
### README.md
### __init__.py
### events
#   ### event.json
```

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

您可以從可用的AWS 快速入門範本清單中選取，或提供您自己的自訂範本位置。

選擇 AWS 快速入門範本的步驟

1. 出現提示時，選取 [AWS 快速入門範本]。
2. 選取要開始使用的 AWS 快速入門範本。以下是範例：

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

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

```
Template: 4
```

若要選擇您自己的自訂範本位置

1. 出現提示時，選取「自訂範本位置」。

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

2. AWS SAMCLI將提示您提供樣板位置。

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

為範本 .zip 檔案封存提供下列任一位置：

- GitHub存放庫 — 存放庫中 .zip 檔案的GitHub路徑。檔案必須位於儲存庫的根目錄中。
- Mercurial存放庫 — 存放庫中 .zip 檔案的Mercurial路徑。檔案必須位於儲存庫的根目錄中。
- .zip 路徑 — 您 .zip 檔案的 HTTPS 或本機路徑。

3. AWS SAMCLI將使用您的自訂範本初始化您的無伺服器應用程式。

選擇執行環境

當您選擇AWS 快速入門範本時，會 AWS SAMCLI提示您為 Lambda 函數選取執行階段。顯示的選項清單 AWS SAMCLI是 Lambda 原生支援的執行階段。

- 執行時間會提供在執行環境中執行的特定語言環境。
- 部署到時 AWS 雲端，Lambda 服務會在執行環境中叫用您的函數。

您可以在自訂執行階段中使用任何其他程式設計語言。若要這麼做，您需要手動建立起始應用程式結構。然後，您可以透過設定自訂範本位置`sam init`來快速初始化應用程式。

從您的選擇中， AWS SAMCLI會為您的 Lambda 函數程式碼和相依性建立起始目錄。

如果 Lambda 在您的執行階段支援多個相依性管理程式，系統會提示您選擇偏好的相依性管理員。

選擇包裹類型

當您選擇AWS 快速入門範本和執行階段時， AWS SAMCLI會提示您選取套件類型。套件類型會決定 Lambda 函數的部署方式，以便與 Lambda 服務搭配使用。支援的套件類型有兩種：

1. 容器映像 — 包含基本作業系統、執行階段、Lambda 延伸模組、您的應用程式程式碼及其相依性。
2. .zip 檔案封存 — 包含您的應用程式程式碼及其相依性。

若要進一步了解部署套件類型，請參閱AWS Lambda 開發人員指南中的 Lambda 部署套件。

以下是將 Lambda 函數封裝為容器映像的應用程式目錄結構範例。會下 AWS SAMCLI載影像，並`Dockerfile`在函數的目錄中建立以指定影像。

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

以下是應用程式的目錄結構範例，其函數封裝為 .zip 檔案歸檔。

```
sam-app
### README.md
### __init__.py
### events
#   ### event.json
### hello_world
#   ### __init__.py
#   ### app.py
```

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

設定 AWS X-Ray 追蹤

您可以選擇啟動 AWS X-Ray 追蹤。若要深入瞭解，請參閱什麼是 AWS X-Ray？ 在AWS X-Ray 開發人員指南中。

如果啟用，則會設 AWS SAMCLI定您的 AWS SAM 範本。以下是範例：

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

使 CloudWatch 用 Amazon 應用程式洞察設定監控

您可以選擇使用 Amazon CloudWatch 應用程式洞察來啟用監控。若要進一步了解，請參閱 Amazon CloudWatch 使用者指南中的 Amazon 應用 CloudWatch 程式深入解析。

如果啟用，則會設 AWS SAMCLI定您的 AWS SAM 範本。以下是範例：

```
Resources:
  ApplicationResourceGroup:
    Type: AWS::ResourceGroups::Group
    Properties:
      Name:
        Fn::Join:
        - ''
```

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

命名您的應用

為您的應用程式提供名稱。 AWS SAMCLI會使用此名稱為您的應用程式建立頂層資料夾。

山姆初始化的選項

以下是您可以搭配`sam init`指令使用的一些主要選項。如需所有選項的清單，請參閱sam init。

使用自訂範本位置初始化應用程式

使用選`--location`項並提供支援的自訂範本位置。以下是範例：

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

初始化沒有互動式流程的應用程式

使用選`--no-interactive`項並在命令列中提供您的組態選擇，以略過互動式流程。以下是範例：

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

故障診斷

若要疑難排解 AWS SAMCLI，請參閱AWS SAMCLI疑難排。

範例

使用 Hello World AWS 入門範本初始化新的無伺服器應用程式

如需此範例，請參閱教學課程：部署 Hello World 應用程式 步驟 1：初始化範例 Hello World 應用程式中的。

使用自訂範本位置初始化新的無伺服器應用程式

以下是為自訂範本提供GitHub位置的範例：

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

以下是本機檔案路徑的範例：

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

以下是 HTTPS 可存取的路徑範例：

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

進一步了解

若要進一步瞭解如何使用`sam init`指令，請參閱下列內容：

- 學習 AWS SAM：山姆初始化 — Serverless Land「學習 AWS SAM」系列YouTube。
- 建構無伺服器應用程式以搭配使用 AWS SAMCLI (SAM S2E7 的工作階段) — 開啟序列的工作階段。 AWS SAM YouTube

後續步驟

現在您已經建立了 AWS SAM 專案，就可以開始編寫應用程式了。如定義您的基礎架構 AWS SAM需執行此操作所需完成之工作的詳細指示，請參閱。

定義您的基礎架構 AWS SAM

現在，您已經建立了專案，就可以使用 AWS SAM. 通過配置 AWS SAM 模板來定義應用程序的資源和屬性（即 AWS SAM 項目中的`template.yaml`文件）來執行此操作。

本節中的主題提供有關在 AWS SAM 範本 (`template.yaml`檔案) 中定義基礎結構的內容。同時也包含針對特定使用案例定義資源的主題，例如使用 Lambda 層、使用巢狀應用程式、控制 API Gateway API 的存取、使用 Step Functions 協調 AWS 資源、簽署應用程式程式碼，以及驗證範本。 AWS SAM

主題

- 在 AWS SAM 範本中定義應用程式資源
- 在 AWS SAM 範本中設定和管理資源存取
- 使用 AWS SAM 範本控制 API 存取
- 使用 Lambda 層來提高效率 AWS SAM
- 使用嵌套應用程序重複使用代碼和資源 AWS SAM
- 使用 EventBridge排程器管理基於時間的事件 AWS SAM
- 協調 AWS 資源 AWS Step Functions
- 為您的應用程式設定 AWS SAM 程式碼簽章
- 驗證 AWS SAM 範本檔

在 AWS SAM 範本中定義應用程式資源

您可以在 AWS SAM 範本的`Resources`區段中定義無伺服器應用程式使用的 AWS 資源。定義資源時，您可以識別資源是什麼、它與其他資源的互動方式，以及如何存取資源 (也就是資源的權限)。

AWS SAM 範本的`Resources`區段可以包含 AWS CloudFormation 資源和 AWS SAM 資源的組合。此外，您可以針對下列資源使用 AWS SAM的簡短語法：

AWS SAM 短手語法	它與相關 AWS 資源有什麼作用
AWS::Serverless::Api	建立可透過 HTTPS 端點叫用的 API Gateway 資源和方法集合。
AWS::Serverless::Application	將 Amazon S3 儲存貯體AWS Serverless Application Repository或來自 Amazon S3 儲存貯體的無伺服器應用程式嵌入為巢狀應用程式。

AWS SAM 短手語法	它與相關 AWS 資源有什麼作用
AWS::Serverless::Connector	設定兩個資源之間的權限。如需連接器的簡介，請參閱使用 AWS SAM 連接器管理資源權限。
AWS::Serverless::Function	建立 AWS Lambda 函數、 AWS Identity and Access Management (IAM) 執行角色，以及觸發函數的事件來源對應。
AWS::Serverless::GraphQLApi	為您的無伺服器應用程式建立並設定 AWS AppSync GraphQL API。
AWS::Serverless::HttpApi	建立 Amazon API Gateway HTTP API，與其他 API 相比，您可以使用更低的延遲和更低的成本來建立 REST 風格 API。
AWS::Serverless::LayerVersion	建立 Lambda LayerVersion ，其中包含 Lambda 函數所需的程式庫或執行階段程式碼。
AWS::Serverless::SimpleTable	建立具有單一屬性主索引鍵的 DynamoDB 表格。
AWS::Serverless::StateMachine	建立 AWS Step Functions 狀態機器，您可以使用它來協調 AWS Lambda 功能和其他 AWS 資源，以形成複雜且強大的工作流程。

上述資源也列於中AWS SAM 資源和屬性。

如需所有 AWS 資源和屬性類型 AWS CloudFormation 和 AWS SAM 支援的參考資訊，請參閱《AWS CloudFormation 使用指南》中的AWS 資源和屬性類型參考資料。

在 AWS SAM 範本中設定和管理資源存取

為了讓您的 AWS 資源彼此互動，必須在資源之間設定適當的存取權限和權限。這樣做需要配置 AWS Identity and Access Management (IAM) 使用者、角色和政策，才能以安全的方式完成互動。

本節中的主題都與設定範本中定義之資源的存取權限有關。本節從一般最佳作法開始。接下來的兩個主題將檢閱您在無伺服器應用程式中參照的資源之間設定存取權和權限的兩個選項： AWS SAM

連接器和 AWS SAM 原則範本。最後一個主題提供了使用與管理使用者相同的機制來管理 AWS CloudFormation 使用者存取的詳細資料。

若要深入瞭解，請參閱《AWS CloudFormation 使用者指南》 AWS Identity and Access Management中的「控制存取權」。

AWS Serverless Application Model (AWS SAM) 提供兩個選項，可簡化無伺服器應用程式的存取與權限管理。

1. AWS SAM 連接器
2. AWS SAM 策略範本

AWS SAM 連接器

連接器是佈建兩個資源之間權限的一種方式。您可以通過描述他們應該如何在 AWS SAM 模板中相互交互來做到這一點。它們可以使用`Connectors`資源屬性或資`AWS::Serverless::Connector`源類型來定義。連接器支援資`Read`源組合之間的 AWS 資料和事件佈建和`Write`存取。若要進一步瞭解 AWS SAM 連接器，請參閱使用 AWS SAM 連接器管理資源權限。

AWS SAM 策略範本

AWS SAM 原則範本是預先定義的權限集合，您可以將這些權限新增至 AWS SAM 範本，以管理 AWS Lambda 功能、 AWS Step Functions 狀態機器及其互動之資源之間的存取和權限。若要深入了解 AWS SAM 策略範本，請參閱AWS SAM策略範本。

AWS CloudFormation 機制

AWS CloudFormation 機制包括設定 IAM 使用者、角色和政策，以管理 AWS 資源之間的許可。如需進一步了解，請參閱使用 AWS CloudFormation 機制管理權限。

最佳實務

在整個無伺服器應用程式中，您可以使用多種方法來設定資源之間的權限。因此，您可以為每個案例選取最佳選項，並在整個應用程式中同時使用多個選項。選擇最適合您的選擇時，請考慮以下幾點：

- AWS SAM 連接器和政策範本都減少了促進 AWS 資源之間安全互動所需的 IAM 專業知識。如果支援，請使用連接器和原則範本。
- AWS SAM 連接器提供簡單且直覺的簡短語法來定義 AWS SAM 範本中的許可，並且需要最少的 IAM 專業知識。當 AWS SAM 連接器和原則範本都受支援時，請使用 AWS SAM 連接器。

- AWS SAM 連接器可以在支援的來源和目標資 AWS SAM 源之間佈建`Read`和`Write`存取資料和事件。如需支援資源的清單，請參閱AWS SAM 連接器參考。支援時，請使用 AWS SAM 連接器。
- 雖然 AWS SAM 政策範本僅限於 Lambda 函數之間的許可，但 Step Functions 會狀態機器及其互動的 AWS 資源，但政策範本確實支援所有 CRUD 作業。如果受支援，以及當您的案例的 AWS SAM 原則範本可用時，請使用 AWS SAM 原則範本。如需可用策略範本的清單，請參閱AWS SAM策略範本。
- 對於所有其他案例，或需要粒度時，請使用 AWS CloudFormation 機制。

使用 AWS SAM 連接器管理資源權限

主題

- 什麼是 AWS SAM 連接器？
- 連接器的範例
- 來源與目標資源之間的支援連線
- 使用連接器
- 連接器如何工作
- AWS SAM 連接器的優點
- 進一步了解
- 提供意見

什麼是 AWS SAM 連接器？

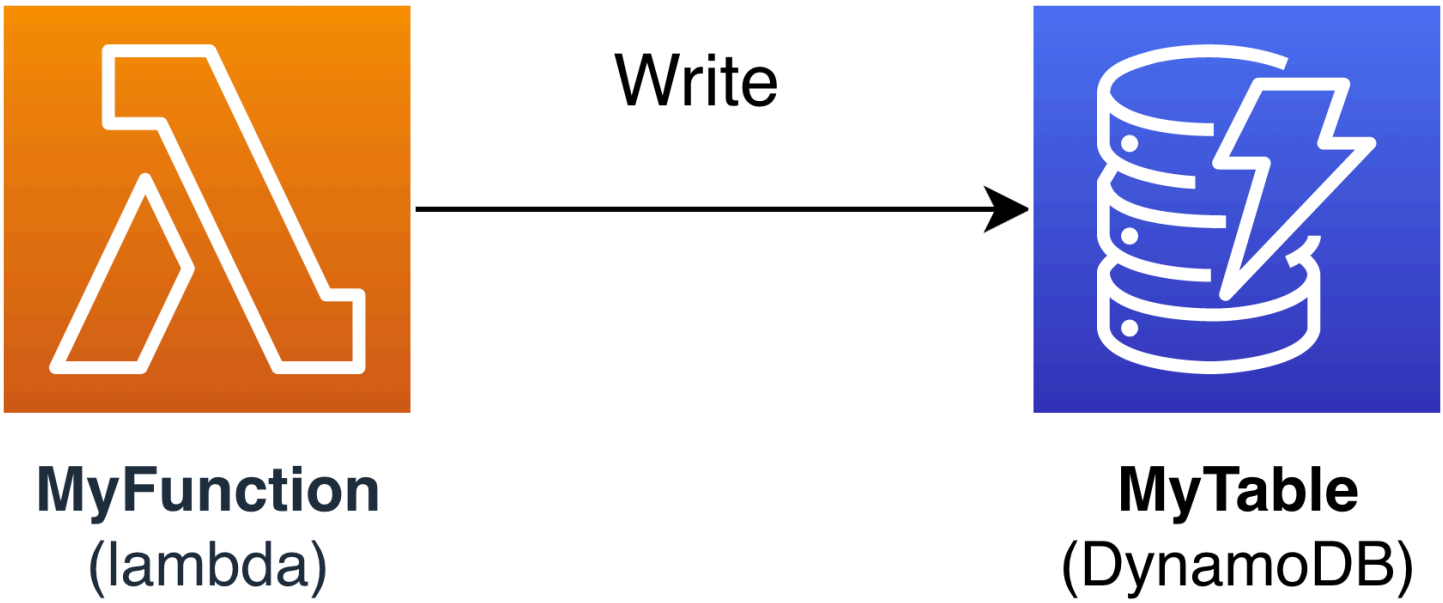
連接器是一種 AWS Serverless Application Model (AWS SAM) 抽象資源類型，識別為`AWS::Serverless::Connector`，可在無伺服器應用程式資源之間提供簡單且適當範圍的權限。透過將`Connectors`資源內嵌在來源資源中來使用資源屬性。然後，定義您的目標資源，並描述資料或事件在這些資源之間的流動方式。 AWS SAM 然後制定必要的訪問策略，以促進所需的交互。

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  <source-resource-logical-id>:
    Type: <resource-type>
    ...
    Connectors:
      <connector-name>:
```

```
        Properties:
          Destination:
            <properties-that-identify-destination-resource>
          Permissions:
            <permission-types-to-provision>
  ...
```

連接器的範例

在此範例中，我們使用連接器將 AWS Lambda 函數中的資料寫入 Amazon DynamoDB 表格。




```
Transform: AWS::Serverless-2016-10-31
Resources:
  MyTable:
    Type: AWS::Serverless::SimpleTable
  MyFunction:
    Type: AWS::Serverless::Function
    Connectors:
      MyConn:
        Properties:
          Destination:
            Id: MyTable
          Permissions:
            - Write
    Properties:
      Runtime: nodejs16.x
      Handler: index.handler
      InlineCode: |
```

```
        const AWS = require("aws-sdk");
        const docClient = new AWS.DynamoDB.DocumentClient();
        exports.handler = async (event, context) => {
          await docClient.put({
            TableName: process.env.TABLE_NAME,
            Item: {
              id: context.awsRequestId,
              event: JSON.stringify(event)
            }
          }).promise();
        }
      Environment:
        Variables:
          TABLE_NAME: !Ref MyTable
```

資Connectors源屬性內嵌於 Lambda 函數來源資源中。DynamoDB 資料表會定義為使用該屬性的目標資源。Id連接器將佈建這兩個資源之間的Write權限。

當您將 AWS SAM 範本部署到時 AWS CloudFormation， AWS SAM 會自動撰寫此連線運作所需的必要存取原則。

來源與目標資源之間的支援連線

連接器支援以Read及來源與目標資源連線的選取組合之間的資Write料和事件權限類型。例如，Write連接器支援來AWS::ApiGateway::RestApi源資源與AWS::Lambda::Function目標資源之間的連線。

您可以使用支援的屬性組合來定義來源和目標資源。屬性需求將取決於您建立的連線以及資源的定義位置。

> **Note**
>
> 連接器可以在支援的無伺服器和非伺服器資源類型之間佈建權限。

如需支援的資源連線及其屬性需求的清單，請參閱連接器支援的來源和目標資源類型。

使用連接器

定義讀取和寫入權限

Read和Write權限可以在單一連接器中佈建：

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Lambda::Function
    Connectors:
      MyTableConn:
        Properties:
          Destination:
            Id: MyTable
          Permissions:
            - Read
            - Write
  MyTable:
    Type: AWS::DynamoDB::Table
```

使用其他支援的屬性來定義資源

對於來源和目標資源，在同一個範本中定義時，請使用Id屬性。或者，您Qualifier可以新增一個來縮小已定義資源的範圍。當資源不在同一個範本中時，請使用支援的屬性組合。

- 如需來源和目標資源支援的屬性組合清單，請參閱連接器支援的來源和目標資源類型。
- 如需可與連接器搭配使用之性質的描述，請參閱〈〉AWS::Serverless::Connector。

當您使用屬性以外的屬性定義來源資源時Id，請使用SourceReference屬性。

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  <source-resource-logical-id>:
    Type: <resource-type>
    ...
    Connectors:
      <connector-name>:
        Properties:
          SourceReference:
            Qualifier: <optional-qualifier>
            <other-supported-properties>
          Destination:
            <properties-that-identify-destination-resource>
```

```
          Permissions:
            <permission-types-to-provision>
```

以下是一個範例，使用 a `Qualifier` 來縮小 Amazon API Gateway 資源的範圍：

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyApi:
    Type: AWS::Serverless::Api
    Connectors:
      ApiToLambdaConn:
        Properties:
          SourceReference:
            Qualifier: Prod/GET/foobar
          Destination:
            Id: MyFunction
          Permissions:
            - Write
  ...
```

`Type`以下是使用支援的`Arn`和組合來定義另一個範本的目標資源的範例：

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Connectors:
      TableConn:
        Properties:
          Destination:
            Type: AWS::DynamoDB::Table
            Arn: !GetAtt MyTable.Arn
  ...
```

從單一來源建立多個連接器

在來源資源中，您可以定義多個連接器，每個連接器都有不同的目標資源。

```
AWSTemplateFormatVersion: '2010-09-09'
```

```
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Connectors:
      BucketConn:
        Properties:
          Destination:
            Id: MyBucket
          Permissions:
            - Read
            - Write
      SQSConn:
        Properties:
          Destination:
            Id: MyQueue
          Permissions:
            - Read
            - Write
      TableConn:
        Properties:
          Destination:
            Id: MyTable
          Permissions:
            - Read
            - Write
      TableConnWithTableArn:
        Properties:
          Destination:
            Type: AWS::DynamoDB::Table
            Arn: !GetAtt MyTable.Arn
          Permissions:
            - Read
            - Write
...
```

建立多目標連接器

在來源資源中，您可以定義具有多個目標資源的單一連接器。以下是連接到亞馬遜簡單儲存服務 (Amazon S3) 儲存貯體和 DynamoDB 表的 Lambda 函數來源資源範例：

```
AWSTemplateFormatVersion: '2010-09-09'
```

```
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Connectors:
      WriteAccessConn:
        Properties:
          Destination:
            - Id: OutputBucket
            - Id: CredentialTable
          Permissions:
            - Write
  ...
  OutputBucket:
    Type: AWS::S3::Bucket
  CredentialTable:
    Type: AWS::DynamoDB::Table
```

使用連接器定義資源屬性

您可以為資源定義資源屬性，以指定其他行為和關係。若要進一步瞭解資源屬性，請參閱AWS CloudFormation 使用指南中的資源屬性參考。

您可以在與連接器內容相同的層級上定義資源屬性，將資源屬性新增至內嵌連接器。當您的 AWS SAM 範本在部署時轉換時，屬性會傳遞至產生的資源。

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Connectors:
      MyConn:
        DeletionPolicy: Retain
        DependsOn: AnotherFunction
        Properties:
          ...
```

連接器如何工作

> **Note**
>
> 本節說明連接器如何在幕後佈建必要的資源。使用連接器時會自動發生這種情況。

首先，內嵌Connectors資源屬性會轉換為AWS::Serverless::Connector資源類型。其邏輯 ID 會自動建立為 *< source-resource-logical-id >< embedded-connector-logical-id >*。

例如，下面是一個嵌入式連接器：

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Lambda::Function
    Connectors:
      MyConn:
        Properties:
          Destination:
            Id: MyTable
          Permissions:
            - Read
            - Write
      MyTable:
        Type: AWS::DynamoDB::Table
```

這將生成以下AWS::Serverless::Connector資源：

```
Transform: AWS::Serverless-2016-10-31
Resources:
  ...
  MyFunctionMyConn:
    Type: AWS::Serverless::Connector
    Properties:
      Source:
        Id: MyFunction
      Destination:
        Id: MyTable
```

```
      Permissions:
        - Read
        - Write
```

> **Note**
>
> 您也可以使用此語法在 AWS SAM 範本中定義連接器。當您的來源資源是在與連接器不同的範本上定義的時候，建議您這麼做。

接下來，會自動構成此連線的必要存取原則。如需有關連接器產生之資源的詳細資訊，請參閱AWS CloudFormation指定時產生的資源 AWS::Serverless::Connector。

AWS SAM 連接器的優點

透過在資源之間自動撰寫適當的存取原則，連接器可讓您編寫無伺服器應用程式並專注於應用程式架構，而不需要 AWS 授權功能、原則語言和服務特定安全性設定方面的專業知識。因此，對於無伺服器開發新手的開發人員或希望提高開發速度的經驗豐富的開發人員來說，連接器是一個很大的好處。

進一步了解

如需使用 AWS SAM 連接器的詳細資訊，請參閱AWS::Serverless::Connector。

提供意見

若要提供有關連接器的意見反應，請在serverless-application-model AWS GitHub儲存庫中提交新問題。

AWS SAM策略範本

AWS Serverless Application Model (AWS SAM) 可讓您從政策範本清單中選擇，將 Lambda 函數和 AWS Step Functions 狀態機器的許可範圍限定在應用程式使用的資源。

AWS SAM 使用原則範本 AWS Serverless Application Repository 的應用程式不需要任何特殊的客戶確認，即可從. AWS Serverless Application Repository

若您希望申請增加新的政策範本，請執行以下動作：

1. 針對專案分支中的原則來源檔案提交提取要求。`develop` AWS SAM GitHub 您可以在網站上的原始檔案中找到原始檔案。 GitHub

2. 在 AWS SAM GitHub 專案中提交問題，其中包含提取請求的原因以及要求連結。使用此連結提交新問題:AWS Serverless Application Model: 問題。

語法

針對您在範本檔案中指定的每個原則 AWS SAM 範本，您必須一律指定包含原則範本預留位置值的物件。如果政策範本不需要任何預留位置值，您必須指定空白物件。

YAML

```
MyFunction:
  Type: AWS::Serverless::Function
  Properties:
    Policies:
      - PolicyTemplateName1:        # Policy template with placeholder value
          Key1: Value1
      - PolicyTemplateName2: {}     # Policy template with no placeholder value
```

範例

範例 1：帶有預留位置值的政策範本

以下範例顯示 SQSPollerPolicy 政策範本應將 QueueName 視為資源。 AWS SAM 範本會擷取 "MyQueue" Amazon SQS 佇列的名稱，您可以在相同的應用程式中建立該佇列，也可以將其作為應用程式的參數要求。

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

範例 2：無預留位置值的政策範本

以下範例包含 CloudWatchPutMetricPolicy 政策範本，其並無任何預留位置值。

> **Note**
>
> 即使沒有預留位置值，您也必須指定空白物件，否則會導致錯誤。

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

策略範本表格

以下是可用策略範本的表格。

政策範本	描述		
AcmGetCertificatePolicy	授與讀取憑證的權限 AWS Certificate Manager。		
AMIDescribePolicy	授予描述 Amazon 機器映像（AMI）的權限。		
AthenaQueryPolicy	授與執行 Athena 查詢的權限。		
AWSSecretsManagerGetSecretValuePolicy	授予取得指定密碼之密碼密 AWS Secrets Manager 碼值的權限。		
AWSSecretsManagerRotationPolicy	授予旋轉密碼的權限 AWS Secrets Manager。		

政策範本	描述		
CloudFormationDescribeStacksPolicy	授予描述 AWS CloudFormation 堆棧的權限。		
CloudWatchDashboardPolicy	授予將指標放置在 CloudWatch 儀表板上操作的權限。		
CloudWatchDescribeAlarmHistoryPolicy	授予描述 CloudWatch 警報歷史記錄的權限。		
CloudWatchPutMetricPolicy	授予將指標傳送至的權限 CloudWatch。		
CodeCommitCrudPolicy	授與在特定存放庫中建立/讀取/更新/刪除物件的權限。 CodeCommit		
CodeCommitReadPolicy	授與讀取特定 CodeCommit 存放庫中物件的權限。		
CodePipelineLambdaExecutionPolicy	授與叫用的 Lambda 函數的權限，CodePipeline 以報告工作狀態。		
CodePipelineReadOnlyPolicy	授予讀取權限以取得 CodePipeline 管線的詳細資料。		
ComprehendBasicAccessPolicy	授予偵測實體、關鍵片語、語言和情緒的權限。		

政策範本	描述		
CostExplorerReadOnlyPolicy	將帳單歷史記錄的唯讀 Cost Explorer API 授與唯讀權限。		
DynamoDBBackupFullAccessPolicy	為表格授與 DynamoDB 隨需備份的讀取和寫入權限。		
DynamoDBCrudPolicy	將建立、讀取、更新和刪除權限授予 Amazon DynamoDB 表格。		
DynamoDBReadPolicy	為 DynamoDB 資料表提供唯讀權限。		
DynamoDBReconfigurePolicy	授與重新設定 DynamoDB 表格的權限。		
DynamoDBRestoreFromBackupPolicy	授與從備份還原 DynamoDB 資料表的權限。		
DynamoDBStreamReadPolicy	授予描述和讀取 DynamoDB 串流和記錄的權限。		
DynamoDBWritePolicy	將唯寫權限授與 DynamoDB 表格。		
EC2CopyImagePolicy	授予複製 Amazon EC2 映像的權限。		
EC2DescribePolicy	授予描述亞馬遜彈性運算雲端 (Amazon EC2) 執行個體的權限。		
EcsRunTaskPolicy	授與啟動任務定義新任務的權限。		

政策範本	描述		
EFSWriteAccessPolicy	授與使用寫入存取權來掛接 Amazon EFS 檔案系統的權限。		
EKSDescribePolicy	授予描述或列出 Amazon EKS 叢集的權限。		
ElasticMapReduceAddJobFlowStepsPolicy	授與將新步驟新增至執行中叢集的權限。		
ElasticMapReduceCancelStepsPolicy	授與取消擱置中步驟或執行中叢集中步驟的權限。		
ElasticMapReduceModifyInstanceFleetPolicy	授與列出叢集中執行個體叢集的詳細資料和修改容量的權限。		
ElasticMapReduceModifyInstanceGroupsPolicy	授與列出叢集中執行個體群組的詳細資料和修改設定的權限。		
ElasticMapReduceSetTerminationProtectionPolicy	授與設定叢集終止保護的權限。		

政策範本	描述		
ElasticMapReduceTerminateJobFlowsPolicy	授與關閉叢集的權限。		
ElasticsearchHttpPostPolicy	給予 POST 權限 Amazon OpenSearch 服務。		
EventBridgePutEventsPolicy	授與將事件傳送至的權限 EventBridge。		
FilterLogEventsPolicy	授與從指定 CloudWatch 記錄群組篩選記錄事件的權限。		
FirehoseCrudPolicy	授予建立、寫入、更新和刪除 Firehose 傳送串流的權限。		
FirehoseWritePolicy	授予寫入 Firehose 交付串流的權限。		
KinesisCrudPolicy	授予建立、發佈和刪除 Amazon Kinesis 串流的權限。		
KinesisStreamReadPolicy	授予列出和讀取 Amazon Kinesis 串流的權限。		
KMSDecryptPolicy	授予使用 AWS Key Management Service (AWS KMS) 金鑰進行解密的權限。		
KMSEncryptPolicy	授予使用 AWS Key Management Service (AWS KMS) 金鑰加密的權限。		
LambdaInvokePolicy	授與叫用 AWS Lambda 函數、別名或版本的權限。		

政策範本	描述		
MobileAnalyticsWriteOnlyAccessPolicy	提供唯寫權限，以便為所有應用程式資源放置事件資料。		
OrganizationsListAccountsPolicy	授與列出子帳號名稱和 ID 的唯讀權限。		
PinpointEndpointAccessPolicy	授予取得和更新 Amazon 精確應用程式端點的權限。		
PollyFullAccessPolicy	授予對 Amazon Polly 詞典資源的完全訪問權限。		
RekognitionDetectOnlyPolicy	授予檢測臉部，標籤和文本的權限。		
RekognitionFacesManagementPolicy	授予在 Amazon Rekognition 集合中新增、刪除和搜尋臉孔的權限。		
RekognitionFacesPolicy	授予比較和檢測臉部和標籤的權限。		
RekognitionLabelsPolicy	授予檢測對象和協調標籤的權限。		
RekognitionNoDataAccessPolicy	授予比較和檢測臉部和標籤的權限。		
RekognitionReadPolicy	授予列出和搜索面孔的權限。		

政策範本	描述		
RekognitionWriteOnlyAccessPolicy	授予建立集合和索引臉孔的權限。		
Route53ChangeResourceRecordSetsPolicy	授予在 Route 53 中變更資源記錄集的權限。		
S3CrudPolicy	授予建立、讀取、更新和刪除權限，以便對 Amazon S3 儲存貯體中的物件執行動作。		
S3FullAccessPolicy	授予對 Amazon S3 儲存貯體中物件執行動作的完整存取權限。		
S3ReadPolicy	授予讀取 Amazon Simple Storage Service (Amazon S3) 儲存貯體中物件的唯讀權限。		
S3WritePolicy	授予將物件寫入 Amazon S3 儲存貯體的寫入權限。		
SageMakerCreateEndpointConfigPolicy	授與在中建立端點組態的權限 SageMaker。		
SageMakerCreateEndpointPolicy	授與在中建立端點的權限 SageMaker。		
ServerlessRepoReadWriteAccessPolicy	授予在服務中建立和列出應用程式的權 AWS Serverless Application Repository 限。		

政策範本	描述		
SESBulkTemplatedCrudPolicy	授予發送電子郵件，模板化電子郵件，模板化批量電子郵件和驗證身份的權限。		
SESBulkTemplatedCrudPolicy_v2	授予傳送 Amazon SES 電子郵件、範本化電子郵件和範本化大量電子郵件的權限，以及驗證身分。		
SESCrudPolicy	授予發送電子郵件和驗證身份的權限。		
SESEmailTemplateCrudPolicy	授予建立、取得、列出、更新和刪除 Amazon SES 電子郵件範本的權限。		
SESSendBouncePolicy	SendBounce 授予亞馬遜簡易電子郵件服務 (Amazon SES) 身分識別的權限。		
SNSCrudPolicy	授予建立、發佈和訂閱 Amazon SNS 主題的權限。		
SNSPublishMessagePolicy	授予將訊息發佈至亞馬遜簡單通知服務 (Amazon SNS) 主題的權限。		
SQSPollerPolicy	授予輪詢 Amazon Simple Queue Service (Amazon SQS) 佇列的權限。		
SQSSendMessagePolicy	授予將訊息傳送至 Amazon SQS 佇列的權限。		
SSMParameterReadPolicy	授予從 Amazon EC2 Systems Manager (SSM) 參數存放區存取參數的權限，以便在此帳戶中載入機密。當參數名稱沒有斜線前綴時使用。		

政策範本	描述		
SSMParameterWithSlashPrefixReadPolicy	授予從 Amazon EC2 Systems Manager (SSM) 參數存放區存取參數的權限，以便在此帳戶中載入機密。當參數名稱具有斜線前綴時使用。		
StepFunctionsExecutionPolicy	授予啟動 Step Functions 狀態機器執行的權限。		
TextractDetectAnalyzePolicy	允許訪問以檢測和分析使用亞馬遜文本提取的文檔。		
TextractGetResultPolicy	允許從 Amazon Textract 獲取檢測到和分析的文檔的訪問權限。		
TextractPolicy	可以完全訪問 Amazon Textract。		
VPCAccessPolicy	提供建立、刪除、描述和卸離彈性網路介面的存取權。		

疑難排解

SAM CLI 錯誤:「必須指定原則範本 '< policy-template-name >' 的有效參數值」

執行 `sam build` 時，您會看到下列錯誤：

```
"Must specify valid parameter values for policy template '<policy-template-name>'"
```

這表示您在宣告沒有任何預留位置值的原則範本時，並未傳遞空白物件。

若要修正此問題，請宣告原則，如下列範例所示CloudWatchPutMetricPolicy。

```
MyFunction:
  Policies:
```

```
    - CloudWatchPutMetricPolicy: {}
```

策略範本清單

以下是可用的原則範本，以及套用至每個原則範本的權限。 AWS Serverless Application Model (AWS SAM) 將適當的資訊自動填入預留位置項目 (例如「 AWS 區域」和「帳戶 ID」)。

主題

- AcmGetCertificatePolicy
- AMIDescribePolicy
- AthenaQueryPolicy
- AWSSecretsManagerGetSecretValuePolicy
- AWSSecretsManagerRotationPolicy
- CloudFormationDescribeStacksPolicy
- CloudWatchDashboardPolicy
- CloudWatchDescribeAlarmHistoryPolicy
- CloudWatchPutMetricPolicy
- CodePipelineLambdaExecutionPolicy
- CodePipelineReadOnlyPolicy
- CodeCommitCrudPolicy
- CodeCommitReadPolicy
- ComprehendBasicAccessPolicy
- CostExplorerReadOnlyPolicy
- DynamoDBBackupFullAccessPolicy
- DynamoDBCrudPolicy
- DynamoDBReadPolicy
- DynamoDBReconfigurePolicy
- DynamoDBRestoreFromBackupPolicy
- DynamoDBStreamReadPolicy
- DynamoDBWritePolicy
- EC2CopyImagePolicy
- EC2DescribePolicy

- EcsRunTaskPolicy
- EFSWriteAccessPolicy
- EKSDescribePolicy
- ElasticMapReduceAddJobFlowStepsPolicy
- ElasticMapReduceCancelStepsPolicy
- ElasticMapReduceModifyInstanceFleetPolicy
- ElasticMapReduceModifyInstanceGroupsPolicy
- ElasticMapReduceSetTerminationProtectionPolicy
- ElasticMapReduceTerminateJobFlowsPolicy
- ElasticsearchHttpPostPolicy
- EventBridgePutEventsPolicy
- FilterLogEventsPolicy
- FirehoseCrudPolicy
- FirehoseWritePolicy
- KinesisCrudPolicy
- KinesisStreamReadPolicy
- KMSDecryptPolicy
- KMSEncryptPolicy
- LambdaInvokePolicy
- MobileAnalyticsWriteOnlyAccessPolicy
- OrganizationsListAccountsPolicy
- PinpointEndpointAccessPolicy
- PollyFullAccessPolicy
- RekognitionDetectOnlyPolicy
- RekognitionFacesManagementPolicy
- RekognitionFacesPolicy
- RekognitionLabelsPolicy
- RekognitionNoDataAccessPolicy
- RekognitionReadPolicy
- RekognitionWriteOnlyAccessPolicy

- Route53ChangeResourceRecordSetsPolicy
- S3CrudPolicy
- S3FullAccessPolicy
- S3ReadPolicy
- S3WritePolicy
- SageMakerCreateEndpointConfigPolicy
- SageMakerCreateEndpointPolicy
- ServerlessRepoReadWriteAccessPolicy
- SESBulkTemplatedCrudPolicy
- SESBulkTemplatedCrudPolicy_v2
- SESCrudPolicy
- SESEmailTemplateCrudPolicy
- SESSendBouncePolicy
- SNSCrudPolicy
- SNSPublishMessagePolicy
- SQSPollerPolicy
- SQSSendMessagePolicy
- SSMParameterReadPolicy
- SSMParameterWithSlashPrefixReadPolicy
- StepFunctionsExecutionPolicy
- TextractDetectAnalyzePolicy
- TextractGetResultPolicy
- TextractPolicy
- VPCAccessPolicy

AcmGetCertificatePolicy

授與讀取憑證的權限 AWS Certificate Manager。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
```

```
      "acm:GetCertificate"
    ],
    "Resource": {
      "Fn::Sub": [
        "${certificateArn}",
        {
          "certificateArn": {
            "Ref": "CertificateArn"
          }
        }
      ]
    }
  }
]
```

AMIDescribePolicy

授予描述 Amazon 機器映像（AMI）的權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ec2:DescribeImages"
    ],
    "Resource": "*"
  }
]
```

AthenaQueryPolicy

授與執行 Athena 查詢的權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "athena:ListWorkGroups",
      "athena:GetExecutionEngine",
      "athena:GetExecutionEngines",
      "athena:GetNamespace",
      "athena:GetCatalogs",
      "athena:GetNamespaces",
```

```
      "athena:GetTables",
      "athena:GetTable"
    ],
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "athena:StartQueryExecution",
      "athena:GetQueryResults",
      "athena:DeleteNamedQuery",
      "athena:GetNamedQuery",
      "athena:ListQueryExecutions",
      "athena:StopQueryExecution",
      "athena:GetQueryResultsStream",
      "athena:ListNamedQueries",
      "athena:CreateNamedQuery",
      "athena:GetQueryExecution",
      "athena:BatchGetNamedQuery",
      "athena:BatchGetQueryExecution",
      "athena:GetWorkGroup"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:athena:${AWS::Region}:${AWS::AccountId}:workgroup/
${workgroupName}",
        {
          "workgroupName": {
            "Ref": "WorkGroupName"
          }
        }
      ]
    }
  }
]
```

AWSSecretsManagerGetSecretValuePolicy

授予取得指定密碼之密碼密 AWS Secrets Manager 碼值的權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
```

```
      "secretsmanager:GetSecretValue"
    ],
    "Resource": {
      "Fn::Sub": [
        "${secretArn}",
        {
          "secretArn": {
            "Ref": "SecretArn"
          }
        }
      ]
    }
  }
]
```

AWSSecretsManagerRotationPolicy

授予旋轉密碼的權限 AWS Secrets Manager。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "secretsmanager:DescribeSecret",
      "secretsmanager:GetSecretValue",
      "secretsmanager:PutSecretValue",
      "secretsmanager:UpdateSecretVersionStage"
    ],
    "Resource": {
      "Fn::Sub": "arn:${AWS::Partition}:secretsmanager:${AWS::Region}:
${AWS::AccountId}:secret:*"
    },
    "Condition": {
      "StringEquals": {
        "secretsmanager:resource/AllowRotationLambdaArn": {
          "Fn::Sub": [
            "arn:${AWS::Partition}:lambda:${AWS::Region}:${AWS::AccountId}:function:
${functionName}",
            {
              "functionName": {
                "Ref": "FunctionName"
              }
            }
          ]
```

```
        }
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "secretsmanager:GetRandomPassword"
    ],
    "Resource": "*"
  }
]
```

CloudFormationDescribeStacksPolicy

授予描述 AWS CloudFormation 堆棧的權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "cloudformation:DescribeStacks"
    ],
    "Resource": {
      "Fn::Sub": "arn:${AWS::Partition}:cloudformation:${AWS::Region}:
${AWS::AccountId}:stack/*"
    }
  }
]
```

CloudWatchDashboardPolicy

授予將指標放置在 CloudWatch 儀表板上操作的權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "cloudwatch:GetDashboard",
      "cloudwatch:ListDashboards",
      "cloudwatch:PutDashboard",
      "cloudwatch:ListMetrics"
    ],
```

```
    "Resource": "*"
  }
]
```

CloudWatchDescribeAlarmHistoryPolicy

授予描述 Amazon CloudWatch 警報歷史記錄的權限。

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

CloudWatchPutMetricPolicy

授予將指標傳送至的權限 CloudWatch。

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

CodePipelineLambdaExecutionPolicy

授與叫用的 Lambda 函數的權限， AWS CodePipeline 以報告工作狀態。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "codepipeline:PutJobSuccessResult",
      "codepipeline:PutJobFailureResult"
    ],
```

```
    "Resource": "*"
  }
]
```

CodePipelineReadOnlyPolicy

授予讀取權限以取得 CodePipeline 管線的詳細資料。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "codepipeline:ListPipelineExecutions"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:codepipeline:${AWS::Region}:${AWS::AccountId}:
${pipelinename}",
        {
          "pipelinename": {
            "Ref": "PipelineName"
          }
        }
      ]
    }
  }
]
```

CodeCommitCrudPolicy

授與在特定 CodeCommit存放庫中建立、讀取、更新和刪除物件的權限。

```
"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

授與讀取特定 CodeCommit 存放庫中物件的權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "codecommit:GitPull",
      "codecommit:GetBranch",
      "codecommit:ListBranches",
      "codecommit:BatchDescribeMergeConflicts",
      "codecommit:DescribeMergeConflicts",
      "codecommit:GetMergeCommit",
      "codecommit:GetMergeOptions",
      "codecommit:BatchGetPullRequests",
      "codecommit:DescribePullRequestEvents",
      "codecommit:GetCommentsForPullRequest",
```

```
      "codecommit:GetCommitsFromMergeBase",
      "codecommit:GetMergeConflicts",
      "codecommit:GetPullRequest",
      "codecommit:ListPullRequests",
      "codecommit:GetBlob",
      "codecommit:GetFile",
      "codecommit:GetFolder",
      "codecommit:GetComment",
      "codecommit:GetCommentsForComparedCommit",
      "codecommit:BatchGetCommits",
      "codecommit:GetCommit",
      "codecommit:GetCommitHistory",
      "codecommit:GetDifferences",
      "codecommit:GetObjectIdentifier",
      "codecommit:GetReferences",
      "codecommit:GetTree",
      "codecommit:GetRepository",
      "codecommit:ListTagsForResource",
      "codecommit:GetRepositoryTriggers",
      "codecommit:TestRepositoryTriggers",
      "codecommit:GetBranch",
      "codecommit:GetCommit",
      "codecommit:GetUploadArchiveStatus"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:codecommit:${AWS::Region}:${AWS::AccountId}:
${repositoryName}",
        {
          "repositoryName": {
            "Ref": "RepositoryName"
          }
        }
      ]
    }
  }
]
```

ComprehendBasicAccessPolicy

授予偵測實體、關鍵片語、語言和情緒的權限。

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

```
    "Effect": "Allow",
    "Action": [
      "comprehend:BatchDetectKeyPhrases",
      "comprehend:DetectDominantLanguage",
      "comprehend:DetectEntities",
      "comprehend:BatchDetectEntities",
      "comprehend:DetectKeyPhrases",
      "comprehend:DetectSentiment",
      "comprehend:BatchDetectDominantLanguage",
      "comprehend:BatchDetectSentiment"
    ],
    "Resource": "*"
  }
]
```

CostExplorerReadOnlyPolicy

將帳單歷史記錄的唯讀 AWS Cost Explorer (Cost Explorer) API 授與唯讀權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ce:GetCostAndUsage",
      "ce:GetDimensionValues",
      "ce:GetReservationCoverage",
      "ce:GetReservationPurchaseRecommendation",
      "ce:GetReservationUtilization",
      "ce:GetTags"
    ],
    "Resource": "*"
  }
]
```

DynamoDBBackupFullAccessPolicy

為表格授與 DynamoDB 隨需備份的讀取和寫入權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "dynamodb:CreateBackup",
```

```
      "dynamodb:DescribeContinuousBackups"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
${tableName}",
        {
          "tableName": {
            "Ref": "TableName"
          }
        }
      ]
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "dynamodb:DeleteBackup",
      "dynamodb:DescribeBackup",
      "dynamodb:ListBackups"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
${tableName}/backup/*",
        {
          "tableName": {
            "Ref": "TableName"
          }
        }
      ]
    }
  }
]
```

DynamoDBCrudPolicy

將建立、讀取、更新和刪除權限授予 Amazon DynamoDB 表格。

```
"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

為 DynamoDB 資料表提供唯讀權限。

```
"Statement": [
  {
    "Effect": "Allow",
```

```
    "Action": [
      "dynamodb:GetItem",
      "dynamodb:Scan",
      "dynamodb:Query",
      "dynamodb:BatchGetItem",
      "dynamodb:DescribeTable"
    ],
    "Resource": [
      {
        "Fn::Sub": [
          "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
${tableName}",
          {
            "tableName": {
              "Ref": "TableName"
            }
          }
        ]
      },
      {
        "Fn::Sub": [
          "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
${tableName}/index/*",
          {
            "tableName": {
              "Ref": "TableName"
            }
          }
        ]
      }
    ]
  }
]
```

DynamoDBReconfigurePolicy

授與重新設定 DynamoDB 表格的權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "dynamodb:UpdateTable"
    ],
```

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

DynamoDBRestoreFromBackupPolicy

授與從備份還原 DynamoDB 資料表的權限。

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

DynamoDBStreamReadPolicy

授予描述和讀取 DynamoDB 串流和記錄的權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "dynamodb:DescribeStream",
      "dynamodb:GetRecords",
      "dynamodb:GetShardIterator"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
${tableName}/stream/${streamName}",
        {
          "tableName": {
            "Ref": "TableName"
          },
          "streamName": {
            "Ref": "StreamName"
          }
        }
      ]
    }
  },
```

```
  {
    "Effect": "Allow",
    "Action": [
      "dynamodb:ListStreams"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
${tableName}/stream/*",
        {
          "tableName": {
            "Ref": "TableName"
          }
        }
      ]
    }
  }
]
```

DynamoDBWritePolicy

將唯寫權限授與 DynamoDB 表格。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "dynamodb:PutItem",
      "dynamodb:UpdateItem",
      "dynamodb:BatchWriteItem"
    ],
    "Resource": [
      {
        "Fn::Sub": [
          "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
${tableName}",
          {
            "tableName": {
              "Ref": "TableName"
            }
          }
        ]
      },
      {
```

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

EC2CopyImagePolicy

授予複製 Amazon EC2 映像的權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ec2:CopyImage"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:ec2:${AWS::Region}:${AWS::AccountId}:image/${imageId}",
        {
          "imageId": {
            "Ref": "ImageId"
          }
        }
      ]
    }
  }
]
```

EC2DescribePolicy

授予描述亞馬遜彈性運算雲 (Amazon EC2) 執行個體的權限。

```
"Statement": [
```

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

EcsRunTaskPolicy

授與啟動任務定義新任務的權限。

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

EFSWriteAccessPolicy

授與使用寫入存取權來掛接 Amazon EFS 檔案系統的權限。

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

EKSDescribePolicy

授予描述或列出 Amazon Elastic Kubernetes Service (Amazon EKS) 叢集的權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "eks:DescribeCluster",
      "eks:ListClusters"
    ],
    "Resource": "*"
  }
```

```
]
```

ElasticMapReduceAddJobFlowStepsPolicy

授與將新步驟新增至執行中叢集的權限。

```
"Statement": [
  {
    "Action": "elasticmapreduce:AddJobFlowSteps",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:
${AWS::AccountId}:cluster/${clusterId}",
        {
          "clusterId": {
            "Ref": "ClusterId"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

ElasticMapReduceCancelStepsPolicy

授與取消擱置中步驟或執行中叢集中步驟的權限。

```
"Statement": [
  {
    "Action": "elasticmapreduce:CancelSteps",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:
${AWS::AccountId}:cluster/${clusterId}",
        {
          "clusterId": {
            "Ref": "ClusterId"
          }
        }
      ]
    },
    "Effect": "Allow"
```

```
  }
]
```

ElasticMapReduceModifyInstanceFleetPolicy

授與列出叢集中執行個體叢集的詳細資料和修改容量的權限。

```
"Statement": [
  {
    "Action": [
      "elasticmapreduce:ModifyInstanceFleet",
      "elasticmapreduce:ListInstanceFleets"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:
${AWS::AccountId}:cluster/${clusterId}",
        {
          "clusterId": {
            "Ref": "ClusterId"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

ElasticMapReduceModifyInstanceGroupsPolicy

授與列出叢集中執行個體群組的詳細資料和修改設定的權限。

```
"Statement": [
  {
    "Action": [
      "elasticmapreduce:ModifyInstanceGroups",
      "elasticmapreduce:ListInstanceGroups"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:
${AWS::AccountId}:cluster/${clusterId}",
        {
```

```
          "clusterId": {
            "Ref": "ClusterId"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

ElasticMapReduceSetTerminationProtectionPolicy

授與設定叢集終止保護的權限。

```
"Statement": [
  {
    "Action": "elasticmapreduce:SetTerminationProtection",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:
${AWS::AccountId}:cluster/${clusterId}",
        {
          "clusterId": {
            "Ref": "ClusterId"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

ElasticMapReduceTerminateJobFlowsPolicy

授與關閉叢集的權限。

```
"Statement": [
  {
    "Action": "elasticmapreduce:TerminateJobFlows",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:
${AWS::AccountId}:cluster/${clusterId}",
```

```
        {
          "clusterId": {
            "Ref": "ClusterId"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

ElasticsearchHttpPostPolicy

給 POST 和 PUT 權限 Amazon OpenSearch 服務。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "es:ESHttpPost",
      "es:ESHttpPut"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:es:${AWS::Region}:${AWS::AccountId}:domain/
${domainName}/*",
        {
          "domainName": {
            "Ref": "DomainName"
          }
        }
      ]
    }
  }
]
```

EventBridgePutEventsPolicy

授予將事件發送到 Amazon 的許可 EventBridge。

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

```
    "Effect": "Allow",
    "Action": "events:PutEvents",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:events:${AWS::Region}:${AWS::AccountId}:event-bus/
${eventBusName}",
        {
          "eventBusName": {
            "Ref": "EventBusName"
          }
        }
      ]
    }
  }
]
```

FilterLogEventsPolicy

授與從指定 CloudWatch 記錄群組篩選記錄事件的權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "logs:FilterLogEvents"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:logs:${AWS::Region}:${AWS::AccountId}:log-group:
${logGroupName}:log-stream:*",
        {
          "logGroupName": {
            "Ref": "LogGroupName"
          }
        }
      ]
    }
  }
]
```

FirehoseCrudPolicy

授予建立、寫入、更新和刪除 Firehose 傳送串流的權限。

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

FirehoseWritePolicy

授予寫入 Firehose 交付串流的權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "firehose:PutRecord",
      "firehose:PutRecordBatch"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:firehose:${AWS::Region}:
${AWS::AccountId}:deliverystream/${deliveryStreamName}",
        {
          "deliveryStreamName": {
            "Ref": "DeliveryStreamName"
```

```
          }
        }
      ]
    }
  }
]
```

KinesisCrudPolicy

授予建立、發佈和刪除 Amazon Kinesis 串流的權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "kinesis:AddTagsToStream",
      "kinesis:CreateStream",
      "kinesis:DecreaseStreamRetentionPeriod",
      "kinesis:DeleteStream",
      "kinesis:DescribeStream",
      "kinesis:DescribeStreamSummary",
      "kinesis:GetShardIterator",
      "kinesis:IncreaseStreamRetentionPeriod",
      "kinesis:ListTagsForStream",
      "kinesis:MergeShards",
      "kinesis:PutRecord",
      "kinesis:PutRecords",
      "kinesis:SplitShard",
      "kinesis:RemoveTagsFromStream"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:kinesis:${AWS::Region}:${AWS::AccountId}:stream/
${streamName}",
        {
          "streamName": {
            "Ref": "StreamName"
          }
        }
      ]
    }
  }
]
```

KinesisStreamReadPolicy

授予列出和讀取 Amazon Kinesis 串流的權限。

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

KMSDecryptPolicy

授予使用 AWS Key Management Service (AWS KMS) 金鑰進行解密的權限。請注意，`keyId`必須是 AWS KMS 金鑰 ID，而不是金鑰別名。

```
"Statement": [
  {
    "Action": "kms:Decrypt",
    "Effect": "Allow",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:kms:${AWS::Region}:${AWS::AccountId}:key/${keyId}",
        {
          "keyId": {
            "Ref": "KeyId"
          }
        }
      ]
    }
  }
]
```

KMSEncryptPolicy

授予使用密 AWS KMS 鑰進行加密的權限。請注意，keyID 必須是 AWS KMS 金鑰 ID，而不是金鑰別名。

```
"Statement": [
  {
    "Action": "kms:Encrypt",
    "Effect": "Allow",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:kms:${AWS::Region}:${AWS::AccountId}:key/${keyId}",
        {
          "keyId": {
            "Ref": "KeyId"
          }
        }
      ]
    }
  }
]
```

LambdaInvokePolicy

授與叫用 AWS Lambda 函數、別名或版本的權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "lambda:InvokeFunction"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:lambda:${AWS::Region}:${AWS::AccountId}:function:
${functionName}*",
        {
          "functionName": {
            "Ref": "FunctionName"
          }
        }
      ]
    }
  }
]
```

MobileAnalyticsWriteOnlyAccessPolicy

提供唯寫權限，以便為所有應用程式資源放置事件資料。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "mobileanalytics:PutEvents"
    ],
    "Resource": "*"
  }
]
```

OrganizationsListAccountsPolicy

授與列出子帳號名稱和 ID 的唯讀權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
```

```
      "organizations:ListAccounts"
    ],
    "Resource": "*"
  }
]
```

PinpointEndpointAccessPolicy

授予取得和更新 Amazon 精確應用程式端點的權限。

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

PollyFullAccessPolicy

授予對 Amazon Polly 詞典資源的完全訪問權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "polly:GetLexicon",
      "polly:DeleteLexicon"
    ],
```

```
    "Resource": [
      {
        "Fn::Sub": [
          "arn:${AWS::Partition}:polly:${AWS::Region}:${AWS::AccountId}:lexicon/
${lexiconName}",
          {
            "lexiconName": {
              "Ref": "LexiconName"
            }
          }
        ]
      }
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "polly:DescribeVoices",
      "polly:ListLexicons",
      "polly:PutLexicon",
      "polly:SynthesizeSpeech"
    ],
    "Resource": [
      {
        "Fn::Sub": "arn:${AWS::Partition}:polly:${AWS::Region}:
${AWS::AccountId}:lexicon/*"
      }
    ]
  }
]
```

RekognitionDetectOnlyPolicy

授予檢測臉部，標籤和文本的權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "rekognition:DetectFaces",
      "rekognition:DetectLabels",
      "rekognition:DetectModerationLabels",
      "rekognition:DetectText"
    ],
```

```
    "Resource": "*"
  }
]
```

RekognitionFacesManagementPolicy

授予在 Amazon Rekognition 集合中新增、刪除和搜尋臉孔的權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "rekognition:IndexFaces",
      "rekognition:DeleteFaces",
      "rekognition:SearchFaces",
      "rekognition:SearchFacesByImage",
      "rekognition:ListFaces"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:rekognition:${AWS::Region}:${AWS::AccountId}:collection/
${collectionId}",
        {
          "collectionId": {
            "Ref": "CollectionId"
          }
        }
      ]
    }
  }
]
```

RekognitionFacesPolicy

授予比較和檢測臉部和標籤的權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "rekognition:CompareFaces",
      "rekognition:DetectFaces"
    ],
```

```
    "Resource": "*"
  }
]
```

RekognitionLabelsPolicy

授予檢測對象和協調標籤的權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "rekognition:DetectLabels",
      "rekognition:DetectModerationLabels"
    ],
    "Resource": "*"
  }
]
```

RekognitionNoDataAccessPolicy

授予比較和檢測臉部和標籤的權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "rekognition:CompareFaces",
      "rekognition:DetectFaces",
      "rekognition:DetectLabels",
      "rekognition:DetectModerationLabels"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:rekognition:${AWS::Region}:${AWS::AccountId}:collection/
${collectionId}",
        {
          "collectionId": {
            "Ref": "CollectionId"
          }
        }
      ]
    }
```

```
  }
]
```

RekognitionReadPolicy

授予列出和搜索面孔的權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "rekognition:ListCollections",
      "rekognition:ListFaces",
      "rekognition:SearchFaces",
      "rekognition:SearchFacesByImage"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:rekognition:${AWS::Region}:${AWS::AccountId}:collection/
${collectionId}",
        {
          "collectionId": {
            "Ref": "CollectionId"
          }
        }
      ]
    }
  }
]
```

RekognitionWriteOnlyAccessPolicy

授予建立集合和索引臉孔的權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "rekognition:CreateCollection",
      "rekognition:IndexFaces"
    ],
    "Resource": {
      "Fn::Sub": [
```

```
        "arn:${AWS::Partition}:rekognition:${AWS::Region}:${AWS::AccountId}:collection/
${collectionId}",
        {
          "collectionId": {
            "Ref": "CollectionId"
          }
        }
      ]
    }
  }
]
```

Route53ChangeResourceRecordSetsPolicy

授予在 Route 53 中變更資源記錄集的權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "route53:ChangeResourceRecordSets"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:route53:::hostedzone/${HostedZoneId}",
        {
          "HostedZoneId": {
            "Ref": "HostedZoneId"
          }
        }
      ]
    }
  }
]
```

S3CrudPolicy

授予建立、讀取、更新和刪除權限，以便對 Amazon S3 儲存貯體中的物件執行動作。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
```

```
      "s3:GetObject",
      "s3:ListBucket",
      "s3:GetBucketLocation",
      "s3:GetObjectVersion",
      "s3:PutObject",
      "s3:PutObjectAcl",
      "s3:GetLifecycleConfiguration",
      "s3:PutLifecycleConfiguration",
      "s3:DeleteObject"
    ],
    "Resource": [
      {
        "Fn::Sub": [
          "arn:${AWS::Partition}:s3:::${bucketName}",
          {
            "bucketName": {
              "Ref": "BucketName"
            }
          }
        ]
      },
      {
        "Fn::Sub": [
          "arn:${AWS::Partition}:s3:::${bucketName}/*",
          {
            "bucketName": {
              "Ref": "BucketName"
            }
          }
        ]
      }
    ]
  }
]
```

S3FullAccessPolicy

授予對 Amazon S3 儲存貯體中物件執行動作的完整存取權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
```

```
      "s3:GetObjectAcl",
      "s3:GetObjectVersion",
      "s3:PutObject",
      "s3:PutObjectAcl",
      "s3:DeleteObject",
      "s3:DeleteObjectTagging",
      "s3:DeleteObjectVersionTagging",
      "s3:GetObjectTagging",
      "s3:GetObjectVersionTagging",
      "s3:PutObjectTagging",
      "s3:PutObjectVersionTagging"
    ],
    "Resource": [
      {
        "Fn::Sub": [
          "arn:${AWS::Partition}:s3:::${bucketName}/*",
          {
            "bucketName": {
              "Ref": "BucketName"
            }
          }
        ]
      }
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "s3:ListBucket",
      "s3:GetBucketLocation",
      "s3:GetLifecycleConfiguration",
      "s3:PutLifecycleConfiguration"
    ],
    "Resource": [
      {
        "Fn::Sub": [
          "arn:${AWS::Partition}:s3:::${bucketName}",
          {
            "bucketName": {
              "Ref": "BucketName"
            }
          }
        ]
      }
```

```
    ]
  }
]
```

S3ReadPolicy

授予讀取 Amazon Simple Storage Service (Amazon S3) 儲存貯體中物件的唯讀權限。

```
"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

授予將物件寫入 Amazon S3 儲存貯體的寫入權限。

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

SageMakerCreateEndpointConfigPolicy

授與在中建立端點組態的權限 SageMaker。

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

```
    "Action": [
      "sagemaker:CreateEndpointConfig"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:sagemaker:${AWS::Region}:${AWS::AccountId}:endpoint-
config/${endpointConfigName}",
        {
          "endpointConfigName": {
            "Ref": "EndpointConfigName"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

SageMakerCreateEndpointPolicy

授與在中建立端點的權限 SageMaker。

```
"Statement": [
  {
    "Action": [
      "sagemaker:CreateEndpoint"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:sagemaker:${AWS::Region}:${AWS::AccountId}:endpoint/
${endpointName}",
        {
          "endpointName": {
            "Ref": "EndpointName"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

ServerlessRepoReadWriteAccessPolicy

授予在 AWS Serverless Application Repository (AWS SAM) 服務中建立及列出應用程式的權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "serverlessrepo:CreateApplication",
      "serverlessrepo:CreateApplicationVersion",
      "serverlessrepo:GetApplication",
      "serverlessrepo:ListApplications",
      "serverlessrepo:ListApplicationVersions"
    ],
    "Resource": [
      {
        "Fn::Sub": "arn:${AWS::Partition}:serverlessrepo:${AWS::Region}:
${AWS::AccountId}:applications/*"
      }
    ]
  }
]
```

SESBulkTemplatedCrudPolicy

授予傳送 Amazon SES 電子郵件、範本化電子郵件和範本化大量電子郵件的權限，以及驗證身分。

> **Note**
>
> 此動`ses:SendTemplatedEmail`作需要範本 ARN。請改用 `SESBulkTemplatedCrudPolicy_v2`。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ses:GetIdentityVerificationAttributes",
      "ses:SendEmail",
      "ses:SendRawEmail",
      "ses:SendTemplatedEmail",
```

```
      "ses:SendBulkTemplatedEmail",
      "ses:VerifyEmailIdentity"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:ses:${AWS::Region}:${AWS::AccountId}:identity/
${identityName}",
        {
          "identityName": {
            "Ref": "IdentityName"
          }
        }
      ]
    }
  }
]
```

SESBulkTemplatedCrudPolicy_v2

授予傳送 Amazon SES 電子郵件、範本化電子郵件和範本化大量電子郵件的權限，以及驗證身分。

```
"Statement": [
  {
    "Action": [
      "ses:SendEmail",
      "ses:SendRawEmail",
      "ses:SendTemplatedEmail",
      "ses:SendBulkTemplatedEmail"
    ],
    "Effect": "Allow",
    "Resource": [
      {
        "Fn::Sub": [
          "arn:${AWS::Partition}:ses:${AWS::Region}:${AWS::AccountId}:identity/
${identityName}",
          {
            "identityName": {
              "Ref": "IdentityName"
            }
          }
        ]
      },
      {
        "Fn::Sub": [
```

```
          "arn:${AWS::Partition}:ses:${AWS::Region}:${AWS::AccountId}:template/
${templateName}",
          {
            "templateName": {
              "Ref": "TemplateName"
            }
          }
        ]
      }
    ]
  },
  {
    "Action": [
      "ses:GetIdentityVerificationAttributes",
      "ses:VerifyEmailIdentity"
    ],
    "Effect": "Allow",
    "Resource": "*"
  }
]
```

SESCrudPolicy

授予發送電子郵件和驗證身份的權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ses:GetIdentityVerificationAttributes",
      "ses:SendEmail",
      "ses:SendRawEmail",
      "ses:VerifyEmailIdentity"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:ses:${AWS::Region}:${AWS::AccountId}:identity/
${identityName}",
        {
          "identityName": {
            "Ref": "IdentityName"
          }
        }
      ]
```

```
    }
  }
]
```

SESEmailTemplateCrudPolicy

授予建立、取得、列出、更新和刪除 Amazon SES 電子郵件範本的權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ses:CreateTemplate",
      "ses:GetTemplate",
      "ses:ListTemplates",
      "ses:UpdateTemplate",
      "ses:DeleteTemplate",
      "ses:TestRenderTemplate"
    ],
    "Resource": "*"
  }
]
```

SESSendBouncePolicy

SendBounce 授予亞馬遜簡易電子郵件服務 (Amazon SES) 身分識別的權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ses:SendBounce"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:ses:${AWS::Region}:${AWS::AccountId}:identity/
${identityName}",
        {
          "identityName": {
            "Ref": "IdentityName"
          }
        }
      ]
```

```
    }
  }
]
```

SNSCrudPolicy

授予建立、發佈和訂閱 Amazon SNS 主題的權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "sns:ListSubscriptionsByTopic",
      "sns:CreateTopic",
      "sns:SetTopicAttributes",
      "sns:Subscribe",
      "sns:Publish"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:sns:${AWS::Region}:${AWS::AccountId}:${topicName}*",
        {
          "topicName": {
            "Ref": "TopicName"
          }
        }
      ]
    }
  }
]
```

SNSPublishMessagePolicy

授予將訊息發佈至亞馬遜簡單通知服務 (Amazon SNS) 主題的權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "sns:Publish"
    ],
    "Resource": {
```

```
      "Fn::Sub": [
        "arn:${AWS::Partition}:sns:${AWS::Region}:${AWS::AccountId}:${topicName}",
        {
          "topicName": {
            "Ref": "TopicName"
          }
        }
      ]
    }
  }
]
```

SQSPollerPolicy

授予輪詢 Amazon Simple Queue Service (Amazon SQS) 佇列的權限。

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

SQSSendMessagePolicy

授予將訊息傳送至 Amazon SQS 佇列的權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "sqs:SendMessage*"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:sqs:${AWS::Region}:${AWS::AccountId}:${queueName}",
        {
          "queueName": {
            "Ref": "QueueName"
          }
        }
      ]
    }
  }
]
```

SSMParameterReadPolicy

授予從 Amazon EC2 Systems Manager (SSM) 參數存放區存取參數的權限，以便在此帳戶中載入機密。當參數名稱沒有斜線前綴時使用。

> **Note**
>
> 如果您不使用預設金鑰，您也需要原KMSDecryptPolicy則。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ssm:DescribeParameters"
    ],
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "ssm:GetParameters",
      "ssm:GetParameter",
```

```
      "ssm:GetParametersByPath"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:ssm:${AWS::Region}:${AWS::AccountId}:parameter/
${parameterName}",
        {
          "parameterName": {
            "Ref": "ParameterName"
          }
        }
      ]
    }
  }
]
```

SSMParameterWithSlashPrefixReadPolicy

授予從 Amazon EC2 Systems Manager (SSM) 參數存放區存取參數的權限，以便在此帳戶中載入機密。當參數名稱具有斜線前綴時使用。

> **Note**
>
> 如果您不使用預設金鑰，您也需要原KMSDecryptPolicy則。

```
"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

授予啟動 Step Functions 狀態機器執行的權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "states:StartExecution"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:states:${AWS::Region}:${AWS::AccountId}:stateMachine:
${stateMachineName}",
        {
          "stateMachineName": {
            "Ref": "StateMachineName"
          }
        }
      ]
    }
  }
]
```

TextractDetectAnalyzePolicy

允許訪問以檢測和分析使用亞馬遜文本提取的文檔。

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

```
    "Effect": "Allow",
    "Action": [
      "textract:DetectDocumentText",
      "textract:StartDocumentTextDetection",
      "textract:StartDocumentAnalysis",
      "textract:AnalyzeDocument"
    ],
    "Resource": "*"
  }
]
```

TextractGetResultPolicy

允許從 Amazon Textract 獲取檢測到和分析的文檔的訪問權限。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "textract:GetDocumentTextDetection",
      "textract:GetDocumentAnalysis"
    ],
    "Resource": "*"
  }
]
```

TextractPolicy

可以完全訪問 Amazon Textract。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "textract:*"
    ],
    "Resource": "*"
  }
]
```

VPCAccessPolicy

提供建立、刪除、描述和卸離彈性網路介面的存取權。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ec2:CreateNetworkInterface",
      "ec2:DeleteNetworkInterface",
      "ec2:DescribeNetworkInterfaces",
      "ec2:DetachNetworkInterface"
    ],
    "Resource": "*"
  }
]
```

使用 AWS CloudFormation 機制管理權限

若要控制對 AWS 資源的存取， AWS Serverless Application Model (AWS SAM) 可以使用與 AWS CloudFormation. 若要取得更多資訊，請參閱《使用指南》 AWS Identity and Access Management中的AWS CloudFormation 〈控制存取〉。

授與使用者管理無伺服器應用程式的權限有三個主要選項。每個選項都為使用者提供不同層級的存取控制。

- 授予管理員權限。
- 附加必要的 AWS 受管理策略。
- 授予特定 AWS Identity and Access Management (IAM) 許可。

根據您選擇的選項，使用者只能管理包含其有權存取之 AWS 資源的無伺服器應用程式。

以下各節將詳細說明每個選項。

授予管理員權限

如果您將管理員權限授與使用者，他們可以管理包含任何 AWS 資源組合的無伺服器應用程式。這是最簡單的選項，但它也會授與使用者最廣泛的權限集，因此可讓使用者執行影響最高的動作。

如需有關授與管理員權限給使用者的詳細資訊，請參閱 IAM 使用者指南中的建立您的第一個 IAM 管理員使用者和群組。

附加必要的 AWS 受管政策

您可以使用AWS 受管理的策略授與使用者權限子集，而不是授與完整的管理員權限。如果您使用此選項，請確定 AWS 受管理的策略集涵蓋了使用者管理的無伺服器應用程式所需的所有動作和資源。

例如，下列 AWS 受管理的原則足以部署範例 Hello World 應用程式：

- AWSCloudFormationFullAccess
- IAM FullAccess
- AWSLambda_FullAccess
- 亞馬遜 API GatewayAdministrator
- 亞馬遜 3 FullAccess
- 亚马逊 ContainerRegistryFullAccess

如需將政策附加至 IAM 使用者的相關資訊，請參閱 IAM 使用者指南中的變更 IAM 使用者的許可。

授予特定的 IAM 許可

對於最精細的存取控制層級，您可以使用政策陳述式將特定的 IAM 許可授予使用者。如果您使用此選項，請確定原則陳述式包含使用者管理的無伺服器應用程式所需的所有動作和資源。

此選項的最佳做法是拒絕使用者建立角色 (包括 Lambda 執行角色) 的權限，以便他們無法授與自己提升的權限。因此，身為管理員的您必須先建立 Lambda 執行角色，該角色將在使用者將管理的無伺服器應用程式中指定。如需建立 Lambda 執行角色的相關資訊，請參閱在 IAM 主控台中建立執行角色。

對於示例 Hello World 應用程序 AWSLambdaBasicExecutionRole，足以運行應用程序。建立 Lambda 執行角色之後，請修改範例 Hello World 應用程式的範本檔案，將下列屬性新增至`AWS::Serverless::Function`資源： AWS SAM

```
  Role: lambda-execution-role-arn
```

修改後的 Hello World 應用程式，下列原則陳述式會授予使用者部署、更新及刪除應用程式的足夠權限：

```
{
    "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**
>
> 本節中的範例原則陳述式授與足夠權限，讓您可以部署、更新和刪除範例 Hello World 應用程式。如果您在應用程式中新增其他資源類型，則需要更新政策陳述式以包含下列項目：
>
> 1. 您的應用程式呼叫服務動作的權限。
> 2. 服務主體 (如果需要服務的動作)。
>
> 例如，如果您新增「Step Functions」工作流程，您可能需要為此處列出的動作及`states.amazonaws.com`服務主體新增權限。

如需 IAM 政策的詳細資訊，請參閱 IAM 使用者指南中的管理 IAM 政策。

使用 AWS SAM 範本控制 API 存取

控制 API Gateway API 的存取有助於確保您的無伺服器應用程式安全無虞，而且只能透過您啟用的授權存取。您可以在 AWS SAM 範本中啟用授權，以控制誰可以存取您的 API Gateway API。

AWS SAM 支援數種控制 API Gateway API 存取的機制。`AWS::Serverless::HttpApi`和`AWS::Serverless::Api`資源類型之間的支援機制集不同。

下表摘要說明每個資源類型支援的機制。

控制存取的機制	AWS::Serverless::HttpApi	AWS::Serverless::Api
Lambda 授權人	✓	✓
IAM 許可		✓
Amazon Cognito 使用者集區	✓ *	✓
API 金鑰		✓
資源政策		✓
OAuth 2.0/JWT 授權者	✓	

* 您可以使用 Amazon Cognito 做為具有`AWS::Serverless::HttpApi`資源類型的 JSON 網頁權杖 (JWT) 發行者。

- Lambda 授權者 — Lambda 授權者 (以前稱為自訂授權者) 是您提供的 Lambda 函數，可供您控制 API 的存取。呼叫 API 時，會使用要求內容或用戶端應用程式提供的授權權杖來叫用此 Lambda 函數。Lambda 函數會回應呼叫者是否獲授權執行要求的作業。

  `AWS::Serverless::HttpApi`和`AWS::Serverless::Api`資源類型都支援 Lambda 授權者。

  如需使用 Lambda 授權者的詳細資訊`AWS::Serverless::HttpApi`，請參閱 API Gateway 開發人員指南中的使用 HTTP API 的 AWS Lambda 授權人。如需使用 Lambda 授權者的詳細資訊`AWS::Serverless::Api`，請參閱 API Gateway 開發人員指南中的使用 API Gateway Lambda 授權器。

  如需任一資源類型的 Lambda 授權者範例，請參閱Lambda 授權者範例。

- IAM 許可 — 您可以控制誰可以使用 AWS Identity and Access Management (IAM) 許可調用您的 API。呼叫您 API 的使用者必須使用 IAM 登入資料進行驗證。只有當 IAM 政策附加到代表 API 呼叫者的 IAM 使用者、包含該使用者的 IAM 群組或使用者假設的 IAM 角色時，呼叫您的 API 才會成功。

  只有資`AWS::Serverless::Api`源類型支援 IAM 許可。

  如需詳細資訊，請參閱《API Gateway 開發人員指南》中的使用 IAM 許可控制 API 的存取。如需範例，請參閱IAM 權限範例。

- Amazon Cognito 使用者集區 — Amazon Cognito 使用者集區是 Amazon Cognito 中的使用者目錄。API 的客戶端必須首先將用戶登錄到用戶池，並獲取用戶的身份或訪問令牌。然後，客戶端使用返回的令牌之一調用您的 API。只有在必要的權杖有效時，API 呼叫才會成功。

  資`AWS::Serverless::Api`源類型支援 Amazon Cognito 使用者集區。
  資`AWS::Serverless::HttpApi`源類型支援使用 Amazon Cognito 做為 JWT 發行者。

  如需詳細資訊，請參閱《API Gateway 開發人員指南》中的使用 Amazon Cognito 使用者集區作為授權方來控制對 REST API 的存取。如需範例，請參閱Amazon Cognito 用戶池示例。

- API 金鑰 — API 金鑰是您分發給應用程式開發人員客戶的英數字串值，以授予 API 存取權。

  只有資`AWS::Serverless::Api`源類型支援 API 金鑰。

如需 API 金鑰的詳細資訊，請參閱《API Gateway 開發人員指南》中的「建立和使用 API 金鑰」的使用計劃。如需 API 金鑰的範例，請參閱API 金鑰範例。

- 資源策略 — 資源策略是您可以附加到 API Gateway API 的 JSON 政策文件。使用資源政策來控制指定的主體 (通常是 IAM 使用者或角色) 是否可以叫用 API。

  只有資`AWS::Serverless::Api`源類型支援資源原則，做為控制 API Gateway API 存取的機制。

  如需有關資源政策的詳細資訊，請參閱《API Gateway 開發人員指南》中的使用 API Gateway 資源政策控制 API 的存取。如需資源策略的範例，請參閱資源政策範例。

- OAuth 2.0/JWT 授權者 — 您可以使用 JWT 作為 OpenID Connect（OIDC）和 O Auth 2.0 框架的一部分來控制對 API 的訪問。API Gateway 會驗證用戶端透過 API 要求提交的 JWT，並根據權杖驗證和權杖中的範圍 (選擇性) 允許或拒絕要求。

  只有`AWS::Serverless::HttpApi`資源類型支持 OAuth 2.0/JWT 授權者。

  如需詳細資訊，請參閱《API Gateway 開發人員指南》中的使用 JWT 授權方控制對 HTTP API 的存取。如需範例，請參閱OAuth 2.0/JWT 授權者示例。

選擇控制存取的機制

您選擇用來控制 API Gateway API 存取的機制取決於幾個因素。例如，如果您的綠地專案未設定授權或存取控制，則 Amazon Cognito 使用者集區可能是您的最佳選擇。這是因為當您設定使用者集區時，您也會自動設定驗證和存取控制。

不過，如果您的應用程式已設定驗證，則使用 Lambda 授權者可能是您的最佳選擇。這是因為您可以呼叫現有的驗證服務，並根據回應傳回原則文件。此外，如果您的應用程式需要使用者集區不支援的自訂驗證或存取控制邏輯，則 Lambda 授權者可能是您的最佳選擇。

選擇要使用的機制之後，請參閱中範例的對應章節，瞭解如何使用 AWS SAM 來設定應用程式以使用該機制。

自訂錯誤回應

您可以使 AWS SAM 用自訂某些 API Gateway 錯誤回應的內容。只有資`AWS::Serverless::Api`源類型支援自訂的 API Gateway 回應。

如需 API Gateway 回應的詳細資訊，請參閱 API Gateway 開發人員指南中的 API Gateway 回應。如需自訂回應的範例，請參閱客製化回應範例。

範例

- Lambda 授權者範例
- IAM 權限範例
- Amazon Cognito 用戶池示例
- API 金鑰範例
- 資源政策範例
- OAuth 2.0/JWT 授權者示例
- 客製化回應範例

Lambda 授權者範例

資AWS::Serverless::Api源類型支援兩種類型的 Lambda 授權者：授權者和TOKENREQUEST授權者。資AWS::Serverless::HttpApi源類型僅支援REQUEST授權者。以下是每種類型的範例。

Lambda **TOKEN** 授權者範例 () AWS::Serverless::Api

您可以在 AWS SAM 範本中定義 Lambda TOKEN 授權者，以控制 API 的存取權限。要做到這一點，你使用的ApiAuth數據類型。

以下是 Lambda TOKEN 授權者的 AWS SAM 範例範本區段：

> **Note**
>
> 在下列範例中，會隱含產生 SAM FunctionRole。

```
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
```

如需有關 Lambda 授權人員的詳細資訊，請參閱 API Gateway 開發人員指南中的使用 API Gateway Lambda 授權器。

Lambda **REQUEST** 授權者範例 () AWS::Serverless::Api

您可以在 AWS SAM 範本中定義 Lambda REQUEST 授權者，以控制 API 的存取權限。要做到這一點，你使用的ApiAuth數據類型。

以下是 Lambda REQUEST 授權者的 AWS SAM 範例範本區段：

```
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
```

如需有關 Lambda 授權人員的詳細資訊，請參閱 API Gateway 開發人員指南中的使用 API Gateway Lambda 授權器。

Lambda 授權者範例 () AWS::Serverless::HttpApi

您可以在 AWS SAM 範本中定義 Lambda 授權者，以控制對 HTTP API 的存取權限。要做到這一點，你使用的HttpApiAuth數據類型。

以下是 Lambda 授權者的 AWS SAM 範例範本區段：

```
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 權限範例

您可以在 AWS SAM 範本中定義 IAM 許可來控制 API 的存取權限。要做到這一點，你使用的ApiAuth數據類型。

以下是 IAM 許可使用的 AWS SAM 範例範本：

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
  MyApi:
    Type: AWS::Serverless::Api
```

```
    Properties:
      StageName: Prod
      Description: 'API with IAM authorization'
      Auth:
        DefaultAuthorizer: AWS_IAM #sets AWS_IAM auth for all methods in this API
  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.handler
      Runtime: python3.10
      Events:
        GetRoot:
          Type: Api
          Properties:
            RestApiId: !Ref MyApi
            Path: /
            Method: get
      InlineCode: |
        def handler(event, context):
          return {'body': 'Hello World!', 'statusCode': 200}
```

如需 IAM 許可的詳細資訊，請參閱 API Gateway 開發人員指南中的控制叫用 API 的存取權限。

Amazon Cognito 用戶池示例

您可以透過在 AWS SAM 範本中定義 Amazon Cognito 使用者集區來控制對 API 的存取。要做到這一點，你使用的ApiAuth數據類型。

以下是使用者集區的 AWS SAM 範例範本區段：

```
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
```

如需有關 Amazon Cognito 使用者集區的詳細資訊，請參閱 API Gateway 開發人員指南中的使用 Amazon Cognito 使用者集區做為授權者控制 REST API 的存取。

API 金鑰範例

您可以在 AWS SAM 範本中要求 API 金鑰來控制 API 的存取權限。要做到這一點，你使用的ApiAuth數據類型。

以下是 API 密鑰的示例 AWS SAM 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
```

如需 API 金鑰的詳細資訊，請參閱《API Gateway 開發人員指南》中的「建立和使用 API 金鑰」的使用計劃。

資源政策範例

您可以在 AWS SAM 範本中附加資源策略來控制 API 的存取。要做到這一點，你使用的ApiAuth數據類型。

以下是私有 API 的 AWS SAM 範例範本。私有 API 必須具有資源策略才能部署。

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
  MyPrivateApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      EndpointConfiguration: PRIVATE  # Creates a private API. Resource policies are
 required for all private APIs.
```

```
      Auth:
        ResourcePolicy:
          CustomStatements: {
              Effect: 'Allow',
              Action: 'execute-api:Invoke',
              Resource: ['execute-api:/*/*/*'],
              Principal: '*'
            }
  MyFunction:
    Type: 'AWS::Serverless::Function'
    Properties:
      InlineCode: |
        def handler(event, context):
          return {'body': 'Hello World!', 'statusCode': 200}
      Handler: index.handler
      Runtime: python3.10
      Events:
        AddItem:
          Type: Api
          Properties:
            RestApiId:
              Ref: MyPrivateApi
            Path: /
            Method: get
```

如需有關資源政策的詳細資訊，請參閱《API Gateway 開發人員指南》中的使用 API Gateway 資源政策控制 API 的存取。如需有關私有 API 的詳細資訊，請參閱 API Gateway 開發人員指南中的在 Amazon API G ateway 道中建立私有 API。

OAuth 2.0/JWT 授權者示例

您可以使用 JWT 作為 OpenID Connect（OIDC）和 O Auth 2.0 框架的一部分來控制對 API 的訪問。要做到這一點，你使用的HttpApiAuth數據類型。

以下是 OAuth 2.0/JWT 授權者的 AWS SAM 範例範本區段：

```
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
```

如需 OAuth 2.0/JWT 授權者的詳細資訊，請參閱《API Gateway 開發人員指南》中的使用 JWT 授權人員控制 HTTP API 的存取。

客製化回應範例

您可以在 AWS SAM 範本中定義回應標頭，以自訂某些 API Gateway 錯誤回應。要執行此操作，請使用「閘道響應對象」數據類型。

以下是建立DEFAULT_5XX錯誤自訂回應的 AWS SAM 範例範本。

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
  MyApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      GatewayResponses:
```

```
        DEFAULT_5XX:
          ResponseParameters:
            Headers:
              Access-Control-Expose-Headers: "'WWW-Authenticate'"
              Access-Control-Allow-Origin: "'*'"
              ErrorHeader: "'MyCustomErrorHeader'"
          ResponseTemplates:
            application/json: "{\"message\": \"Error on the $context.resourcePath
resource\" }"

  GetFunction:
    Type: AWS::Serverless::Function
    Properties:
      Runtime: python3.10
      Handler: index.handler
      InlineCode: |
        def handler(event, context):
          raise Exception('Check out the new response!')
      Events:
        GetResource:
          Type: Api
          Properties:
            Path: /error
            Method: get
            RestApiId: !Ref MyApi
```

如需 API Gateway 回應的詳細資訊，請參閱 API Gateway 開發人員指南中的 API Gateway 回應。

使用 Lambda 層來提高效率 AWS SAM

使用 AWS SAM，您可以在無伺服器應用程式中包含層。 AWS Lambda 層可讓您從 Lambda 函數擷取程式碼至 Lambda 層，然後可用於多個 Lambda 函數。這樣做可以減少部署套件的大小、將核心函數邏輯與相依性分開，以及在多個函式之間共用相依性。如需有關圖層的詳細資訊，請參閱AWS Lambda 開發人員指南中的 Lambda 層。

本主題提供下列相關資訊：

- 在應用程式中包含圖層
- 如何在本機快取圖層

如需建置自訂圖層的資訊，請參閱建置 Lambda 層。

在應用程式中包含圖層

若要在應用程式中包含圖層，請使用AWS::Serverless::Function資源類型的Layers屬性。

以下是包含 Lambda 函數的 AWS SAM 範例範本，其中包含圖層：

```
ServerlessFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: .
    Handler: my_handler
    Runtime: Python3.7
    Layers:
        - <LayerVersion ARN>
```

如何在本機快取圖層

當您使用其中一個sam local命令叫用函數時，函數的 Layer 套件會下載並快取到本機主機上。

下表顯示不同作業系統的預設快取目錄位置。

作業系統	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

快取套件之後，會將圖層 AWS SAMCLI疊加到用來叫用函數的 Docker 影像上。會 AWS SAMCLI產生它所建置的影像名稱，以及保留在快取中的影像名稱。 LayerVersions 您可以在以下各節中找到有關結構描述的更多詳細資訊。

要檢查覆蓋的圖層，請執行以下命令以在要檢查的圖像中啟動 bash 會話：

```
docker run -it --entrypoint=/bin/bash samcli/lambda:<Tag following the schema outlined
 in Docker Image Tag Schema> -i
```

圖層快取目錄名稱綱要

鑑於您 LayerVersionArn 的模板中定義了一個，從 ARN 中 AWS SAMCLI提取 LayerName 和版本。它創建一個目錄，以將圖層內容放置在命名中`LayerName-Version-<first 10 characters of sha256 of ARN>`。

範例：

```
ARN = arn:aws:lambda:us-west-2:111111111111:layer:myLayer:1
Directory name = myLayer-1-926eeb5ff1
```

碼頭圖片標籤架構

若要計算唯一圖層雜湊值，請將所有唯一的圖層名稱與「-」分隔符合使用，取 SHA256 雜湊值，然後取出前 10 個字元。

範例：

```
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
```

唯一名稱的計算方式與「層級快取目錄」名稱綱要相同：

```
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
```

要計算唯一層哈希值，請將所有唯一圖層名稱與 '-' 的分隔符合併，取 sha256 哈希值，然後取前 25 個字符：

```
myLayer-1-926eeb5ff1-mySecondLayer-1-6bc1022bdf = 2dd7ac5ffb30d515926aef
```

然後將此值與函數的運行時間和體系結構相結合，並使用分隔符 '-'：

```
python3.7-x86_64-2dd7ac5ffb30d515926aefffd
```

使用嵌套應用程序重複使用代碼和資源 AWS SAM

無伺服器應用程式可以包含一或多個巢狀應用程式。巢狀應用程式是較大應用程式的一部分，可以封裝和部署為獨立的人工因素，或做為大型應用程式的元件。巢狀應用程式可讓您將常用的程式碼轉換成自己的應用程式，然後在較大的無伺服器應用程式或多個無伺服器應用程式中重複使用。

隨著無伺服器架構的成長，通常會出現常見的模式，在多個應用程式範本中定義相同的元件。巢狀應用程式可讓您在不同的 AWS SAM 範本中重複使用通用程式碼、功能、資源和組態，讓您只能維護來自單一來源的程式碼。這減少了重複的代碼和配置。此外，這種模組化方法可簡化開發、強化程式碼組織，並促進無伺服器應用程式之間的一致性。使用巢狀應用程式，您可以更專注於應用程式獨有的商務邏輯。

若要在無伺服器應用程式中定義巢狀應用程式，請使用資AWS::Serverless::Application源類型。

您可以從下列兩個來源定義巢狀應用程式：

- 應用AWS Serverless Application Repository 程式 — 您可以使用中帳戶可用的應用程式來定義巢狀應用程式 AWS Serverless Application Repository。這些應用程式可以是您帳戶中的私人應用程式、與您的帳戶私下共用的應用程式，或是在中公開共用的應用程式 AWS Serverless Application Repository。如需有關不同部署權限層級的詳細資訊，請參閱AWS Serverless Application Repository 開發人員指南中的應用程式部署權限和發佈應用程式。
- 本機應用程式 — 您可以使用儲存在本機檔案系統上的應用程式來定義巢狀應用程式。

如需如何在無伺服器應用程式中使 AWS SAM 用定義這兩種巢狀應用程式類型的詳細資訊，請參閱下列章節。

> **Note**
>
> 無伺服器應用程式中可以巢狀化的應用程式數目上限為 200 個。
> 巢狀應用程式可以擁有的參數上限為 60 個。

定義巢狀應用程式 AWS Serverless Application Repository

您可以使用中提供的應用程式來定義巢狀應用程式 AWS Serverless Application Repository。您也可以使用儲存和散發包含巢狀應用程式的應用程式 AWS Serverless Application Repository。若要檢閱中巢

狀應用程式的詳細資訊 AWS Serverless Application Repository，您可以使用 AWS SDK AWS CLI、或 Lambda 主控台。

若要定義託管在無伺服器應用程式 AWS SAM 範本 AWS Serverless Application Repository 中的應用程式，請使用每個 AWS Serverless Application Repository 應用程式詳細資料頁面上的「複製為 SAM 資源」按鈕。若要這麼做，請依照下列步驟進行：

1. 請確定您已登入 AWS Management Console。
2. 使用AWS Serverless Application Repository 開發人員指南中〈瀏覽、搜尋和部署應用程式〉一節中的步驟，尋找您要巢狀化的應用程式。 AWS Serverless Application Repository
3. 選擇「複製為 SAM 資源」按鈕。您正在檢視的應用程式的 SAM 範本區段現在位於剪貼簿中。
4. 針對您要在此應用程式中巢狀化的應用程式，將 SAM 範本`Resources:`區段貼到 SAM 範本檔案的區段中。

以下是託管於中的巢狀應用程式的 SAM 範本區段範例 AWS Serverless Application Repository：

```
Transform: AWS::Serverless-2016-10-31

Resources:
  applicationaliasname:
    Type: AWS::Serverless::Application
    Properties:
      Location:
        ApplicationId: arn:aws:serverlessrepo:us-
east-1:123456789012:applications/application-alias-name
        SemanticVersion: 1.0.0
      Parameters:
        # Optional parameter that can have default value overridden
        # ParameterName1: 15 # Uncomment to override default value
        # Required parameter that needs value to be provided
        ParameterName2: YOUR_VALUE
```

如果沒有必要的參數設定，您可以省略範本的`Parameters:`部分。

> **Important**
>
> 包含託管於中之巢狀應用程式的應用程式 AWS Serverless Application Repository 會繼承巢狀應用程式的共用限制。

例如，假設某個應用程式是公開共用的，但其中包含的巢狀應用程式僅與建立父應用程式的 AWS 帳戶私下共用。在這種情況下，如果您的 AWS 帳戶沒有部署巢狀應用程式的權限，您將無法部署父應用程式。如需部署應用程式權限的詳細資訊，請參閱AWS Serverless Application Repository 開發人員指南中的應用程式部署權限和發佈應用程式。

從本機檔案系統定義巢狀應用程式

您可以使用儲存在本機檔案系統上的應用程式來定義巢狀應用程式。您可以透過指定儲存在本機檔案系統上的 AWS SAM 範本檔案路徑來執行此操作。

以下是巢狀本機應用程式的 SAM 範本區段範例：

```
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
```

如果沒有參數設定，您可以省略範本的`Parameters:`部分。

部署巢狀應用

您可以使用 AWS SAMCLI指令來部署巢狀應用程式`sam deploy`。如需詳細資訊，請參閱部署您的應用程式和資源 AWS SAM。

> **Note**
>
> 當您部署包含巢狀應用程式的應用程式時，您必須確認。您可以通過將能力 _ 自動擴展傳遞給 CreateCloudFormationChangeSet API 或使用命令來完成此操作。`aws serverlessrepo create-cloud-formation-change-set` AWS CLI
> 如需確認巢狀應用程式的詳細資訊，請參閱AWS Serverless Application Repository 開發人員指南中的部署應用程式時確認 IAM 角色、資源政策和巢狀應用程式。

使用 EventBridge排程器管理基於時間的事件 AWS SAM

什麼是 Amazon EventBridge 調度程序？

Amazon EventBridge Scheduler 是一種排程服務，可讓您跨所有服務建立、啟動和管理數千萬個事件和任 AWS 務。此服務對於與時間有關的事件特別有用。您可以使用它來安排事件和基於時間的循環調用。它還支持一次性事件以及帶有開始和結束時間的速率和 chron 表達式。

若要進一步了解 Amazon EventBridge 排程器，請參閱什麼是 Amazon EventBridge 排程器？ 在「EventBridge 排程器使用指南」中。

主題

EventBridge 排程器支援 AWS SAM

AWS Serverless Application Model (AWS SAM) 範本規格提供簡單、簡短的語法，您可以使用這些語法來 EventBridge 排程與的 Scheduler 事件。 AWS Lambda AWS Step Functions

建立 EventBridge 排程器事件 AWS SAM

將內`ScheduleV2`容設定為 AWS SAM 範本中的事件類型，以定義您的 EventBridge Scheduler 事件。此內容支援`AWS::Serverless::Function`和`AWS::Serverless::StateMachine`資源類型。

```
MyFunction:
  Type: AWS::Serverless::Function
  Properties:
    Events:
      CWSchedule:
        Type: ScheduleV2
        Properties:
          ScheduleExpression: 'rate(1 minute)'
          Name: TestScheduleV2Function
          Description: Test schedule event
```

```
MyStateMachine:
  Type: AWS::Serverless::StateMachine
  Properties:
    Events:
      CWSchedule:
        Type: ScheduleV2
        Properties:
          ScheduleExpression: 'rate(1 minute)'
          Name: TestScheduleV2StateMachine
          Description: Test schedule event
```

EventBridge 排程器事件排程也支援未處理事件的無效字母佇列 (DLQ)。如需有關無效字母佇列的詳細資訊，請參閱《排程器使用指EventBridge 南》中的 < 設定 EventBridge 排程器的無效字母佇列 > 。

指定 DLQ ARN 時，會 AWS SAM 設定排程器排程的權限，以便將訊息傳送至 DLQ。當未指定 DLQ ARN 時， AWS SAM 將會建立 DLQ 資源。

範例

定義排 EventBridge 程器事件的基本範例 AWS SAM

```
Transform: AWS::Serverless-2016-10-31
Resources:
  MyLambdaFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.handler
      Runtime: python3.8
      InlineCode: |
        def handler(event, context):
            print(event)
            return {'body': 'Hello World!', 'statusCode': 200}
      MemorySize: 128
      Events:
        Schedule:
          Type: ScheduleV2
          Properties:
            ScheduleExpression: rate(1 minute)
            Input: '{"hello": "simple"}'

  MySFNFunction:
    Type: AWS::Serverless::Function
```

```
    Properties:
      Handler: index.handler
      Runtime: python3.8
      InlineCode: |
        def handler(event, context):
            print(event)
            return {'body': 'Hello World!', 'statusCode': 200}
      MemorySize: 128

  StateMachine:
    Type: AWS::Serverless::StateMachine
    Properties:
      Type: STANDARD
      Definition:
        StartAt: MyLambdaState
        States:
          MyLambdaState:
            Type: Task
            Resource: !GetAtt MySFNFunction.Arn
            End: true
      Policies:
        - LambdaInvokePolicy:
            FunctionName: !Ref MySFNFunction
      Events:
        Events:
        Schedule:
          Type: ScheduleV2
          Properties:
            ScheduleExpression: rate(1 minute)
            Input: '{"hello": "simple"}'
```

進一步了解

若要深入瞭解如何定義「ScheduleV2 EventBridge 排程器」屬性，請參閱：

- ScheduleV2對於AWS::Serverless::Function.
- ScheduleV2對於AWS::Serverless::StateMachine.

協調 AWS 資源 AWS Step Functions

您可以用AWS Step Functions來協調 AWS Lambda 功能和其他 AWS 資源，以形成複雜且強大的工作流程。Step Functions 告訴您的應用程式何時及在何種條件下使用您的 AWS 資源 (例如 AWS Lambda

函數)。這簡化了形成複雜而強大的工作流程的過程。使用時AWS::Serverless::StateMachine，您可以定義工作流程中的個別步驟、關聯每個步驟中的資源，然後將這些步驟排列在一起。您還可以在需要的地方加入轉變和條件。這簡化了製作複雜而強大的工作流程的過程。

> **Note**
>
> 若要管理包含 Step Functions 狀態機器的 AWS SAM 範本，您必須使用. AWS SAMCLI 若要檢查您擁有的版本，請執行指令`sam --version`。

Step Functions 是基於任務和狀態機的概念。您可以使用 JSON 為基礎的 Amazon 州語言定義狀態機器。Step Functions 主控台會顯示狀態機器結構的圖形檢視，讓您可以直觀地檢查狀態機器的邏輯並監視執行。

使用 AWS Serverless Application Model （ AWS SAM ）中的 Step Functions 支持，您可以執行以下操作：

- 直接在 AWS SAM 範本中或在單獨的檔案中定義狀態機
- 透過 AWS SAM 原則範本、內嵌原則或受管理的原則建立狀態機器執行角色
- 使用 API Gateway 或 Amazon EventBridge 事件、 AWS SAM 範本中的排程或直接呼叫 API 來觸發狀態機器執行
- 針對常見 Step Functions 開發模式使用可用的AWS SAM 原則範本。

範例

下列範 AWS SAM 本檔案中的範例程式碼片段會在定義檔案中定義 Step Functions 狀態機器。請注意，`my_state_machine.asl.json`檔案必須以 Amazon 州語言撰寫。

```
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
      ...
```

若要下載包含 Step Functions 狀態機器的範例 AWS SAM 應用程式，請參閱AWS Step Functions 開發人員指南 AWS SAM中的建立 Step Functions 狀態機器使用。

其他資訊

若要深入瞭解 Step Functions 並將其搭配使用 AWS SAM，請參閱下列內容：

- AWS Step Functions 的運作方式
- AWS Step Functions 而且 AWS Serverless Application Model
- 教學課程：使用建立 Step Functions 狀態機器 AWS SAM
- AWS SAM 規格: AWS::Serverless::StateMachine

為您的應用程式設定 AWS SAM 程式碼簽章

若要確保只部署受信任的程式碼，您可以使用 AWS SAM 來啟用無伺服器應用程式的程式碼簽章。簽署您的程式碼有助於確保程式碼自簽署後未遭到變更，而且只有來自受信任發行者的已簽署程式碼套件才會在 Lambda 函數中執行 這有助於讓組織免於在其部署管線中建置閘道管理員元件的負擔。

如需程式碼簽章的詳細資訊，請參閱AWS Lambda 開發人員指南中的設定 Lambda 函數的程式碼簽章。

您必須先使用 AWS Signer 建立簽署設定檔，才能為無伺服器應用程式設定程式碼簽章。您可以使用此簽署設定檔執行下列工作：

1. 建立程式碼簽章配置 — 宣告`AWS::Lambda::CodeSigningConfig`資源以指定受信任的發行者的簽署設定檔，並設定驗證檢查的原則動作。您可以在與無伺服器函數相同的 AWS SAM 範本、不同的範本或 AWS SAM 範本中宣告此物件 AWS CloudFormation 。然後，透過使用資源的 Amazon 資源名稱 (ARN) 指定函數的`CodeSigningConfigArn`屬性，以啟用無伺服器函數的程式碼簽章。`AWS::Lambda::CodeSigningConfig`
2. 簽署程式碼 — 使用`sam package`或`sam deploy`指令搭配選`--signing-profiles`項。

> **Note**
>
> 若要使用`sam package`或`sam deploy`命令成功簽署程式碼，必須為您搭配這些命令使用的 Amazon S3 儲存貯體啟用版本控制。如果您使用的是為您 AWS SAM 建立的 Amazon S3 儲存貯體，則會自動啟用版本控制。如需有關 Amazon S3 儲存貯體版本控制的詳細資訊，以及在

您提供的 Amazon S3 儲存貯體上啟用版本控制的指示，請參閱 Amazon 簡單儲存服務使用者指南中的在 Amazon S3 儲存貯體中使用版本控制。

當您部署無伺服器應用程式時，Lambda 會對您啟用程式碼簽章的所有函數執行驗證檢查。Lambda 也會對這些函數所依賴的任何層執行驗證檢查。如需 Lambda 驗證檢查的詳細資訊，請參閱AWS Lambda 開發人員指南中的簽名驗證。

範例

建立簽署設定檔

若要建立簽署設定檔，請執行下列命令：

```
aws signer put-signing-profile --platform-id "AWSLambda-SHA384-ECDSA" --profile-
name MySigningProfile
```

如果上一個命令成功，您會看到傳回簽署設定檔的 ARN。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"
}
```

此`profileVersionArn`欄位包含建立程式碼簽章設定時要使用的 ARN。

建立程式碼簽章設定並啟用函式的程式碼簽章

下列範例 AWS SAM 範本宣告`AWS::Lambda::CodeSigningConfig`資源並啟用 Lambda 函數的程式碼簽章。在此範例中，有一個受信任的設定檔，如果簽章檢查失敗，則會拒絕部署。

```
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"
```

簽署您的程式碼

您可以在封裝或部署應用程式時簽署程式碼。使用sam package或指sam deploy令指定--signing-profiles選項，如下列範例指令所示。

封裝應用程式時簽署函數程式碼：

```
sam package --signing-profiles HelloWorld=MySigningProfile --s3-bucket test-bucket --
output-template-file packaged.yaml
```

在打包應用程序時，簽名函數代碼和函數所依賴的層：

```
sam package --signing-profiles HelloWorld=MySigningProfile MyLayer=MySigningProfile --
s3-bucket test-bucket --output-template-file packaged.yaml
```

簽署您的函數代碼和層，然後執行部署：

```
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**
>
> 若要使用sam package或sam deploy命令成功簽署程式碼，必須為您搭配這些命令使用的 Amazon S3 儲存貯體啟用版本控制。如果您使用的是為您 AWS SAM 建立的 Amazon S3 儲存貯體，則會自動啟用版本控制。如需有關 Amazon S3 儲存貯體版本控制的詳細資訊，以及在您提供的 Amazon S3 儲存貯體上啟用版本控制的指示，請參閱 Amazon 簡單儲存服務使用者指南中的在 Amazon S3 儲存貯體中使用版本控制。

提供簽署設定檔 `sam deploy --guided`

當您使用已設定程式碼簽章的無伺服器應用程式執行`sam deploy --guided`命令時， AWS SAM 會提示您提供用於程式碼簽章的簽署設定檔。若要取得有關`sam deploy --guided`提示的更多資訊，請參閱指 AWS SAMCLI令參考sam deploy中的。

驗證 AWS SAM 範本檔

使用驗證您的範本`sam validate`。目前，此命令會驗證所提供的範本是否有效的 JSON /YAML。與大多數 AWS SAMCLI指令一樣，依預設，它會在目前的工作目錄中尋找`template.[yaml|yml]`檔案。您可以使用`-t`或`--template`選項指定不同的範本檔案/位置。

範例：

```
$ sam validate
<path-to-template>/template.yaml is a valid SAM Template
```

> **Note**
>
> 此命`sam validate`令需要設定 AWS 認證。如需更多詳細資訊，請參閱 配置 AWS SAMCLI。

建置您的應用程式 AWS SAM

將基礎結構作為代碼（IaC）添加到 AWS SAM 模板後，您就可以開始使用該sam build命令構建應用程序。這個命令會從應用程式專案目錄中的檔案 (也就是 AWS SAM 範本檔案、應用程式程式碼，以及任何適用的語言特定檔案和相依性) 建立組建成品。這些建置成品會準備您的無伺服器應用程式，以便稍後進行應用程式開發的步驟，例如本機測試和部署到 AWS 雲端。測試和部署都使用構建工件作為輸入。

您可以用sam build來建置整個無伺服器應用程式。此外，您還可以創建自定義構建，例如具有特定功能，層或自定義運行時的構建。若要深入瞭解使用方式和原因sam build，請參閱本節中的主題。若要取得使用`sam build`指令的簡介，請參閱〈〉使用sam build指令建立簡介。

主題

- 使用sam build指令建立簡介
- 默認構建 AWS SAM
- 定制構建 AWS SAM

使用sam build指令建立簡介

使用命 AWS Serverless Application Model 令列介面 (AWS SAMCLI) `sam build` 命令為開發工作流程中的後續步驟準備無伺服器應用程式，例如本機測試或部署到 AWS 雲端. 此命令會建立一個`.aws-sam`目錄，以及`sam deploy`需要的格式和位置來建構應用程式。`sam local`

- 如需「」的簡介 AWS SAMCLI，請參閱什麼是 AWS SAMCLI？。
- 如需指`sam build`令選項的清單，請參閱sam build。
- 如需在典型開發工作流程`sam build`期間使用的範例，請參閱步驟 2：建置您的應用程式。

> **Note**
>
> 使用時，您`sam build`必須從開發電腦上的無伺服器應用程式的基本元件開始。這包括一個 AWS SAM 模板， AWS Lambda 函數代碼，以及任何特定語言的文件和依賴關係。如需進一步了解，請參閱使用sam init指令建立您的應用程式。

主題

使用 sam 構建構建應用程式

使用前`sam build`，請考慮設定下列項目：

1. Lambda 函數和層 — 該`sam build`命令可以建立 Lambda 函數和層。若要進一步了解 Lambda 層，請參閱建置 Lambda 層。
2. Lambda 執行階段 — 執行階段提供語言特定的環境，可在呼叫時在執行環境中執行函數。您可以設定原生和自訂執行階段。

a. 原生執行階段 — 在支援的 Lambda 執行階段中撰寫您的 Lambda 函數，並建立函數，以便在 AWS 雲端.

b. 自訂執行階段 — 使用任何程式設計語言撰寫 Lambda 函數，並使用在makefile或第三方建置器 (例如) 中定義的自訂程序來建立執行階段esbuild。如需進一步了解，請參閱使用自訂執行階段建置 Lambda 函數。

3. Lambda 套件類型 — Lambda 函數可以封裝在下列 Lambda 部署套件類型中：

a. .zip 檔案封存 — 包含您的應用程式程式碼及其相依性。

b. 容器映像 — 包含基本作業系統、執行階段、Lambda 擴充功能、您的應用程式程式碼及其相依性。

您可以在使用初始化應用程式時設定這些應用`sam init`程式設定。

- 若要進一步瞭解如何使用`sam init`，請參閱使用sam init指令建立您的應用程式。
- 若要進一步瞭解如何在應用程式中設定這些設定，請參閱默認構建 AWS SAM。

若要建置應用程式

1. `cd`到你的項目的根目錄。這與您的 AWS SAM 範本位置相同。

```
$ cd sam-app
```

2. 執行下列命令：

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

> **Note**
>
> 常用的選項是`--use-container`。如需進一步了解，請參閱在提供的容器內部構建 Lambda 函數。

以下是 AWS SAMCLI輸出的範例：

```
sam-app $ sam build
Starting Build use cache
```

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

Build Succeeded

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

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

3. 會建 AWS SAMCLI立組`.aws-sam`建目錄。以下是範例：

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

視應用程式的設定方式而定， AWS SAMCLI會執行下列動作：

1. 下載，安裝和組織`.aws-sam/build`目錄中的依賴關係。
2. 準備您的 Lambda 式碼。這可能包括編譯程式碼、建立可執行的二進位檔案，以及建置容器映像檔。
3. 將組建加工品複製到`.aws-sam`目錄。格式會根據您的應用程式套件類型而有所不同。
   a. 對於 .zip 封裝類型，成品尚未壓縮，因此可用於本機測試。使用時 AWS SAMCLI拉鍊您的應用`sam deploy`程序。

b. 對於容器映像封裝類型，會在本機建立容器映像，並在.aws-sam/build.toml檔案中參考。

4. 將 AWS SAM 範本複製到.aws-sam目錄中，並在必要時使用新的檔案路徑對其進行修改。

以下是組成.aws-sam目錄中組成構成品的主要元件：

- 建置目錄 — 包含彼此獨立結構的 Lambda 函數和層。這會為.aws-sam/build目錄中的每個函數或圖層產生唯一的結構。
- AWS SAM 範本-根據建置流程期間的變更以更新的值進行修改。
- 建置的 .toml 檔案 — 組態檔案，其中包含使用的組建設定。 AWS SAMCLI

本機測試與部署

使用執行本機測試sam local或使用部署時sam deploy， AWS SAMCLI會執行下列動作：

1. 它會先檢查.aws-sam目錄是否存在，以及 AWS SAM 範本是否位於該目錄內。如果符合這些條件，會 AWS SAMCLI將其視為應用程式的根目錄。
2. 如果不符合這些條件，會 AWS SAMCLI將 AWS SAM 範本的原始位置視為應用程式的根目錄。

開發時，如果對原始應用程式檔案進行了變更，請在本機測試之前執行sam build以更新.aws-sam目錄。

最佳實務

- 不要編輯.aws-sam/build目錄下的任何代碼。相反，請更新項目文件夾中的原始源代碼，然後運行sam build以更新目.aws-sam/build錄。
- 當您修改原始檔案時，sam build請執行以更新目.aws-sam/build錄。
- 您可能希 AWS SAMCLI望引用項目的原始根目錄而不是.aws-sam目錄，例如在使用sam local. 刪除目.aws-sam錄中的目錄或 AWS SAM 模板，以使您的原始項目目錄 AWS SAMCLI識別為根項目目錄。.aws-sam準備就緒後，sam build再次執行以建立目.aws-sam錄。
- 當您執行時sam build，每次都會覆寫.aws-sam/build目錄。目.aws-sam錄沒有。如果您想要儲存檔案 (例如記錄)，請將它們儲存在中，.aws-sam以防止它們遭到覆寫。

山姆構建的選項

建立單一資源

提供資源的邏輯 ID 以僅建置該資源。以下是範例：

```
$ sam build HelloWorldFunction
```

若要建立巢狀應用程式或堆疊的資源，請使用下列格式提供應用程式或堆疊邏輯 ID 以及資源邏輯 ID*<stack-logical-id>*/*<resource-logical-id>*：

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

在提供的容器內部構建 Lambda 函數

`--use-container`此選項會下載容器映像檔，並使用它來建立您的 Lambda 函數。然後在您的`.aws-sam/build.toml`文件中引用本地容器。

需要Docker安裝此選項。如需說明，請參閱安裝 Docker。

以下是此命令的範例：

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

您可以指定要與`--build-image`選項搭配使用的容器映像檔。以下是範例：

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

若要指定用於單一函數的容器映像檔，請提供函數邏輯 ID。以下是範例：

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

將環境變量傳遞給構建容器

使用`--container-env-var`將環境變數傳遞至組建容器。以下是範例：

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

若要從檔案傳遞環境變數，請使用--container-env-var-file選項。以下是範例：

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

env.json文件示例：

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

加速構建包含多種功能的應用程序

當您在具有多個功能的應用程序sam build上運行時，每次 AWS SAMCLI構建一個函數。若要加速建置程序，請使用--parallel選項。這會同時構建所有功能和層。

以下是此命令的範例：

```
$ sam build —-parallel
```

通過在源文件夾中構建項目來加快構建時間

對於支持的運行時和構建方法，您可以使用該--build-in-source選項直接在源文件夾中構建項目。默認情況下， AWS SAM CLI構建在臨時目錄中，其中涉及複製源代碼和項目文件。使用--build-in-source，直接在源文件夾中 AWS SAM CLI構建，通過消除將文件複製到臨時目錄的需要來加快構建過程。

如需支援的執行階段和建置方法的清單，請參閱--build-in-source。

故障診斷

若要疑難排解 AWS SAMCLI，請參閱AWS SAMCLI疑難排。

範例

建置使用原生執行階段和 .zip 封裝類型的應用程式

如需此範例，請參閱教學課程：部署 Hello World 應用程式。

建置使用原生執行階段和影像封裝類型的應用程式

首先，我們運行`sam init`初始化一個新的應用程序。在互動流程中，我們選取`Image`套件類型。以下是範例：

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

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

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

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

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

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

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

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

Project name [sam-app]: ENTER

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

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

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

...
```

AWS SAMCLI初始化應用程式並建立下列專案目錄：

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

```
### template.yaml
```

接下來，我們運行構sam build建我們的應用程序：

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

Build Succeeded

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

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

建置包含已編譯程式設計語言的應用程式

在此範例中，我們使用Go執行階段建立包含 Lambda 函數的應用程式。

首先，我們初始化一個新的應用程序使用sam init和配置我們的應用程序使用Go：

```
$ sam init
```

```
...

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

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

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

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

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

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

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

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

Project name [sam-app]: ENTER

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

```
    -----------------------
    Generating application:
    -----------------------
    Name: sam-app
    Runtime: go1.x
    Architectures: x86_64
    Dependency Manager: mod
    Application Template: hello-world
    Output Directory: .
    Configuration file: sam-app/samconfig.toml

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

...
```

AWS SAMCLI初始化應用程式。以下是應用程式目錄結構的範例：

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

我們會針對此應用程式的需求參考該README.md檔案。

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

接下來，我們運行測sam local invoke試我們的功能。這個命令錯誤，因Go為沒有安裝在我們的本地機器上：

```
sam-app $ sam local invoke
Invoking hello-world (go1.x)
Local image was not found.
Removing rapid images for repo public.ecr.aws/sam/emulation-go1.x
Building
 image.....................................................................................
Using local image: public.ecr.aws/lambda/go:1-rapid-x86_64.

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

接下來，我們執sam build行建置應用程式。我們遇到一個錯誤，因Go為沒有安裝在我們的本地機器上：

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

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

雖然我們可以配置我們的本地計算機來正確構建我們的功能，但我們改--use-container用sam build. 下 AWS SAMCLI載一個容器映像，使用本機構建我們的函數 GoModulesBuilder，並將生成的二進製文件複製到我們的.aws-sam/build/HelloWorldFunction目錄中。

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

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

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

Build Succeeded

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

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

以下是目.aws-sam錄的範例：

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

接下來，我們跑sam local invoke。我們的函數被成功調用：

```
sam-app $ sam local invoke
Invoking hello-world (go1.x)
Local image is up-to-date
```

```
Using local image: public.ecr.aws/lambda/go:1-rapid-x86_64.

Mounting /Users/.../Playground/build/sam-app/.aws-sam/build/HelloWorldFunction as /var/
task:ro,delegated inside runtime container
START RequestId: cfc8ffa8-29f2-49d4-b461-45e8c7c80479 Version: $LATEST
END RequestId: cfc8ffa8-29f2-49d4-b461-45e8c7c80479
REPORT RequestId: cfc8ffa8-29f2-49d4-b461-45e8c7c80479  Init Duration: 1.20 ms
 Duration: 1782.46 ms        Billed Duration: 1783 ms        Memory Size: 128 MB
 Max Memory Used: 128 MB
{"statusCode":200,"headers":null,"multiValueHeaders":null,"body":"Hello,
 72.21.198.67\n"}%
```

進一步了解

若要進一步瞭解如何使用`sam build`指令，請參閱下列內容：

- 學習 AWS SAM：sam 構建 — 無服務器土地「學習 AWS SAM」系列上YouTube。
- 學習 AWS SAM | 山姆構建 | E3 — 無服務器土地「學習 AWS SAM」系列。YouTube
- AWS SAM 構建：它如何為部署提供工件（使用 SAM S2E8 的會話）-開啟序列的 AWS SAM 會話。YouTube
- AWS SAM 自定義構建：如何使用生成文件在 SAM（S2E9）中自定義構建-開啟系列的會話。 AWS SAM YouTube

默認構建 AWS SAM

若要建置無伺服器應用程式，請使用指`sam build`令。此命令也會收集應用程式相依性的建置成品，並將它們置於適當的格式和位置，以便進行後續步驟，例如本機測試、封裝和部署。

您可以在資訊清單檔案中指定應用程式的相依性，例如 `requirements.txt` `package.json` (Python) 或 (Node.js)，或使用函數資源的`Layers`屬性。此`Layers`屬性包含 Lambda 函數所依賴的AWS Lambda 圖層資源清單。

應用程序的構建成品的格式取決於每個函數的`PackageType`屬性。此屬性的選項包括：

- **`Zip`**— .zip 檔案封存，其中包含您的應用程式程式碼及其相依性。如果您將程式碼封裝為 .zip 檔案封存，則必須為函數指定 Lambda 執行階段。
- **`Image`**— 容器映像檔，其中包括基本作業系統、執行階段和擴充功能，以及您的應用程式程式碼及其相依性。

如需 Lambda 套件類型的詳細資訊，請參閱AWS Lambda 開發人員指南中的 Lambda 部署套件。

主題

- 建立 .zip 檔案封存
- 建立容器映像檔
- 容器環境變量文件
- 通過在源文件夾中構建項目來加快構建時間
- 範例
- 建立以外的功能 AWS SAM

建立 .zip 檔案封存

若要將無伺服器應用程式建置為 .zip 檔案封存，請`PackageType: Zip`針對無伺服器函數宣告。

AWS SAM 針對您指定的架構建置應用程式。如果您未指定架構，則`x86_64`依預設 AWS SAM 會使用。

如果您的 Lambda 函數依賴於具有原生編譯程式的套件，請使用`--use-container`旗標。此旗標會在本機編譯 Docker 容器中的函數，該容器的行為類似於 Lambda 環境，因此當您將函數部署到雲端時，它們會採用正確的格式。 AWS

當您使用`--use-container`此選項時，依預設會從 Amazon ECR 公開 AWS SAM提取容器映像檔。例如，如果您想要從另一個儲存庫提取容器映像檔 DockerHub，您可以使用此`--build-image`選項並提供替代容器映像檔的 URI。以下是使用 DockerHub 儲存庫中的容器映像建置應用程式的兩個範例命令：

```
# Build a Node.js 20 application using a container image pulled from DockerHub
sam build --use-container --build-image amazon/aws-sam-cli-build-image-nodejs20.x

# Build a function resource using the Python 3.12 container image pulled from DockerHub
sam build --use-container --build-image Function1=amazon/aws-sam-cli-build-image-
python3.12
```

如需可搭配使用的 URI 清單`--build-image`，請參閱映像儲存庫其中包含許多受支援執行階段的 DockerHub URI。

如需建立 .zip 檔案封存應用程式的其他範例，請參閱本主題稍後的 < 範例 > 一節。

建立容器映像檔

若要將無伺服器應用程式建置為容器映像檔，請PackageType: Image針對無伺服器函數宣告。您也必須使用下列項目宣告Metadata資源屬性：

`Dockerfile`

與 Lambda 函數相關聯的碼頭檔案的名稱。

`DockerContext`

碼頭文件的位置。

`DockerTag`

(選擇性) 要套用至建置影像的標記。

`DockerBuildArgs`

為構建構建構建立參數。

以下是Metadata資源屬性區段的範例：

```
  Metadata:
    Dockerfile: Dockerfile
    DockerContext: ./hello_world
    DockerTag: v1
```

若要下載使用Image套件類型設定的範例應用程式，請參閱教學課程：部署 Hello World 應用程式教學課程：部署 Hello World 應用程式中的。在詢問您要安裝哪種套件類型的提示下，選擇Image。

> **Note**
>
> 如果您在 Dockerfile 中指定了多架構基礎映像檔，請為主機的架構 AWS SAM 建立容器映像檔。若要針對不同的架構進行建置，請指定使用特定目標架構的基礎映像檔。

容器環境變量文件

若要提供包含組建容器環境變數的 JSON 檔案，請搭配sam build命令使用--container-env-var-file引數。您可以提供套用至所有無伺服器資源的單一環境變數，或為每個資源提供不同的環境變數。

格式

將環境變數傳遞至組建容器的格式取決於您為資源提供的環境變數數目。

若要為所有資源提供單一環境變數，請指定如下所示的Parameters物件：

```
{
  "Parameters": {
    "GITHUB_TOKEN": "TOKEN_GLOBAL"
  }
}
```

若要為每個資源提供不同的環境變數，請為每個資源指定物件，如下所示：

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

將您的環境變數儲存為檔案，例如命名env.json。以下命令使用此文件將環境變量傳遞給構建容器：

```
sam build --use-container --container-env-var-file env.json
```

優先順序

- 您為特定資源提供的環境變數優先於所有資源的單一環境變數。
- 您在命令列上提供的環境變數優先於檔案中的環境變數。

通過在源文件夾中構建項目來加快構建時間

對於支持的運行時和構建方法，您可以使用該--build-in-source選項直接在源文件夾中構建項目。默認情況下， AWS SAM CLI構建在臨時目錄中，其中涉及複製源代碼和項目文件。使用--build-in-source，直接在源文件夾中 AWS SAM CLI構建，通過消除將文件複製到臨時目錄的需要來加快構建過程。

如需支援的執行階段和建置方法的清單，請參閱--build-in-source。

範例

範例 1：.zip 檔案封存

下列sam build指令會建立 .zip 檔案封存：

```
# Build all functions and layers, and their dependencies
sam build

# Run the build process inside a Docker container that functions like a Lambda
 environment
sam build --use-container

# Build a Node.js 20 application using a container image pulled from DockerHub
sam build --use-container --build-image amazon/aws-sam-cli-build-image-nodejs20.x

# Build a function resource using the Python 3.12 container image pulled from DockerHub
sam build --use-container --build-image Function1=amazon/aws-sam-cli-build-image-
python3.12

# Build and run your functions locally
sam build && sam local invoke

# For more options
sam build --help
```

範例 2：容器影像

下列 AWS SAM 範本會建置為容器映像檔：

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      PackageType: Image
      ImageConfig:
        Command: ["app.lambda_handler"]
    Metadata:
      Dockerfile: Dockerfile
      DockerContext: ./hello_world
      DockerTag: v1
```

下面是一個例子碼頭文件：

```
FROM public.ecr.aws/lambda/python:3.12

COPY app.py requirements.txt ./

RUN python3.12 -m pip install -r requirements.txt

# Overwrite the command by providing a different command directly in the template.
CMD ["app.lambda_handler"]
```

範例 3：故宮 CI

對於 Node.js 應用程序，您可以使用npm ci而不是npm install安裝依賴項。若要使用npm ci，請UseNpmCi: TrueBuildProperties在 Lambda 函數的Metadata資源屬性中指定下方。若要使用npm ci，您的應用程式必須具有 Lambda 函數CodeUri的package-lock.json或npm-shrinkwrap.json檔案。

下列範例會npm ci在您執行時安裝相依性sam build：

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: hello-world/
      Handler: app.handler
      Runtime: nodejs20.x
      Architectures:
        - x86_64
      Events:
        HelloWorld:
          Type: Api
          Properties:
            Path: /hello
            Method: get
    Metadata:
      BuildProperties:
        UseNpmCi: True
```

建立以外的功能 AWS SAM

依預設，當您執行時sam build，會 AWS SAM 建置所有函式資源。其他選項包括：

- 在以外構建所有函數資源 AWS SAM-如果您手動或通過其他工具構建所有函數資源，sam build則不需要。您可以跳過sam build並繼續進行程序中的下一個步驟，例如執行本機測試或部署應用程式。
- 在以外建置一些函式資源 AWS SAM — 如果您想 AWS SAM 要建置一些函式資源，同時在其他函式資源之外建置 AWS SAM，您可以在 AWS SAM 範本中指定此資源。

在以外構建一些函數資源 AWS SAM

要在使用時 AWS SAM 跳過某個功能sam build，請在 AWS SAM 模板中配置以下內容：

1. 將SkipBuild: True中繼資料屬性新增至您的函數。
2. 指定建置函式資源的路徑。

這是一個示例，TestFunction配置為被跳過。它的內置資源位於built-resources/TestFunction.zip.

```
TestFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: built-resources/TestFunction.zip
    Handler: TimeHandler::handleRequest
    Runtime: java11
  Metadata:
    SkipBuild: True
```

現在，當您運行時sam build， AWS SAM 將執行以下操作：

1. AWS SAM 將跳過配置的功能SkipBuild: True。
2. AWS SAM 將構建所有其他函數資源並將其緩存在.aws-sam構建目錄中。
3. 對於跳過的函數，它們在 .aws-sam build 目錄中的模板將自動更新，以引用指定的路徑到您的構建函數資源。

   以下是.aws-sam構建目錄TestFunction中緩存模板的示例：

   ```
   TestFunction:
     Type: AWS::Serverless::Function
     Properties:
       CodeUri: ../../built-resources/TestFunction.zip
       Handler: TimeHandler::handleRequest
       Runtime: java11
   ```

```
  Metadata:
    SkipBuild: True
```

定制構建 AWS SAM

您可以自訂組建以包含特定的 Lambda 函數或 Lambda 層。函數是您可以叫用以在 Lambda 中執行程式碼的資源。Lambda 層可讓您從 Lambda 函數擷取程式碼，然後可在多個 Lambda 函數中重複使用。如果您想要專注於開發和部署個別無伺服器功能，而不需要管理共用相依性或資源的複雜性，您可以選擇使用特定的 Lambda 函數自訂組建。此外，您可以選擇建立 Lambda 層來協助您減少部署套件的大小、將核心函數邏輯與相依性區隔開來，以及讓您在多個函數之間共用相依性。

本節中的主題將探索建置 Lambda 函數的一些不同方式 AWS SAM。這包括使用客戶執行階段建置 Lambda 函數，以及建置 Lambda 層。自訂執行階段可讓您安裝和使用 AWS Lambda 開發人員指南中未列在 Lambda 執行階段中的語言。這可讓您建立執行無伺服器函數和應用程式的專用執行環境。只建置 Lambda 層 (而不是建置整個應用程式) 可以透過幾種方式為您帶來好處。它可以幫助您減少部署軟件包的大小，將核心函數邏輯與依賴項分開，並允許您在多個函數之間共享依賴關係。

如需有關函數的詳細資訊，請參閱AWS Lambda 開發人員指南中的 Lambda 概念。

主題

- 使用電子建置建置 Node.js Lambda 函數
- 使用原生 AOT 編譯來建置 .NET Lambda 函數
- 使用建置 Lambda 鏽函數 Cargo Lambda
- 使用自訂執行階段建置 Lambda 函數
- 建置 Lambda 層

使用電子建置建置 Node.js Lambda 函數

要構建和打包 Node.js AWS Lambda 函數，您可以使用 AWS SAMCLI與 esbuild JavaScript 捆綁程序。電子構建捆綁程序支持您編寫的 Lambda 函數。 TypeScript

若要使用電子建置來建置 Node.js Lambda 函數，請將`Metadata`物件新增至您的`AWS:Serverless::Function`資源，並`esbuild`為. `BuildMethod` 當您執行sam build命令時，AWS SAM 會使用電子建置來捆綁您的 Lambda 函數程式碼。

元數據屬性

該`Metadata`對象支持以下 ebuild 屬性。

BuildMethod

指定應用程式的捆綁器。唯一支援的值為 `esbuild`。

BuildProperties

指定 Lambda 函數程式碼的組建屬性。

該`BuildProperties`對象支持以下 ebuild 屬性。所有屬性都是可選的。依預設， AWS SAM 會使用 Lambda 函數處理常式做為入口點。

EntryPoints

指定應用程式的進入點。

外部

指定要從組建中省略的套件清單。如需詳細資訊，請參閱esbuild網站中的「外部」。

格式

指定應用程式中產生 JavaScript 檔案的輸出格式。如需詳細資訊，請參閱 esbuild 網站中的格式。

載入器

指定載入給定檔案類型之資料的組態清單。

MainFields

指定解析封裝時要嘗試匯入的`package.json`欄位。預設值為 `main,module`。

縮小

指定是否要縮小隨附的輸出程式碼。預設值為 `true`。

OutExtension

自訂 esbuild 所產生之檔案的副檔名。如需詳細資訊，請參閱 esbuild 網站中的 Out 擴充功能。

源映射

指定捆綁器是否產生源映射文件。預設值為 `false`。

設定為時`true`，`NODE_OPTIONS: --enable-source-maps`會附加至 Lambda 函數的環境變數，並產生來源對應並包含在函數中。

或者，當包含`NODE_OPTIONS: --enable-source-maps`在函數的環境變數中時，`Sourcemap`會自動設定為`true`。

發生衝突時，優Sourcemap: false先順序高於NODE_OPTIONS: --enable-source-maps。

> **Note**
>
> 根據預設，Lambda 會使用 AWS Key Management Service (AWS KMS) 加密所有靜態環境變數。使用來源對應時，若要成功部署，函數的執行角色必須具有執行kms:Encrypt動作的權限。

SourcesContent

指定是否在來源對應檔案中包含原始程式碼。設定為時，請Sourcemap設定此內容'true'。

- 指SourcesContent: 'true'定包含所有原始程式碼。
- 指定排SourcesContent: 'false'除所有原始程式碼。這會導致較小的源映射文件大小，這通過減少啟動時間在生產中非常有用。但是，調試器中將無法使用源代碼。

預設值為 SourcesContent: true。

如需詳細資訊，請參閱 esbuild 網站中的來源內容。

目標

指定目標電子印刷稿版本。預設值為 es2020。

TypeScript Lambda 函數示例

下列範例 AWS SAM 範本程式碼片段使用電子建置，從中hello-world/app.ts的程式 TypeScript 碼建立 Node.js Lambda 函數。

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: hello-world/
      Handler: app.handler
      Runtime: nodejs20.x
      Architectures:
        - x86_64
      Events:
        HelloWorld:
```

```
        Type: Api
        Properties:
          Path: /hello
          Method: get
    Environment:
      Variables:
        NODE_OPTIONS: --enable-source-maps
  Metadata:
    BuildMethod: esbuild
    BuildProperties:
      Format: esm
      Minify: false
      OutExtension:
        - .js=.mjs
      Target: "es2020"
      Sourcemap: true
      EntryPoints:
        - app.ts
      External:
        - "<package-to-exclude>"
```

使用原生 AOT 編譯來建置 .NET Lambda 函數

使用 AWS Serverless Application Model (AWS SAM) 構建和打包您的 .NET 8 AWS Lambda 函數，利用本機提前時間（AOT）編譯來改善冷啟動時間。 AWS Lambda

主題

- .NET 8 原生 AOT 概述
- 搭 AWS SAM 配 .NET 8 Lambda 函數使用
- 安裝先決條
- 在 AWS SAM 範本中定義 .NET 8 個 Lambda 函數
- 建置您的應用程式 AWS SAMCLI
- 進一步了解

.NET 8 原生 AOT 概述

從過去看，.NET Lambda 函數具有冷啟動時間，這會影響無伺服器應用程式的使用者體驗、系統延遲和使用成本。使用 .NET 原生 AOT 編譯，您可以改善 Lambda 函數的冷啟動時間。若要深入瞭解 .NET 8 的原生 AOT，請參閱在 Dot GitHub net 儲存庫中使用原生 AOT。

搭 AWS SAM 配 .NET 8 Lambda 函數使用

請執行下列動作，以使用 AWS Serverless Application Model (AWS SAM) 設定 .NET 8 Lambda 函數：

- 在您的開發電腦上安裝必要條件。
- 在 AWS SAM 範本中定義 .NET 8 Lambda 函數。
- 使用建置您的應用程式 AWS SAMCLI。

安裝先決條

以下是必要的先決條件：

- 該 AWS SAMCLI
- 網路核 CLI
- 亞馬遜. Lambda. 工具. NET 核心全球工具
- Docker

安裝 AWS SAMCLI

1. 若要檢查您是否已 AWS SAMCLI安裝，請執行下列命令：

```
sam --version
```

2. 若要安裝 AWS SAMCLI，請參閱安裝 AWS SAMCLI。
3. 若要升級已安裝的版本 AWS SAMCLI，請參閱升級 AWS SAMCLI。

安裝. NET 核心 CLI

1. 若要下載並安裝 .NET 核心 CLI，請參閱從微軟的網站下載 .NET。
2. 如需有關 .NET 核心 CLI 的詳細資訊，請參閱AWS Lambda 開發人員指南中的 .NET 核心 CLI。

安裝亞馬遜. Lambda. 工具. NET 核心全球工具

1. 執行以下命令：

```
dotnet tool install -g Amazon.Lambda.Tools
```

2. 如果您已安裝此工具，您可以使用以下命令，確保使用的是最新版本。

```
dotnet tool update -g Amazon.Lambda.Tools
```

3. 如需有關亞馬遜網路核心全域工具的詳細資訊，請參閱上的 .NET CLI 存放庫的AWS 擴充功能。GitHub

安裝 Docker

- 使用本機 AOT 構建，需Docker要安裝。如需安裝指示，請參閱安裝泊塢視窗以搭配使用 AWS SAMCLI。

在 AWS SAM 範本中定義 .NET 8 個 Lambda 函數

若要在 AWS SAM 範本中定義 .NET8 Lambda 函數，請執行下列動作：

1. 從您選擇的起始目錄執行下列命令：

```
sam init
```

2. 選取AWS Quick Start Templates此選項可選擇起始範本。
3. 選擇Hello World Example範本。
4. 輸入以選擇不使用最常用的執行階段和套件類型n。
5. 對於執行階段，請選擇dotnet8。
6. 針對封裝類型，請選擇Zip。
7. 對於您的初學者範本，請選擇Hello World Example using native AOT。

安裝 Docker

- 使用本機 AOT 構建，需Docker要安裝。如需安裝指示，請參閱安裝泊塢視窗以搭配使用 AWS SAMCLI。

```
Resources:
HelloWorldFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: ./src/HelloWorldAot/
```

```
    Handler: bootstrap
    Runtime: dotnet8
    Architectures:
      - x86_64
    Events:
      HelloWorldAot:
        Type: Api
        Properties:
          Path: /hello
          Method: get
```

建置您的應用程式 AWS SAMCLI

從項目的根目錄中運行`sam build`以開始構建應用程序。如果`PublishAot`屬性已在 .NET 8 項目文件中定義，則 AWS SAMCLI將使用本機 AOT 編譯構建。若要深入了解`PublishAot`屬性，請參閱 Microsoft 的 .NET 文件中的原生 AOT 部署。

為了構建你的函數， AWS SAMCLI調用 .NET 核心 CLI 它使用亞馬遜. Lambda. 工具. NET 核心全局工具。

> **Note**
>
> 構建時，如果`.sln`文件存在於項目的同一目錄或父目錄中，則包含該`.sln`文件的目錄將被掛載到容器中。如果找不到`.sln`檔案，則僅會裝載專案資料夾。因此，如果您要建置多專案應用程式，請確定`.sln`檔案所在的屬性。

進一步了解

如需有關建置 .NET 8 Lambda 函數的詳細資訊，請參閱 AWS Lambda.

如需指sam build令的參考，請參閱sam build。

使用建置 Lambda 鏽函數 Cargo Lambda

此功能正在的預覽版中， AWS SAM 且可能會變更。

使用指 AWS Serverless Application Model 令行介面 (AWS SAMCLI) 搭配您的 Rust AWS Lambda 函數。

主題

必要條件

Rust語言

若要安裝Rust，請參閱Rust在Rust語言網站中安裝。

Cargo Lambda

AWS SAMCLI需要安裝 Cargo Lambda，的子指令。Cargo如需安裝指示，請參閱Cargo Lambda文件中的安裝。

Docker

需要建置和測試 Rust Lambda 函數Docker。如需安裝指示，請參閱安裝 Docker。

選擇加入 AWS SAMCLI測試版功能

由於此功能處於預覽狀態，因此您必須使用下列其中一種方法來選擇加入：

1. 使用環境變數：`SAM_CLI_BETA_RUST_CARGO_LAMBDA=1`。
2. 將以下內容新增到您的 `samconfig.toml` 檔案：

```
[default.build.parameters]
beta_features = true
[default.sync.parameters]
beta_features = true
```

3. 使用支援的 AWS SAMCLI指令時，請使用`--beta-features`此選項。例如：

```
$ sam build --beta-features
```

4. y當 AWS SAMCLI提示您選擇加入時選擇選項。以下是範例：

```
$ sam build
Starting Build use cache
Build method "rust-cargolambda" is a beta feature.
Please confirm if you would like to proceed
You can also enable this beta feature with "sam build --beta-features". [y/N]: y
```

設定 AWS SAM 為與 Rust 函數搭 Lambda 使用

步驟 1：設定 AWS SAM 範本

使用以下內容配置您的 AWS SAM 模板：

- 二進位 — 選用。指定您的範本何時包含多個 Rust Lambda 函數。
- BuildMethod – `rust-cargolambda`.
- CodeUri— `Cargo.toml` 檔案的路徑。
- 處理程序 —`bootstrap`.
- 運行時-`provided.al2`。

若要深入瞭解自訂執行階段，請參閱AWS Lambda 開發人員指南中的自 AWS Lambda 訂執行階段。

以下是已設定範 AWS SAM 本的範例：

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Metadata:
      BuildMethod: rust-cargolambda
      BuildProperties: function_a
    Properties:
      CodeUri: ./rust_app
      Handler: bootstrap
      Runtime: provided.al2
...
```

步驟 2：搭 AWS SAMCLI配您的 Rust Lambda 函數使用

對 AWS SAM 範本使用任何 AWS SAMCLI指令。如需詳細資訊，請參閱 該 AWS SAMCLI。

範例

你好世界的例子

在這個例子中，我們使用Rust作為我們的運行時構建示例 Hello World 應用程序。

首先，我們初始化一個新的無伺服器應用程式使用`sam init`. 在互動式流程中，我們選取 Hello World 應用程式並選擇 Rust 執行階段。

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

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

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

Which runtime would you like to use?
        1 - aot.dotnet7 (provided.al2)
        2 - dotnet6
        3 - dotnet5.0
        ...
        18 - python3.7
        19 - python3.10
        20 - ruby2.7
        21 - rust (provided.al2)
Runtime: 21

Based on your selections, the only Package type available is Zip.
We will proceed to selecting the Package type as Zip.

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

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

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

```
Project name [sam-app]: hello-rust

    -----------------------
    Generating application:
    -----------------------
    Name: hello-rust
    Runtime: rust (provided.al2)
    Architectures: x86_64
    Dependency Manager: cargo
    Application Template: hello-world
    Output Directory: .
    Configuration file: hello-rust/samconfig.toml

    Next steps can be found in the README file at hello-rust/README.md


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

以下是我們的 Hello World 應用程序的結構：

```
hello-rust
### README.md
### events
#   ### event.json
### rust_app
#   ### Cargo.toml
#   ### src
#       ### main.rs
### samconfig.toml
### template.yaml
```

在我們的 AWS SAM 模板中，我們的Rust函數定義如下：

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  HelloWorldFunction:
```

```
    Type: AWS::Serverless::Function
    Metadata:
      BuildMethod: rust-cargolambda
    Properties:
      CodeUri: ./rust_app
      Handler: bootstrap
      Runtime: provided.al2
      Architectures:
        - x86_64
      Events:
        HelloWorld:
          Type: Api
            Path: /hello
            Method: get
```

接下來，我們執sam build行建置應用程式並準備部署。 AWS SAMCLI創建一個.aws-sam目錄並在那裡組織我們的構建工件。我們的函數是使用構建Cargo Lambda並存儲為可執行二進製文件在.aws-sam/build/HelloWorldFunction/bootstrap.

> **Note**
>
> 如果您打算在 MacOS 中運行sam local invoke命令，則需要在調用之前構建不同的功能。若要這麼做，請使用下列命令：
>
> - SAM_BUILD_MODE=debug sam build
>
> 只有在本地測試將完成時才需要此命令。建置進行部署時，不建議這樣做。

```
hello-rust$ sam build
Starting Build use cache
Build method "rust-cargolambda" is a beta feature.
Please confirm if you would like to proceed
You can also enable this beta feature with "sam build --beta-features". [y/N]: y

Experimental features are enabled for this session.
Visit the docs page to learn more about the AWS Beta terms https://aws.amazon.com/
service-terms/.

Cache is invalid, running build and copying resources for following functions
 (HelloWorldFunction)
```

```
Building codeuri: /Users/.../hello-rust/rust_app runtime: provided.al2 metadata:
 {'BuildMethod': 'rust-cargolambda'} architecture: x86_64 functions: HelloWorldFunction
Running RustCargoLambdaBuilder:CargoLambdaBuild
Running RustCargoLambdaBuilder:RustCopyAndRename

Build Succeeded

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

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

接下來，我們部署應用程式使用`sam deploy --guided`.

```
hello-rust$ sam deploy --guided

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

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

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

```
        SAM configuration environment [default]: ENTER

        Looking for resources needed for deployment:

        ...

        Uploading to hello-rust/56ba6585d80577dd82a7eaaee5945c0b  817973 / 817973
 (100.00%)

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

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

        Uploading to hello-rust/a4fc54cb6ab75dd0129e4cdb564b5e89.template  1239 / 1239
 (100.00%)


Waiting for changeset to be created..

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

                            orldPermissionProd

...
-----------------------------------------------------------------------------------------------------
```

```
Changeset created successfully. arn:aws:cloudformation:us-
west-2:012345678910:changeSet/samcli-deploy1681427201/
f0ef1563-5ab6-4b07-9361-864ca3de6ad6


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

2023-04-13 13:07:17 - Waiting for stack create/update to complete

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

CREATE_IN_PROGRESS         AWS::IAM::Role             HelloWorldFunctionRole
 Resource creation
...
---------------------------------------------------------------------------------------------------

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

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

Description         Implicit IAM Role created for Hello World function

Value               arn:aws:iam::012345678910:role/hello-rust-
HelloWorldFunctionRole-10II2P13AUDUY

Key                 HelloWorldApi

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

Value               https://ggdxec9le9.execute-api.us-west-2.amazonaws.com/Prod/hello/


Key                 HelloWorldFunction
```

```
Description         Hello World Lambda Function ARN

Value               arn:aws:lambda:us-west-2:012345678910:function:hello-rust-
HelloWorldFunction-
yk4HzGzYeZBj

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


Successfully created/updated stack - hello-rust in us-west-2
```

若要進行測試，我們可以使用 API 端點叫用 Lambda 函數。

```
$ curl https://ggdxec91e9.execute-api.us-west-2.amazonaws.com/Prod/hello/
Hello World!%
```

為了在本地測試我們的功能，首先我們確保我們的功能Architectures屬性與我們的本地機器匹配。

```
...
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function # More info about Function Resource:
 https://github.com/awslabs/serverless-application-model/blob/master/
versions/2016-10-31.md#awsserverlessfunction
    Metadata:
      BuildMethod: rust-cargolambda # More info about Cargo Lambda: https://github.com/
cargo-lambda/cargo-lambda
    Properties:
      CodeUri: ./rust_app   # Points to dir of Cargo.toml
      Handler: bootstrap    # Do not change, as this is the default executable name
 produced by Cargo Lambda
      Runtime: provided.al2
      Architectures:
        - arm64
...
```

由於我們arm64在此示例中x86_64將架構從修改為，因此我們運行sam build以更新構建工件。然後我們運行sam local invoke到本地調用我們的函數。

```
hello-rust$ sam local invoke
Invoking bootstrap (provided.al2)
```

```
Local image was not found.
Removing rapid images for repo public.ecr.aws/sam/emulation-provided.al2
Building
 image.....................................................................................
Using local image: public.ecr.aws/lambda/provided:al2-rapid-arm64.

Mounting /Users/.../hello-rust/.aws-sam/build/HelloWorldFunction as /var/
task:ro,delegated, inside runtime container
START RequestId: fbc55e6e-0068-45f9-9f01-8e2276597fc6 Version: $LATEST
{"statusCode":200,"body":"Hello World!"}END RequestId:
 fbc55e6e-0068-45f9-9f01-8e2276597fc6
REPORT RequestId: fbc55e6e-0068-45f9-9f01-8e2276597fc6  Init Duration: 0.68 ms
 Duration: 130.63 ms     Billed Duration: 131 ms     Memory Size: 128 MB     Max Memory
 Used: 128 MB
```

單一 Lambda 函數專案

以下是一個包含一個 Rust Lambda 函數的無伺服器應用程式範例。

項目目錄結構：

```
.
### Cargo.lock
### Cargo.toml
### src
#   ### main.rs
### template.yaml
```

AWS SAM 模板：

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Metadata:
      BuildMethod: rust-cargolambda
    Properties:
      CodeUri: ./
      Handler: bootstrap
      Runtime: provided.al2
```

```
...
```

多個 Lambda 函數專案

以下是一個包含多個 Rust Lambda 函數的無伺服器應用程式範例。

項目目錄結構：

```
.
### Cargo.lock
### Cargo.toml
### src
#   ### function_a.rs
#   ### function_b.rs
### template.yaml
```

AWS SAM 模板：

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  FunctionA:
    Type: AWS::Serverless::Function
    Metadata:
      BuildMethod: rust-cargolambda
      BuildProperties:
        Binary: function_a
    Properties:
      CodeUri: ./
      Handler: bootstrap
      Runtime: provided.al2
  FunctionB:
    Type: AWS::Serverless::Function
    Metadata:
      BuildMethod: rust-cargolambda
      BuildProperties:
        Binary: function_b
    Properties:
      CodeUri: ./
      Handler: bootstrap
      Runtime: provided.al2
```

Cargo.toml 檔案：

```
[package]
name = "test-handler"
version = "0.1.0"
edition = "2021"

[dependencies]
lambda_runtime = "0.6.0"
serde = "1.0.136"
tokio = { version = "1", features = ["macros"] }
tracing = { version = "0.1", features = ["log"] }
tracing-subscriber = { version = "0.3", default-features = false, features = ["fmt"] }

[[bin]]
name = "function_a"
path = "src/function_a.rs"

[[bin]]
name = "function_b"
path = "src/function_b.rs"
```

使用自訂執行階段建置 Lambda 函數

您可以使用命sam build令建立 Lambda 函數所需的自訂執行階段。您可以透過指定Runtime: provided函數來宣告 Lambda 函數以使用自訂執行階段。

若要建立自訂執行階段，請使用BuildMethod: makefile項目宣告Metadata資源屬性。您提供了一個自定義 makefile，您可以在其中聲明包含運行時構建命令的表單build-*function-logical-id*的構建目標。您的 makefile 負責在必要時編譯自訂執行階段，並將組建成品複製到工作流程中後續步驟所需的適當位置。makefile 的位置由函數資源的CodeUri屬性指定，並且必須命名Makefile。

範例

示例 1：用 Rust 編寫的函數的自定義運行時

> **Note**
>
> 我們建議您使用Cargo Lambda. 如需進一步了解，請參閱使用建置 Lambda 鏽函數 Cargo Lambda。

以下 AWS SAM 模板聲明了一個函數，該函數使用 Rust 編寫的 Lambda 函數的自定義運行時，並指示`sam build`為`build-HelloRustFunction`構建目標執行命令。

```
Resources:
  HelloRustFunction:
    Type: AWS::Serverless::Function
    Properties:
      FunctionName: HelloRust
      Handler: bootstrap.is.real.handler
      Runtime: provided
      MemorySize: 512
      CodeUri: .
    Metadata:
      BuildMethod: makefile
```

下面的 makefile 包含構建目標和將被執行的命令。請注意，`CodeUri`屬性設定為`.`，因此 makefile 必須位於專案根目錄中 (亦即，與應用程式的 AWS SAM 範本檔案相同的目錄)。檔案名稱必須是`Makefile`。

```
build-HelloRustFunction:
  cargo build --release --target x86_64-unknown-linux-musl
  cp ./target/x86_64-unknown-linux-musl/release/bootstrap $(ARTIFACTS_DIR)
```

如需有關設定開發環境以執行前一個`cargo build`指令的詳細資訊`makefile`，請參閱 Rust Runtime 的 AWS Lambda部落格文章。

示例 2：Python3.12 的生成文件構建器（替代使用捆綁的構建器）

您可能需要使用未包含在捆綁構建器中的庫或模塊。此示例顯示了具有生成文件構建器的 Python3.12 運行時的 AWS SAM 模板。

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: hello_world/
      Handler: app.lambda_handler
      Runtime: python3.12
    Metadata:
      BuildMethod: makefile
```

下面的 makefile 包含構建目標和將被執行的命令。請注意，CodeUri屬性設定為hello_world，因此 makefile 必須位於hello_world子目錄的根目錄中，且檔案名稱必須是。Makefile

```
build-HelloWorldFunction:
    cp *.py $(ARTIFACTS_DIR)
    cp requirements.txt $(ARTIFACTS_DIR)
    python -m pip install -r requirements.txt -t $(ARTIFACTS_DIR)
    rm -rf $(ARTIFACTS_DIR)/bin
```

建置 Lambda 層

您可以用 AWS SAM 來建置自訂 Lambda 層。Lambda 層可讓您從 Lambda 函數擷取程式碼，然後可在多個 Lambda 函數中重複使用。只建置 Lambda 層 (而不是建置整個應用程式) 可以透過幾種方式為您帶來好處。它可以幫助您減少部署軟件包的大小，將核心函數邏輯與依賴項分開，並允許您在多個函數之間共享依賴關係。如需層的相關資訊，請參閱AWS Lambda 開發人員指南中的 AWS Lambda 層。

如何在中構建一個 Lambda 層 AWS SAM

> **Note**
>
> 您必須先在 AWS SAM 範本中撰寫 Lambda 層，才能建立 Lambda 層。如需執行此作業的資訊和範例，請參閱使用 Lambda 層來提高效率 AWS SAM。

要構建自定義層，請在 AWS Serverless Application Model （ AWS SAM ）模板文件中聲明它，並包含帶有BuildMethod條目的Metadata資源屬性部分。的有效值BuildMethod是AWS Lambda 執行階段的識別碼，或makefile。包含一個BuildArchitecture項目，以指定您的層支援的指令集架構。的有效值BuildArchitecture為 Lambda 指令集架構。

如果您指定makefile，請提供自訂 makefile，您可以在其中宣告包含圖層建置命令build-*layer-logical-id*之表單的建置目標。您的 makefile 負責在必要時編譯層，並將組建加工品複製到工作流程中後續步驟所需的適當位置。makefile 的位置由圖層資源的ContentUri屬性指定，並且必須命名Makefile。

> **Note**
>
> 創建自定義圖層時， AWS Lambda 取決於環境變量來查找圖層代碼。Lambda 執行階段包括將圖層程式碼複製到的/opt目錄中的路徑。專案的建置成品資料夾結構必須與執行階段預期的資料夾結構相符，才能找到您的自訂圖層程式碼。
> 例如，對於 Python，您可以將代碼放在python/子目錄中。對於 NodeJS，您可以將代碼放在子目錄中nodejs/node_modules/。
> 如需詳細資訊，請參閱《AWS Lambda 開發人員指南》中的在圖層中包含程式庫相依性。

以下是範例Metadata資源屬性區段。

```
Metadata:
  BuildMethod: python3.8
  BuildArchitecture: arm64
```

> **Note**
>
> 如果不包含Metadata資源屬性部分，則不 AWS SAM 會構建圖層。而是從圖層資源CodeUri屬性中指定的位置複製組建加工品。如需詳細資訊，請參閱AWS::Serverless::LayerVersion資源類型的ContentUri內容。

當您包括資Metadata源屬性區段時，您可以使用sam build指令將圖層建置為獨立物件或 AWS Lambda 函數的相依性。

- 作為一個獨立的對象。您可能只想要建置圖層物件，例如當您在本機測試圖層的程式碼變更，而不需要建置整個應用程式時。若要獨立建置圖層，請使用指sam build *layer-logical-id*令指定圖層資源。
- 作為 Lambda 函數的依賴關係。當您在相同 AWS SAM 範本檔案的 Lambda 函數Layers屬性中包含層的邏輯 ID 時，該層就是該 Lambda 函數的相依性。當該圖層還包含含有BuildMethod項目的Metadata資源屬性區段時，您可以透過使用指令建置整個應用程式，或透過使用指sam build令指定函數資源來建立圖層。sam build *function-logical-id*

範例

範本範例 1：針對 Python 3.9 執行階段環境建置圖層

下列範例 AWS SAM 範本會針對 Python 3.9 執行階段環境建置圖層。

```
Resources:
  MyLayer:
    Type: AWS::Serverless::LayerVersion
    Properties:
      ContentUri: my_layer
      CompatibleRuntimes:
        - python3.9
    Metadata:
      BuildMethod: python3.9   # Required to have AWS SAM build this layer
```

模板示例 2：使用自定義生成文件構建圖層

下列範例 AWS SAM 範本使用自訂`makefile`來建立圖層。

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

以下`makefile`包含構建目標和將要執行的命令。請注意，`ContentUri`屬性設定為`my_layer`，因此 makefile 必須位於`my_layer`子目錄的根目錄中，且檔案名稱必須是。`Makefile`另請注意，構建加工品被複製到`python/`子目錄中 AWS Lambda ，以便能夠找到層代碼。

```
build-MyLayer:
  mkdir -p "$(ARTIFACTS_DIR)/python"
  cp *.py "$(ARTIFACTS_DIR)/python"
  python -m pip install -r requirements.txt -t "$(ARTIFACTS_DIR)/python"
```

示例 sam 構建命令

下列`sam build`指令會建置包含`Metadata`資源屬性區段的圖層。

```
# 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
```

測試您的無伺服器應用程式 AWS SAM

撰寫並建立應用程式之後，您就可以準備測試應用程式，以驗證應用程式是否正常運作。使用命 AWS SAM 令列介面 (CLI)，您可以在本機測試無伺服器應用程式，然後再將其上傳到 AWS 雲端。測試應用程式可協助您確認應用程式的功能、可靠性和效能，同時識別需要解決的問題 (錯誤)。

本節提供測試應用程式時可遵循的常見做法指引。本節中的主題主要集中在您在部署到 AWS 雲端之前可以執行的本機測試。部署前進行測試可協助您主動識別問題，減少與部署問題相關的不必要成本。本節中的每個主題都會說明您可以執行的測試、說明使用測試的優點，以及示範如何執行測試的範例。測試應用程式之後，您就可以準備好偵錯發現的任何問題。

主題

- 使用sam local指令進行測試簡介
- 使用本機叫用 Lambda 函數 AWS SAM
- 在本機執行 API Gateway AWS SAM
- 雲端測試簡介 sam remote test-event
- 雲端測試簡介 sam remote invoke
- 自動執行本機整合測試 AWS SAM
- 產生範例事件承載

使用sam local指令進行測試簡介

使用指 AWS Serverless Application Model 令行介面 (AWS SAMCLI) `sam local` 指令在本機測試您的無伺服器應用程式。

如需「」的簡介 AWS SAMCLI，請參閱什麼是 AWS SAMCLI？。

必要條件

若要使用`sam local`，請完成下列 AWS SAMCLI步驟來安裝：

- AWS SAM 前提.
- 安裝 AWS SAMCLI.

在使用之前`sam local`，我們建議對以下內容進行基本了解：

- 配置 AWS SAMCLI.
- 使用sam init指令建立您的應用程式.
- 使用sam build指令建立簡介.
- 使用sam deploy指令部署簡介.

使用指sam local令

使用該`sam local`命令搭配其任何子命令，為您的應用程式執行不同類型的本機測試。

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

若要進一步瞭解每個子指令，請參閱下列內容：

- 介紹到 sam local generate-event— 產生用於本機測試的 AWS 服務 事件。
- 介紹到 sam local invoke— 在本機啟動 AWS Lambda 函式的一次性叫用。
- 介紹到 sam local start-api— 使用本機 HTTP 伺服器執行您的 Lambda 函數。
- 介紹到 sam local start-lambda— 使用本機 HTTP 伺服器執行 Lambda 函數，以搭配 AWS CLI 或開發套件使用。

測試簡介 sam local generate-event

使用命 AWS Serverless Application Model 令列介面 (AWS SAMCLI) `sam local generate-event` 子命令產生受支援 AWS 服務的事件承載範例。然後，您可以修改這些事件並將其傳遞給本機資源以進行測試。

- 如需「」的簡介 AWS SAMCLI，請參閱什麼是 AWS SAMCLI？。
- 如需指`sam local generate-event`令選項的清單，請參閱sam local generate-event。

事件是 JSON 物件，會在 AWS 服務 執行動作或工作時產生。這些事件包含特定資訊，例如已處理的資料或事件的時間戳記。大多數 AWS 服務 生成事件，每個服務的事件都是針對其服務的唯一格式化。

由一個服務產生的事件會作為事件來源傳遞至其他服務。例如，放置在 Amazon Simple Storage Service (Amazon S3) 貯體中的項目可以產生事件。然後，此事件可用作 AWS Lambda 函數的事件來源，以進一步處理資料。

您使用產生的事件`sam local generate-event`會以與 AWS 服務建立的實際事件相同的結構格式化。您可以修改這些事件的內容，並使用它們來測試應用程式中的資源。

必要條件

若要使用`sam local generate-event`，請完成下列 AWS SAMCLI步驟來安裝：

- AWS SAM 前提.
- 安裝 AWS SAMCLI.

使用前`sam local generate-event`，我們建議對以下內容進行基本了解：

- 配置 AWS SAMCLI.
- 使用sam init指令建立您的應用程式.
- 使用sam build指令建立簡介.
- 使用sam deploy指令部署簡介.

產生範例事件

使用 AWS SAMCLI`sam local generate-event`子指令產生受支援 AWS 服務的事件。

若要查看支援的清單 AWS 服務

1. 執行下列命令：

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

2. AWS 服務 將顯示支援的清單。以下是範例：

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

```
  ...
```

產生本機事件

1. 執行sam local generate-event並提供支援的服務名稱。這將顯示您可以產生的事件類型清單。以下是範例：

```
$ sam local generate-event s3

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

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

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

2. 若要產生範例事件，請執行sam local generate-event，並提供服務和事件類型。

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

以下是範例：

```
$ sam local generate-event s3 put
{
  "Records": [
    {
      "eventVersion": "2.0",
      "eventSource": "aws:s3",
      "awsRegion": "us-east-1",
      "eventTime": "1970-01-01T00:00:00.000Z",
      "eventName": "ObjectCreated:Put",
      "userIdentity": {
        "principalId": "EXAMPLE"
      },
      "requestParameters": {
        "sourceIPAddress": "127.0.0.1"
      },
      "responseElements": {
```

```
        "x-amz-request-id": "EXAMPLE123456789",
        "x-amz-id-2": "EXAMPLE123/5678abcdefghijklambdaisawesome/
mnopqrstuvwxyzABCDEFGH"
      },
      "s3": {
        "s3SchemaVersion": "1.0",
        "configurationId": "testConfigRule",
        "bucket": {
          "name": "example-bucket",
          "ownerIdentity": {
            "principalId": "EXAMPLE"
          },
          "arn": "arn:aws:s3:::example-bucket"
        },
        "object": {
          "key": "test/key",
          "size": 1024,
          "eTag": "0123456789abcdef0123456789abcdef",
          "sequencer": "0A1B2C3D4E5F678901"
        }
      }
    }
  ]
}
```

這些範例事件包含預留位置值。您可以修改這些值以參考應用程式中的實際資源或值，以協助進行本機測試。

若要修改範例事件

1. 您可以在命令提示字元中修改範例事件。若要查看您的選項，請執行下列指令：

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

   以下是範例：

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

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

   Options:
   ```

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

2. 在命令提示字元中使用這些選項中的任何一個來修改範例事件承載。以下是範例：

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

{
  "Records": [
    {
      "eventVersion": "2.0",
      "eventSource": "aws:s3",
      "awsRegion": "us-east-1",
      "eventTime": "1970-01-01T00:00:00.000Z",
      "eventName": "ObjectCreated:Put",
      "userIdentity": {
        "principalId": "EXAMPLE"
      },
      "requestParameters": {
        "sourceIPAddress": "127.0.0.1"
      },
      "responseElements": {
        "x-amz-request-id": "EXAMPLE123456789",
        "x-amz-id-2": "EXAMPLE123/5678abcdefghijklambdaisawesome/
mnopqrstuvwxyzABCDEFGH"
      },
      "s3": {
        "s3SchemaVersion": "1.0",
        "configurationId": "testConfigRule",
        "bucket": {
          "name": "MyBucket",
```

```
          "ownerIdentity": {
            "principalId": "EXAMPLE"
          },
          "arn": "arn:aws:s3:::MyBucket"
        },
        "object": {
          "key": "test/key",
          "size": 1024,
          "eTag": "0123456789abcdef0123456789abcdef",
          "sequencer": "0A1B2C3D4E5F678901"
        }
      }
    }
  ]
}
```

使用產生的事件進行本機測試

在本地保存生成的事件，並使用其他sam local子命令進行測試。

若要在本機儲存您產生的事件

- 執行下列命令：

  ```
  $ sam local generate-event <service> <event> <event-option> > <filename.json>
  ```

  以下是一個事件被保存為我們項目s3.json文件events夾中的文件的示例。

  ```
  sam-app$ sam local generate-event s3 put --bucket MyBucket > events/s3.json
  ```

若要使用產生的事件進行本機測試

- 使用--event選項將事件與其他sam local子命令一起傳遞。

  以下是使用s3.json事件在本機叫用 Lambda 函數的範例：

  ```
  sam-app$ sam local invoke --event events/s3.json S3JsonLoggerFunction

  Invoking src/handlers/s3-json-logger.s3JsonLoggerHandler (nodejs18.x)
  Local image is up-to-date
  ```

```
Using local image: public.ecr.aws/lambda/nodejs:18-rapid-x86_64.

Mounting /Users/.../sam-app/.aws-sam/build/S3JsonLoggerFunction as /var/
task:ro,delegated, inside runtime container
START RequestId: f4f45b6d-2ec6-4235-bc7b-495ec2ae0128 Version: $LATEST
END RequestId: f4f45b6d-2ec6-4235-bc7b-495ec2ae0128
REPORT RequestId: f4f45b6d-2ec6-4235-bc7b-495ec2ae0128  Init Duration: 1.23 ms
 Duration: 9371.93 ms      Billed Duration: 9372 ms        Memory Size: 128 MB
 Max Memory Used: 128 MB
```

進一步了解

如需所有`sam local generate-event`選項的清單，請參閱sam local generate-event。

有關使用的演示`sam local`，請參AWS SAM 閱本地開發。在 SAM 系列的無伺服器陸地工作階段中，測試來自本機開發環境的 AWS 雲端 YouTube資源。

測試簡介 sam local invoke

使用指 AWS Serverless Application Model 令行介面 (AWS SAMCLI) `sam local invoke` 子指令在本機啟動 AWS Lambda 函數的一次性呼叫。

- 如需「」的簡介 AWS SAMCLI，請參閱什麼是 AWS SAMCLI？。
- 如需指`sam local invoke`令選項的清單，請參閱sam local invoke。
- 如需在典型開發工作流程`sam local invoke`期間使用的範例，請參閱步驟 7：（可選）在本地測試您的應用程序。

必要條件

若要使用`sam local invoke`，請完成下列 AWS SAMCLI步驟來安裝：

- AWS SAM 前提.
- 安裝 AWS SAMCLI.

使用前`sam local invoke`，我們建議對以下內容進行基本了解：

- 配置 AWS SAMCLI.
- 使用sam init指令建立您的應用程式.

- 使用sam build指令建立簡介.
- 使用sam deploy指令部署簡介.

在本機叫用 Lambda 函數

當您執行時`sam local invoke`，會 AWS SAMCLI假設您目前的工作目錄是專案的根目錄。 AWS SAMCLI將首先查找子`template.[yaml|yml]`文件`.aws-sam`夾中的文件。如果找不到， AWS SAMCLI會在您目前的工作目錄中尋找`template.[yaml|yml]`檔案。

若要在本機叫用 Lambda 函數

1. 從項目的根目錄中運行以下命令：

```
$ sam local invoke <options>
```

2. 如果您的應用程式包含多個函數，請提供函數的邏輯 ID。以下是範例：

```
$ sam local invoke HelloWorldFunction
```

3. 將您的函數 AWS SAMCLI構建在本地容器中使用Docker. 然後它調用你的函數並輸出你的函數的響應。

   以下是範例：

```
$ sam local invoke
Invoking app.lambda_handler (python3.9)
Local image is out of date and will be updated to the latest runtime. To skip this,
 pass in the parameter --skip-pull-image
Building
 image.....................................................................................
Using local image: public.ecr.aws/lambda/python:3.9-rapid-x86_64.

Mounting /Users/.../sam-app/.aws-sam/build/HelloWorldFunction as /var/
task:ro,delegated, inside runtime container
START RequestId: 64bf7e54-5509-4762-a97c-3d740498d3df Version: $LATEST
END RequestId: 64bf7e54-5509-4762-a97c-3d740498d3df
REPORT RequestId: 64bf7e54-5509-4762-a97c-3d740498d3df  Init Duration: 1.09 ms
  Duration: 608.42 ms       Billed Duration: 609 ms Memory Size: 128 MB     Max
 Memory Used: 128 MB
{"statusCode": 200, "body": "{\"message\": \"hello world\"}"}%
```

管理 日誌

使用時`sam local invoke`，Lambda 函數執行階段輸出 (例如記錄檔) 會輸出至`stderr`，並將 Lambda 函數結果輸出至`stdout`。

以下是基本 Lambda 函數的範例：

```
def handler(event, context):
    print("some log") # this goes to stderr
    return "hello world" # this goes to stdout
```

您可以儲存這些標準輸出。以下是範例：

```
$ sam local invoke 1> stdout.log
...

$ cat stdout.log
"hello world"

$ sam local invoke 2> stderr.log
...

$ cat stderr.log
Invoking app.lambda_handler (python3.9)
Local image is up-to-date
Using local image: public.ecr.aws/lambda/python:3.9-rapid-x86_64.
Mounting /Users/.../sam-app/.aws-sam/build/HelloWorldFunction as /var/
task:ro,delegated, inside runtime container
START RequestId: 0b46e646-3bdf-4b58-8beb-242d00912c46 Version: $LATEST
some log
END RequestId: 0b46e646-3bdf-4b58-8beb-242d00912c46
REPORT RequestId: 0b46e646-3bdf-4b58-8beb-242d00912c46  Init Duration: 0.91 ms
 Duration: 589.19 ms Billed Duration: 590 ms Memory Size: 128 MB Max Memory Used: 128
 MB
```

您可以使用這些標準輸出來進一步自動化您的本機開發程序。

選項

傳遞自訂事件以叫用 Lambda 函數

若要將事件傳遞至 Lambda 函數，請使用`--event`選項。以下是範例：

```
$ sam local invoke --event events/s3.json S3JsonLoggerFunction
```

您可以使用sam local generate-event子命令建立事件。如需進一步了解，請參閱測試簡介 sam local generate-event。

調用 Lambda 函數時傳遞環境變量

如果您的 Lambda 函數使用環境變數，您可以在本機測試期間使用--env-vars此選項來傳遞它們。這是使用應用程式中已部署雲端的服務在本機測試 Lambda 函數的好方法。以下是範例：

```
$ sam local invoke --env-vars locals.json
```

指定範本或函數

若要指定要參考 AWS SAMCLI的範本，請使用--template選項。 AWS SAMCLI將加載該 AWS SAM 模板及其指向的資源。

若要叫用巢狀應用程式或堆疊的函數，請提供應用程式或堆疊邏輯 ID 以及函數邏輯 ID。以下是範例：

```
$ sam local invoke StackLogicalId/FunctionLogicalId
```

從您的Terraform專案測試 Lambda 函數

使用--hook-name此選項在本機測試Terraform專案中的 Lambda 函數。如需進一步了解，請參閱使用 AWS SAMCLI與進Terraform行本機除錯和測試。

以下是範例：

```
$ sam local invoke --hook-name terraform --beta-features
```

最佳實務

如果您的應用程式有執行中的.aws-sam目錄sam build，請務必在sam build每次更新函數程式碼時執行。然後，運行sam local invoke以在本地測試更新的函數代碼。

本地測試是部署到雲之前快速開發和測試的絕佳解決方案。但是，本地測試不會驗證所有內容，例如雲中資源之間的權限。請盡可能在雲端中測試您的應用程式。我們建議您使sam sync用以加速雲端測試工作流程。

範例

產生 Amazon API Gateway 範例事件，並使用它在本機叫用 Lambda 函數

首先，我們產生 API Gateway HTTP API 事件承載，並將其儲存到我們的`events`資料夾。

```
$ sam local generate-event apigateway http-api-proxy > events/apigateway_event.json
```

接下來，我們修改 Lambda 函數，以便從事件傳回參數值。

```
def lambda_handler(event, context):
    print("HelloWorldFunction invoked")
    return {
        "statusCode": 200,
        "body": json.dumps({
            "message": event['queryStringParameters']['parameter2'],
        }),
    }
```

接下來，我們在本機叫用 Lambda 函數並提供自訂事件。

```
$ sam local invoke --event events/apigateway_event.json

Invoking app.lambda_handler (python3.9)
Local image is up-to-date
Using local image: public.ecr.aws/lambda/python:3.9-rapid-x86_64.

Mounting /Users/...sam-app/.aws-sam/build/HelloWorldFunction as /var/task:ro,delegated,
 inside runtime container
START RequestId: 59535d0d-3d9e-493d-8c98-6264e8e961b8 Version: $LATEST
some log
END RequestId: 59535d0d-3d9e-493d-8c98-6264e8e961b8
REPORT RequestId: 59535d0d-3d9e-493d-8c98-6264e8e961b8  Init Duration: 1.63 ms
 Duration: 564.07 ms       Billed Duration: 565 ms Memory Size: 128 MB     Max Memory
 Used: 128 MB
{"statusCode": 200, "body": "{\"message\": \"value\"}"}%
```

在本地調用 Lambda 函數時傳遞環境變量

此應用程式具有使用環境變數做為 Amazon DynamoDB 表格名稱的 Lambda 函數。以下是在 AWS SAM 模板中定義的函數的示例：

```
AWSTemplateFormatVersion: 2010-09-09
Transform: AWS::Serverless-2016-10-31
...
Resources:
  getAllItemsFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: src/get-all-items.getAllItemsHandler
      Description: get all items
      Policies:
        - DynamoDBReadPolicy:
            TableName: !Ref SampleTable
      Environment:
        Variables:
          SAMPLE_TABLE: !Ref SampleTable
...
```

我們希望在本機測試 Lambda 函數，同時讓它與雲端中的 DynamoDB 表互動。要做到這一點，我們創建我們的環境變量文件，並將其保存在我們的項目的根目錄`locals.json`。此處提供的值供`SAMPLE_TABLE`參考雲端中的 DynamoDB 表格。

```
{
    "getAllItemsFunction": {
        "SAMPLE_TABLE": "dev-demo-SampleTable-1U991234LD5UM98"
    }
}
```

接下來，我們運行`sam local invoke`並通過我們的環境變量與選`--env-vars`項。

```
$ sam local invoke getAllItemsFunction --env-vars locals.json

Mounting /Users/...sam-app/.aws-sam/build/HelloWorldFunction as /var/task:ro,delegated,
 inside runtime container
START RequestId: 59535d0d-3d9e-493d-8c98-6264e8e961b8 Version: $LATEST
some log
END RequestId: 59535d0d-3d9e-493d-8c98-6264e8e961b8
REPORT RequestId: 59535d0d-3d9e-493d-8c98-6264e8e961b8  Init Duration: 1.63 ms
 Duration: 564.07 ms       Billed Duration: 565 ms Memory Size: 128 MB     Max Memory
 Used: 128 MB
{"statusCode":200,"body":"{}"}
```

進一步了解

如需所有`sam local invoke`選項的清單，請參閱sam local invoke。

有關使用的演示`sam local`，請參AWS SAM 閱本地開發。在 SAM 系列的無伺服器陸地工作階段中，測試來自本機開發環境的 AWS 雲端 YouTube資源。

測試簡介 sam local start-api

使用命 AWS Serverless Application Model 令列介面 (AWS SAMCLI) `sam local start-api` 子命令在本機執行 AWS Lambda 函數，並透過本機 HTTP 伺服器主機進行測試。這種類型的測試對於 Amazon API Gateway 端點叫用的 Lambda 函數很有幫助。

- 如需「」的簡介 AWS SAMCLI，請參閱什麼是 AWS SAMCLI？。
- 如需指`sam local start-api`令選項的清單，請參閱sam local start-api。
- 如需在典型開發工作流程`sam local start-api`期間使用的範例，請參閱步驟 7：（可選）在本地測試您的應用程序。

必要條件

若要使用`sam local start-api`，請完成下列 AWS SAMCLI步驟來安裝：

- AWS SAM 前提.
- 安裝 AWS SAMCLI.

在使用之前`sam local start-api`，我們建議對以下內容進行基本了解：

- 配置 AWS SAMCLI.
- 使用sam init指令建立您的應用程式.
- 使用sam build指令建立簡介.
- 使用sam deploy指令部署簡介.

使用山姆本地啟動 API

當您執行時`sam local start-api`，會 AWS SAMCLI假設您目前的工作目錄是專案的根目錄。AWS SAMCLI將首先查找子`template.[yaml|yml]`文件`.aws-sam`夾中的文件。如果找不到，AWS SAMCLI會在您目前的工作目錄中尋找`template.[yaml|yml]`檔案。

若要啟動本機 HTTP 伺服器

1. 從專案的根目錄中，執行下列命令：

```
$ sam local start-api <options>
```

2. 這會在本機Docker容器中 AWS SAMCLI建置您的 Lambda 函數。然後它會輸出 HTTP 伺服器端點的本機位址。以下是範例：

```
$ sam local start-api

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

Mounting /Users/.../sam-app/.aws-sam/build/HelloWorldFunction as /var/
task:ro,delegated, inside runtime container
Containers Initialization is done.
Mounting HelloWorldFunction at http://127.0.0.1:3000/hello [GET]
You can now browse to the above endpoints to invoke your functions. You do not
 need to restart/reload SAM CLI while working on your functions, changes will be
 reflected instantly/automatically. If you used sam build before running local
 commands, you will need to re-run sam build for the changes to be picked up. You
 only need to restart SAM CLI if you update your AWS SAM template
2023-04-12 14:41:05 WARNING: This is a development server. Do not use it in a
 production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:3000
```

3. 您可以透過瀏覽器或命令提示字元叫用 Lambda 函數。以下是範例：

```
sam-app$ curl http://127.0.0.1:3000/hello
{"message": "Hello world!"}%
```

4. 當您變更 Lambda 函數程式碼時，請考慮下列事項來重新整理本機 HTTP 伺服器：

   - 如果您的應用程序沒有.aws-sam目錄，並且您的函數使用解釋型語言，則 AWS SAMCLI會通過創建新容器並託管它來自動更新您的函數。
   - 如果您的應用程序確實有一個.aws-sam目錄，則需要運行sam build以更新您的功能。然後sam local start-api再次運行以託管該函數。
   - 如果您的函數使用已編譯的語言，或者您的項目需要複雜的包裝支持，請運行自己的構建解決方案來更新您的函數。然後sam local start-api再次運行以託管該函數。

使用 Lambda 授權器的 Lambda 函數

> **Note**
>
> 此功能是 1.80.0 AWS SAMCLI 版本中的新功能。若要升級，請參閱升級 AWS SAMCLI。

對於使用 Lambda 授權器的 Lambda 函數，在叫用 Lambda 函數端點之前， AWS SAMCLI會自動叫用您的 Lambda 授權器。

以下是針對使用 Lambda 授權器的函數啟動本機 HTTP 伺服器的範例：

```
$ sam local start-api
2023-04-17 15:02:13 Attaching import module proxy for analyzing dynamic imports

AWS SAM CLI does not guarantee 100% fidelity between authorizers locally
and authorizers deployed on AWS. Any application critical behavior should
be validated thoroughly before deploying to production.

Testing application behaviour against authorizers deployed on AWS can be done using the
 sam sync command.

Mounting HelloWorldFunction at http://127.0.0.1:3000/authorized-request [GET]
You can now browse to the above endpoints to invoke your functions. You do not need
 to restart/reload SAM CLI while working on your functions, changes will be reflected
 instantly/automatically. If you used sam build before running local commands, you will
 need to re-run sam build for the changes to be picked up. You only need to restart SAM
 CLI if you update your AWS SAM template
2023-04-17 15:02:13 WARNING: This is a development server. Do not use it in a
 production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:3000
2023-04-17 15:02:13 Press CTRL+C to quit
```

當您透過本機 HTTP 伺服器叫用 Lambda 函數端點時， AWS SAMCLI第一個會叫用您的 Lambda 授權者。如果授權成功， AWS SAMCLI將會叫用您的 Lambda 函數端點。以下是範例：

```
$ curl http://127.0.0.1:3000/authorized-request --header "header:my_token"
{"message": "from authorizer"}%

Invoking app.authorizer_handler (python3.8)
Local image is up-to-date
Using local image: public.ecr.aws/lambda/python:3.8-rapid-x86_64.
```

```
Mounting /Users/.../sam-app/... as /var/task:ro,delegated, inside runtime container
START RequestId: 38d3b472-a2c8-4ea6-9a77-9b386989bef0 Version: $LATEST
END RequestId: 38d3b472-a2c8-4ea6-9a77-9b386989bef0
REPORT RequestId: 38d3b472-a2c8-4ea6-9a77-9b386989bef0    Init Duration: 1.08 ms
 Duration: 628.26 msBilled Duration: 629 ms    Memory Size: 128 MB    Max Memory Used:
 128 MB
Invoking app.request_handler (python3.8)
Using local image: public.ecr.aws/lambda/python:3.8-rapid-x86_64.

Mounting /Users/.../sam-app/... as /var/task:ro,delegated, inside runtime container
START RequestId: fdc12255-79a3-4365-97e9-9459d06446ff Version: $LATEST
END RequestId: fdc12255-79a3-4365-97e9-9459d06446ff
REPORT RequestId: fdc12255-79a3-4365-97e9-9459d06446ff    Init Duration: 0.95 ms
 Duration: 659.13 msBilled Duration: 660 ms    Memory Size: 128 MB    Max Memory Used:
 128 MB
No Content-Type given. Defaulting to 'application/json'.
2023-04-17 15:03:03 127.0.0.1 - - [17/Apr/2023 15:03:03] "GET /authorized-request
 HTTP/1.1" 200 -
```

選項

不斷重用容器以加速本地函數調用

默認情況下，每次通過本地 HTTP 服務器調用函數時 AWS SAMCLI創建一個新的容器。使用`--warm-containers`此選項可自動重複使用容器進行函數叫用。這可加快準備 Lambda 函數 AWS SAMCLI以進行本機叫用所需的時間。您可以透過提供`eager`或`lazy`引數來進一步自訂此選項。

- `eager`— 所有函數的容器都會在啟動時載入，並在呼叫之間持續存在。
- `lazy`— 只有在第一次叫用每個函數時，才會載入容器。然後，它們會繼續進行其他調用。

以下是範例：

```
$ sam local start-api --warm-containers eager
```

使用`--warm-containers`和修改 Lambda 函數程式碼時：

- 如果您的應用程序具有`.aws-sam`目錄，請運行`sam build`以更新應用程序構建成品中的函數代碼。
- 偵測到程式碼變更時，會 AWS SAMCLI自動關閉 Lambda 函數容器。

- 當您再次調用該函數時， AWS SAMCLI會自動創建一個新的容器。

指定要用於 Lambda 函數的容器映像檔

預設情況下， AWS SAMCLI使用來自亞馬遜彈性容器登錄 (Amazon ECR) 的 Lambda 基礎映像檔在本機叫用您的函數。使用`--invoke-image`此選項可參考自訂容器映像檔。以下是範例：

```
$ sam local start-api --invoke-image public.ecr.aws/sam/emu-python3.8
```

您可以指定要與自訂容器映像檔搭配使用的函數。以下是範例：

```
$ sam local start-api --invoke-image Function1=amazon/aws/sam-cli-emulation-image-python3.8
```

指定要在本機測試的範本

若要指定要參考 AWS SAMCLI的範本，請使用`--template`選項。 AWS SAMCLI將只加載該 AWS SAM 模板及其指向的資源。以下是範例：

```
$ sam local start-api --template myTemplate.yaml
```

指定 Lambda 函數的主機開發環境

根據預設，`sam local start-api`子指令會使用 localhost IP 位址`127.0.0.1`建立 HTTP 伺服器。如果您的本機開發環境與本機電腦隔離，您可以自訂這些值。

使用`--container-host`選項指定主機。以下是範例：

```
$ sam local start-api --container-host host.docker.internal
```

使用此`--container-host-interface`選項可指定容器連接埠應繫結之主機網路的 IP 位址。以下是範例：

```
$ sam local start-api --container-host-interface 0.0.0.0
```

最佳實務

如果您的應用程式有執行中的`.aws-sam`目錄`sam build`，請務必在`sam build`每次更新函數程式碼時執行。然後，運行`sam local start-api`以在本地測試更新的函數代碼。

本地測試是部署到雲之前快速開發和測試的絕佳解決方案。但是，本地測試不會驗證所有內容，例如雲中資源之間的權限。請盡可能在雲端中測試您的應用程式。我們建議您使sam sync用以加速雲端測試工作流程。

進一步了解

如需所有sam local start-api選項的清單，請參閱sam local start-api。

測試簡介 sam local start-lambda

使用指 AWS Serverless Application Model 令行介面 (AWS SAMCLI) sam local start-lambda 子指令透過 AWS Command Line Interface (AWS CLI) 或 SDK 叫用您的 AWS Lambda 函數。此指令會啟動模擬 AWS Lambda的本機端點。

- 如需「」的簡介 AWS SAMCLI，請參閱什麼是 AWS SAMCLI？。
- 如需指sam local start-lambda令選項的清單，請參閱sam local start-lambda。

必要條件

若要使用sam local start-lambda，請完成下列 AWS SAMCLI步驟來安裝：

- AWS SAM 前提.
- 安裝 AWS SAMCLI.

在使用之前sam local start-lambda，我們建議對以下內容進行基本了解：

- 配置 AWS SAMCLI.
- 使用sam init指令建立您的應用程式.
- 使用sam build指令建立簡介.
- 使用sam deploy指令部署簡介.

使用山姆本地啟動

當您執行時sam local start-lambda，會 AWS SAMCLI假設您目前的工作目錄是專案的根目錄。 AWS SAMCLI將首先查找子template.[yaml|yml]文件.aws-sam夾中的文件。如果找不到，AWS SAMCLI會在您目前的工作目錄中尋找template.[yaml|yml]檔案。

要使用山姆本地開始-拉姆達

1. 從專案的根目錄中，執行下列命令：

```
$ sam local start-lambda <options>
```

2. 這會在本機Docker容器中 AWS SAMCLI建置您的 Lambda 函數。然後，它將本地地址輸出到您的 HTTP 服務器端點。以下是範例：

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

Mounting /Users/.../sam-app/hello_world as /var/task:ro,delegated, inside runtime
 container
Containers Initialization is done.
Starting the Local Lambda Service. You can now invoke your Lambda Functions defined
 in your template through the endpoint.
2023-04-13 07:25:43 WARNING: This is a development server. Do not use it in a
 production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:3001
2023-04-13 07:25:43 Press CTRL+C to quit
```

3. 使用 AWS CLI 或開發套件在本機叫用 Lambda 函數。

   以下是使用下列項目的範例 AWS CLI：

```
$ aws lambda invoke --function-name "HelloWorldFunction" --endpoint-
url "http://127.0.0.1:3001" --no-verify-ssl out.txt

StatusCode: 200
(END)
```

   以下是使用 in 的範 AWS SDK例Python：

```
import boto3
from botocore.config import Config
from botocore import UNSIGNED

lambda_client = boto3.client('lambda',
                             endpoint_url="http://127.0.0.1:3001",
```

```
                                use_ssl=False,
                                verify=False,
                                config=Config(signature_version=UNSIGNED,
                                              read_timeout=1,
                                              retries={'max_attempts': 0}
                                              )
                               )
lambda_client.invoke(FunctionName="HelloWorldFunction")
```

選項

指定範本

若要指定要參考 AWS SAMCLI的範本，請使用`--template`選項。 AWS SAMCLI將只加載該 AWS SAM 模板及其指向的資源。以下是範例：

```
$ sam local start-lambda --template myTemplate.yaml
```

最佳實務

如果您的應用程式有執行中的`.aws-sam`目錄`sam build`，請務必在`sam build`每次更新函數程式碼時執行。然後，運行`sam local start-lambda`以在本地測試更新的函數代碼。

本地測試是部署到雲之前快速開發和測試的絕佳解決方案。但是，本地測試不會驗證所有內容，例如雲中資源之間的權限。請盡可能在雲端中測試您的應用程式。我們建議您使`sam sync`用以加速雲端測試工作流程。

進一步了解

如需所有`sam local start-lambda`選項的清單，請參閱sam local start-lambda。

使用本機叫用 Lambda 函數 AWS SAM

在雲端測試或部署之前，先在本機叫用 Lambda 函數可以有多種好處。它允許您更快地測試函數的邏輯。首先在本機測試可減少在雲端測試或部署期間識別問題的可能性，這有助於避免不必要的成本。此外，本地測試使調試更容易完成。

您可以使用sam local invoke命令並提供函數的邏輯 ID 和事件檔案，在本機叫用 Lambda 函數。 sam local invoke也接受`stdin`作為一個事件。如需有關事件的詳細資訊，請參閱AWS Lambda 開發人員指

南中的事件。如需不同 AWS 服務之事件訊息格式的相關資訊，請參閱AWS Lambda 開發人員指南中的AWS Lambda 與其他服務搭配使用。

> **Note**
>
> 該sam local invoke命令對應於 AWS Command Line Interface (AWS CLI) 命令aws lambda invoke。您可以使用任一命令來叫用 Lambda 函數。

您必須在包含要叫用之函數的專案目錄中執行sam local invoke命令。

範例：

```
# 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
```

環境變數檔案

若要在本機宣告覆寫範本中定義的值的環境變數，請執行下列動作：

1. 建立包含要覆寫之環境變數的 JSON 檔案。
2. 使用引--env-vars數來覆寫範本中定義的值。

聲明環境變量

若要宣告全域套用至所有資源的環境變數，請指定如下所示的Parameters物件：

```
{
    "Parameters": {
        "TABLE_NAME": "localtable",
        "BUCKET_NAME": "testBucket",
        "STAGE": "dev"
    }
}
```

要為每個資源聲明不同的環境變量，請為每個資源指定對象，如下所示：

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

為每個資源指定物件時，您可以使用下列識別碼，以最高到最低的優先順序列出：

1. `logical_id`
2. `function_id`
3. `function_name`
4. 完整路徑識別碼

您可以使用上述兩種方法，在單一檔案中一起宣告環境變數。這樣做時，您為特定資源提供的環境變數優先於全域環境變數。

將環境變數儲存在 JSON 檔案中，例如`env.json`.

覆寫環境變數值

若要以 JSON 檔案中定義的環境變數覆寫環境變`--env-vars`數，請搭配invoke或start-api指令使用引數。例如：

```
sam local invoke --env-vars env.json
```

圖層

如果您的應用程式包含層，如需有關如何偵錯本機主機上層問題的資訊，請參閱使用 Lambda 層來提高效率 AWS SAM。

進一步了解

如需在本機叫用函式的實際操作範例，請參閱完整 AWS SAM 研討會中的單元 2-在本機執行。

在本機執行 API Gateway AWS SAM

在本機執行 Amazon API Gateway 可以有多種好處。例如，在本機執行 API Gateway 可讓您在部署到 AWS 雲端之前在本機測試 API 端點。如果您先在本機測試，通常可以減少雲端中的測試和開發，這有助於降低成本。此外，在本機執行可讓偵錯更容易。

若要啟動可用來測試 HTTP 要求/回應功能的 API Gateway 本機執行個體，請使用指`sam local start-api` AWS SAMCLI令。此功能具有熱重新加載功能，因此您可以快速開發和迭代函數。

> **Note**
>
> 熱重新載入是指只重新整理變更的檔案，且應用程式的狀態保持不變。相反地，即時重新載入是重新整理整個應用程式，而且應用程式的狀態會遺失。

如需使用`sam local start-api`指令的指示，請參閱測試簡介 sam local start-api。

依預設， AWS SAM 會使用 AWS Lambda Proxy 整合並支援`HttpApi`和`Api`資源類型。如需有關資`HttpApi`源類型之代理整合的詳細資訊，請參閱 API Gateway 開發人員指南中的使用 HTTP API 的 AWS Lambda 代理整合。如需有關代理與`Api`資源類型整合的詳細資訊，請參閱《API Gateway 開發人員指南》中的了解 API Gateway Lambda 代理整合。

範例：

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

AWS SAM 會自動尋找 AWS SAM 範本中已定義`HttpApi`或`Api`事件來源的任何函數。然後，它會在定義的 HTTP 路徑上掛載函數。

在下列`Api`範例中，`Ratings`函數會掛載`ratings.py:handler()`/`ratings`於`GET`要求：

```
Ratings:
  Type: AWS::Serverless::Function
  Properties:
    Handler: ratings.handler
    Runtime: python3.9
    Events:
      Api:
        Type: Api
        Properties:
          Path: /ratings
```

```
          Method: get
```

這是一個示例Api響應：

```
// 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"
    });
}
```

如果您修改函數的程式碼，請執行的sam build命令sam local start-api以偵測變更。

環境變數檔案

若要在本機宣告覆寫範本中定義的值的環境變數，請執行下列動作：

1. 建立包含要覆寫之環境變數的 JSON 檔案。
2. 使用--env-vars引數覆寫範本中定義的值。

聲明環境變量

若要宣告全域套用至所有資源的環境變數，請指定如下所示的Parameters物件：

```
{
    "Parameters": {
        "TABLE_NAME": "localtable",
        "BUCKET_NAME": "testBucket",
        "STAGE": "dev"
    }
}
```

要為每個資源聲明不同的環境變量，請為每個資源指定對象，如下所示：

```
{
    "MyFunction1": {
        "TABLE_NAME": "localtable",
        "BUCKET_NAME": "testBucket",
    },
```

```
    "MyFunction2": {
        "TABLE_NAME": "localtable",
        "STAGE": "dev"
    }
}
```

為每個資源指定物件時，您可以使用下列識別碼，以最高到最低的優先順序列出：

1. `logical_id`
2. `function_id`
3. `function_name`
4. 完整路徑識別碼

您可以使用上述兩種方法，在單一檔案中一起宣告環境變數。這樣做時，您為特定資源提供的環境變數優先於全域環境變數。

將環境變數儲存在 JSON 檔案中，例如`env.json`.

覆寫環境變數值

若要以 JSON 檔案中定義的環境變數覆寫環境變`--env-vars`數，請搭配invoke或start-api指令使用引數。例如：

```
$ sam local start-api --env-vars env.json
```

圖層

如果您的應用程式包含層，如需有關如何偵錯本機主機上層問題的資訊，請參閱使用 Lambda 層來提高效率 AWS SAM。

雲端測試簡介 sam remote test-event

使用指 AWS Serverless Application Model 令行介面 (AWS SAM CLI) `sam remote test-event` 指令可存取和管理 AWS Lambda 函數的可共用測試事件。

若要進一步了解可共用的測試事件，請參閱AWS Lambda 開發人員指南中的可共用測試事件。

主題

- 設定 AWS SAMCLI要使用 sam remote test-event

- 使用指sam remote test-event令
- 使用可共享的測試事件
- 管理可共用的測試事件

必要條件

若要使用`sam remote test-event`，請完成下列 AWS SAMCLI步驟來安裝：

- AWS SAM 前提.
- 安裝 AWS SAMCLI.

如果您已經 AWS SAM CLI安裝了，我們建議您升級到最新版本的 AWS SAMCLI版本。如需進一步了解，請參閱升級 AWS SAMCLI。

使用前`sam remote test-event`，我們建議對以下內容進行基本了解：

- 配置 AWS SAMCLI.
- 使用sam init指令建立您的應用程式.
- 使用sam build指令建立簡介.
- 使用sam deploy指令部署簡介.
- 使用同步sam sync到簡介 AWS 雲端.

設定 AWS SAMCLI要使用 sam remote test-event

完成下列設定步驟以使用 AWS SAM CLI`sam remote test-event`指令：

1. 設定 AWS SAM CLI為使用您的 AWS 帳戶— Lambda 的可共用測試事件可由相同 AWS 帳戶的使用者存取和管理。若要將設定 AWS SAM CLI為使用您的 AWS 帳戶，請參閱配置 AWS SAMCLI。
2. 設定可共用測試事件的權限 — 若要存取和管理可共用的測試事件，您必須擁有適當的權限。若要深入了解，請參閱AWS Lambda 開發人員指南中的可共用測試事件。

使用指sam remote test-event令

該 AWS SAM CLI`sam remote test-event`命令提供了以下子命令，您可以用來訪問和管理可共享的測試事件：

- `delete`— 從 Amazon EventBridge 架構登錄檔中刪除可共用的測試事件。
- `get`— 從 EventBridge 結構描述登錄檔取得可共用的測試事件。
- `list`— 從 EventBridge 結構描述登錄中列出函數的現有可共用測試事件。
- `put`— 將事件從本機檔案儲存至 EventBridge 結構描述登錄。

若要使用列出這些子命令 AWS SAM CLI，請執行下列命令：

```
$ sam remote test-event --help
```

刪除可共享的測試事件

您可以使用子命令和以`delete`下命令來刪除可共享的測試事件：

- 提供要刪除的可共用測試事件的名稱。
- 提供與事件相關聯之 Lambda 函數的可接受識別碼。
- 如果您提供 Lambda 函數邏輯 ID，則還必須提供與 Lambda 函數相關聯的 AWS CloudFormation 堆疊名稱。

以下是範例：

```
$ sam remote test-event delete HelloWorldFunction --stack-name sam-app --name demo-
event
```

若要取得與`delete`子指令搭配使用的選項清單，請參閱〈〉sam remote test-event delete。您也可以從以下位置執行下列命令 AWS SAM CLI：

```
$ sam remote test-event delete --help
```

取得可共用的測試事件

您可以使用子命令以及`get`下列命令，從結 EventBridge 構描述登錄檔取得可共用的測試事件：

- 提供要取得之可共用測試事件的名稱。
- 提供與事件相關聯之 Lambda 函數的可接受識別碼。
- 如果您提供 Lambda 函數邏輯 ID，則還必須提供與 Lambda 函數相關聯的 AWS CloudFormation 堆疊名稱。

下列範例會取得名為demo-event與sam-app堆疊的 HelloWorldFunction Lambda 函數相關聯的可共用測試事件。此命令會將事件打印到您的控制台。

```
$ sam remote test-event get HelloWorldFunction --stack-name sam-app --name demo-event
```

若要取得可共用的測試事件並將其儲存至本機電腦，請使用選--output-file項並提供檔案路徑和名稱。以下是在目前工作目錄demo-event.json中另存demo-event為的範例：

```
$ sam remote test-event get HelloWorldFunction --stack-name sam-app --name demo-event
 --output-file demo-event.json
```

若要取得與get子指令搭配使用的選項清單，請參閱〈〉sam remote test-event get。您也可以從以下位置執行下列命令 AWS SAM CLI：

```
$ sam remote test-event get --help
```

列出可共享的測試事件

您可以從結構描述登錄列出特定 Lambda 函數的所有可共用測試事件。使用list子指令以及下列項目：

- 提供與事件相關聯之 Lambda 函數的可接受識別碼。
- 如果您提供 Lambda 函數邏輯 ID，則還必須提供與 Lambda 函數相關聯的 AWS CloudFormation 堆疊名稱。

以下是取得與堆疊 HelloWorldFunction Lambda 函數相關聯的所有可共用測試事件清單的sam-app範例：

```
$ sam remote test-event list HelloWorldFunction --stack-name sam-app
```

若要取得與list子指令搭配使用的選項清單，請參閱〈〉sam remote test-event list。您也可以從以下位置執行下列命令 AWS SAM CLI：

```
$ sam remote test-event list --help
```

儲存可共用的測試事件

您可以將可共用的測試事件儲存至 EventBridge 結構描述登錄。使用put子指令以及下列項目：

- 提供與可共用測試事件相關聯之 Lambda 函數的可接受識別碼。
- 提供可共用測試事件的名稱。
- 提供要上載之本機事件的檔案路徑和名稱。

下列範例會將本機`demo-event.json`事件另存為，`demo-event`並將其與`sam-app`堆疊的 `HelloWorldFunction` Lambda 函數產生關聯：

```
$ sam remote test-event put HelloWorldFunction --stack-name sam-app --name demo-event
 --file demo-event.json
```

如果 EventBridge 結構描述登錄中存在具有相同名稱的可共用測試事件， AWS SAM CLI將不會覆寫它。若要覆寫，請將選`--force`項新增至您的指令。

若要取得與`put`子指令搭配使用的選項清單，請參閱〈〉sam remote test-event put。您也可以從以下位置執行下列命令 AWS SAM CLI：

```
$ sam remote test-event put --help
```

使用可共享的測試事件

使用可共用的測試事件， AWS 雲端 透過`sam remote invoke`命令在中測試您的 Lambda 函數。如需進一步了解，請參閱將可共用的測試事件傳遞至雲端中的 Lambda 函數。

管理可共用的測試事件

本主題包含如何管理和使用可共用測試事件的範例。

獲取可共享的測試事件，對其進行修改並使用

您可以從 EventBridge 結構描述登錄檔取得可共用的測試事件、在本機修改該事件，然後將本機測試事件與中的 Lambda 函數搭配使用。 AWS 雲端以下是範例：

1. 擷取可共用的測試事件 — 使用`sam remote test-event get`子命令擷取特定 Lambda 函數的可共用測試事件，並將其儲存在本機：

   ```
   $ sam remote test-event get HelloWorldFunction --stack-name sam-app --name demo-event
    --output-file demo-event.json
   ```

2. 修改可共用的測試事件 — 使用您選擇的文字編輯器來修改可共用的測試事件。

3. 使用可共用的測試事件 — 使用`sam remote invoke`指令，並提供事件的檔案路徑和名稱：`--event-file`

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --event-file demo-event.json
```

取得可共用的測試事件、修改、上傳並使用

您可以從 EventBridge 結構描述登錄檔取得可共用的測試事件，在本機修改並上傳。然後，您可以將可共用的測試事件直接傳遞給. AWS 雲端以下是範例：

1. 擷取可共用的測試事件 — 使用`sam remote test-event get`子命令擷取特定 Lambda 函數的可共用測試事件，並將其儲存在本機：

```
$ sam remote test-event get HelloWorldFunction --stack-name sam-app --name demo-event
 --output-file demo-event.json
```

2. 修改可共用的測試事件 — 使用您選擇的文字編輯器來修改可共用的測試事件。
3. 上傳可共用的測試事件 — 使用`sam remote test-event put`子命令將可共用的測試事件上傳並儲存至結構描述 EventBridge 登錄。在此範例中，我們使用`--force`選項覆寫可共用測試的舊版本：

```
$ sam remote test-event put HelloWorldFunction --stack-name sam-app --name demo-event
 --file demo-event.json --force
```

4. 將可共用的測試事件傳遞至 Lambda 函數 — 使用`sam remote invoke`命令將可共用的測試事件直接傳遞給 Lambda 函數，位於下列位置： AWS 雲端

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --test-event-name demo-event
```

雲端測試簡介 sam remote invoke

使用指 AWS Serverless Application Model 令行介面 (AWS SAM CLI) `sam remote invoke` 指令與中支援的 AWS 資源互動 AWS 雲端。您可以使用`sam remote invoke`來調用以下資源：

- Amazon Kinesis Data Streams — 將資料記錄傳送至 Kinesis Data Streams 應用程式。

- AWS Lambda-調用事件並將其傳遞給您的 Lambda 函數。
- Amazon Simple Queue Service (Amazon SQS) — 將消息發送到 Amazon SQS 隊列。
- AWS Step Functions— 叫用 Step Functions 狀態機器以開始執行。

如需「」的簡介 AWS SAMCLI，請參閱什麼是 AWS SAMCLI？。

如需在典型開發工作流程`sam remote invoke`期間使用的範例，請參閱第 5 步：與您的功能進行交互 AWS 雲端。

主題

- 使用 sam 遠程調用命令
- 使用 sam 遠程調用命令選項
- 配置您的項目配置文件
- 範例
- 相關連結

必要條件

若要使用`sam remote invoke`，請完成下列 AWS SAMCLI步驟來安裝：

- AWS SAM 前提.
- 安裝 AWS SAMCLI.

我們也建議您升級到最新版本的 AWS SAMCLI. 如需進一步了解，請參閱升級 AWS SAMCLI。

使用前`sam remote invoke`，我們建議對以下內容進行基本了解：

- 配置 AWS SAMCLI.
- 使用sam init指令建立您的應用程式.
- 使用sam build指令建立簡介.
- 使用sam deploy指令部署簡介.
- 使用同步sam sync到簡介 AWS 雲端.

使用 sam 遠程調用命令

使用此命令之前，您的資源必須部署到 AWS 雲端.

使用以下命令結構並從項目的根目錄運行：

```
$ sam remote invoke <arguments> <options>
```

> **Note**
>
> 此頁面將顯示在命令提示符下提供的選項。您還可以在項目的配置文件中配置選項，而不是在命令提示符下傳遞它們。如需進一步了解，請參閱 設定專案設定。

如需引sam remote invoke數和選項的描述，請參閱sam remote invoke。

搭配 Kinesis Data Streams 使用

您可以將資料記錄傳送至 Kinesis Data Streams 應用程式。 AWS SAM CLI將發送您的數據記錄並返回分片 ID 和序列號。以下是範例：

```
$ sam remote invoke KinesisStream --stack-name kinesis-example --event hello-world

 Putting record to Kinesis data stream KinesisStream

 Auto converting value 'hello-world' into JSON '"hello-world"'. If you don't want auto-
conversion, please provide
 a JSON string as event

 {
   "ShardId": "shardId-000000000000",
   "SequenceNumber": "496462514119148067759808507900504838113011350512022323 22"
 }%
```

若要傳送資料記錄

1. 提供資源 ID 值作為 Kinesis Data Streams 應用程式的引數。如需有效資源 ID 的相關資訊，請參閱資源 ID。

2. 將資料記錄當做事件提供給 Kinesis Data Streams 應用程式。您可以使用該`--event`選項在命令行中提供事件，也可以從文件中使用`--event-file`。如果您不提供事件，則會 AWS SAM CLI 傳送空白事件。

搭配使用 Lambda 函數

您可以在雲端叫用 Lambda 函數，並傳遞空事件，或在命令列或從檔案提供事件。 AWS SAM CLI將調用您的 Lambda 函數並返回其響應。以下是範例：

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

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

若要呼叫 Lambda 函數

1. 提供資源 ID 值做為 Lambda 函數的引數。如需有效資源 ID 的相關資訊，請參閱資源 ID。
2. 提供要傳送至 Lambda 函數的事件。您可以使用該`--event`選項在命令行中提供事件，也可以從文件中使用`--event-file`。如果您不提供事件，則會 AWS SAM CLI傳送空白事件。

以回應串流設定的 Lambda 函數

此命`sam remote invoke`令支援設定為串流回應的 Lambda 函數。您可以設定 Lambda 函數，使用 AWS SAM 範本中的`FunctionUrlConfig`屬性串流回應。使用時`sam remote invoke`， AWS SAMCLI會自動偵測您的 Lambda 組態，並透過回應串流呼叫。

如需範例，請參閱叫用設定為串流回應的 Lambda 函數。

將可共用的測試事件傳遞至雲端中的 Lambda 函數

可共享的測試事件是您可以與其他人共享的測試事件。 AWS 帳戶若要深入了解，請參閱AWS Lambda開發人員指南中的可共用測試事件。

存取和管理可共用的測試事件

您可以使用命 AWS SAM CLIsam remote test-event令來存取和管理可共用的測試事件。例如，您可以使用sam remote test-event來執行下列動作：

- 從 Amazon EventBridge 架構登錄檔擷取可共用的測試事件。
- 在本機修改可共用的測試事件，並將其上傳至 EventBridge 結構描述登錄。
- 從 EventBridge 結構描述登錄中刪除可共用的測試事件。

如需進一步了解，請參閱雲端測試簡介 sam remote test-event。

將可共用的測試事件傳遞至雲端中的 Lambda 函數

若要將可共用的測試事件從 EventBridge 結構描述登錄檔傳遞至雲端中的 Lambda 函數，請使用該--test-event-name選項並提供可共用測試事件的名稱。以下是範例：

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --test-event-name demo-event
```

如果您在本機儲存可共用的測試事件，您可以使用--event-file選項並提供本機測試事件的檔案路徑和名稱。以下是範例：

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --event-file demo-event.json
```

搭配使用 與 Amazon SQS

您可以將訊息傳送到 Amazon SQS 佇列。會 AWS SAM CLI傳回下列內容：

- 訊息 ID
- 郵件內文的 MD5
- 響應元數據

以下是範例：

```
$ sam remote invoke MySqsQueue --stack-name sqs-example -event hello

Sending message to SQS queue MySqsQueue
```

```
{
  "MD5OfMessageBody": "5d41402abc4b2a76b9719d911017c592",
  "MessageId": "05c7af65-9ae8-4014-ae28-809d6d8ec652"
}%
```

傳送訊息

1. 提供資源 ID 值做為 Amazon SQS 佇列的引數。如需有效資源 ID 的相關資訊，請參閱資源 ID。
2. 提供要傳送至 Amazon SQS 佇列的事件。您可以使用該`--event`選項在命令行中提供事件，也可以從文件中使用`--event-file`。如果您不提供事件，則會 AWS SAM CLI傳送空白事件。

與 Step Functions 搭配使用

您可以調用 Step Functions 狀態機開始執行。 AWS SAM CLI將等待狀態機器工作流程完成，並傳回執行中最後一個步驟的輸出。以下是範例：

```
$ sam remote invoke HelloWorldStateMachine --stack-name state-machine-example --
event '{"is_developer": true}'

Invoking Step Function HelloWorldStateMachine

"Hello Developer World"%
```

呼叫狀態機

1. 提供資源 ID 值作為「Step Functions 數」狀態機器的引數。如需有效資源 ID 的相關資訊，請參閱資源 ID。
2. 提供要傳送至狀態機的事件。您可以使用該`--event`選項在命令行中提供事件，也可以從文件中使用`--event-file`。如果您不提供事件，則會 AWS SAM CLI傳送空白事件。

使用 sam 遠程調用命令選項

本節介紹一些您可以與`sam remote invoke`指令搭配使用的主要選項。如需選項的完整清單，請參閱sam remote invoke。

將事件傳遞至您的資源

使用下列選項將事件傳遞至雲端中的資源：

- `--event`— 在命令列傳遞事件。
- `--event-file`— 從檔案傳遞事件。

Lambda 範例

用**`--event`**於在命令行作為字符串值傳遞一個事件：

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --event '{"message":
 "hello!"}'

Invoking Lambda Function HelloWorldFunction


START RequestId: b992292d-1fac-4aa2-922a-c9dc5c6fceab Version: $LATEST
END RequestId: b992292d-1fac-4aa2-922a-c9dc5c6fceab
REPORT RequestId: b992292d-1fac-4aa2-922a-c9dc5c6fceab  Duration: 16.41 ms      Billed
 Duration: 17 ms  Memory Size: 128 MB     Max Memory Used: 67 MB  Init Duration: 185.96
 ms
{"statusCode":200,"body":"{\"message\":\"hello!\"}"}%
```

用**`--event-file`**於從文件傳遞事件並提供文件的路徑：

```
$ cat event.json

{"message": "hello from file"}%

$ sam remote invoke HelloWorldFunction --stack-name sam-app --event-file event.json


Invoking Lambda Function HelloWorldFunction


START RequestId: 3bc71f7d-153a-4b1e-8c9a-901d91b1bec9 Version: $LATEST
END RequestId: 3bc71f7d-153a-4b1e-8c9a-901d91b1bec9
REPORT RequestId: 3bc71f7d-153a-4b1e-8c9a-901d91b1bec9  Duration: 21.15 ms      Billed
 Duration: 22 ms  Memory Size: 128 MB     Max Memory Used: 67 MB
{"statusCode":200,"body":"{\"message\":\"hello from file\"}"}%
```

使用以下方式傳遞事件**`stdin`**：

```
$ cat event.json
```

```
{"message": "hello from file"}%

$ cat event.json | sam remote invoke HelloWorldFunction --stack-name sam-app --event-
file -

Reading event from stdin (you can also pass it from file with --event-file)

Invoking Lambda Function HelloWorldFunction

START RequestId: 85ecc902-8ad0-4a2b-a8c8-9bb4f65f5a7a Version: $LATEST
END RequestId: 85ecc902-8ad0-4a2b-a8c8-9bb4f65f5a7a
REPORT RequestId: 85ecc902-8ad0-4a2b-a8c8-9bb4f65f5a7a  Duration: 1.36 ms       Billed
 Duration: 2 ms   Memory Size: 128 MB       Max Memory Used: 67 MB
{"statusCode":200,"body":"{\"message\":\"hello from file\"}"}%
```

配置響 AWS SAMCLI應輸出

當您使用呼叫支援的資源時`sam remote invoke`，會 AWS SAMCLI傳回包含下列項目的回應：

- 要求中繼資料 — 與要求相關聯的中繼資料。這包括請求 ID 和請求開始時間。
- 資源回應 — 在雲端呼叫後來自資源的回應。

您可以使用該`--output`選項來配置 AWS SAM CLI輸出響應。可用的選項值如下：

- `json`— 在JSON結構中傳回中繼資料和資源回應。響應包含完整的SDK輸出。
- `text`— 以文字結構傳回中繼資料。資源回應會以資源的輸出格式傳回。

以下是`json`輸出的範例：

```
$ sam remote invoke --stack-name sam-app --output json

Invoking Lambda Function HelloWorldFunction


{
  "ResponseMetadata": {
    "RequestId": "3bdf9a30-776d-4a90-94a6-4cccc0fc7b41",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "date": "Mon, 19 Jun 2023 17:15:46 GMT",
      "content-type": "application/json",
```

```
      "content-length": "57",
      "connection": "keep-alive",
      "x-amzn-requestid": "3bdf9a30-776d-4a90-94a6-4cccc0fc7b41",
      "x-amzn-remapped-content-length": "0",
      "x-amz-executed-version": "$LATEST",
      "x-amz-log-result":
 "U1RBUlQgUmVxdWVzdElkOiAzYmRmOWEzMC03NzZkLTRhOTAtOTRhNi00Y2NjYzBmYzdiNDEgVmVyc2lvbjogJExBVEVTV
      "x-amzn-trace-id":
 "root=1-64908d42-17dab270273fcc6b527dd6b8;sampled=0;lineage=2301f8dc:0"
    },
    "RetryAttempts": 0
  },
  "StatusCode": 200,
  "LogResult":
 "U1RBUlQgUmVxdWVzdElkOiAzYmRmOWEzMC03NzZkLTRhOTAtOTRhNi00Y2NjYzBmYzdiNDEgVmVyc2lvbjogJExBVEVTV
  "ExecutedVersion": "$LATEST",
  "Payload": "{\"statusCode\":200,\"body\":\"{\\\"message\\\":\\\"hello world\\\"}\"}"
}%
```

當您指定json輸出時，整個回應會傳回至stdout。以下是範例：

```
$ sam remote invoke --stack-name sam-app --output json 1> stdout.log

Invoking Lambda Function HelloWorldFunction



$ cat stdout.log

{
  "ResponseMetadata": {
    "RequestId": "d30d280f-8188-4372-bc94-ce0f1603b6bb",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "date": "Mon, 19 Jun 2023 17:35:56 GMT",
      "content-type": "application/json",
      "content-length": "57",
      "connection": "keep-alive",
      "x-amzn-requestid": "d30d280f-8188-4372-bc94-ce0f1603b6bb",
      "x-amzn-remapped-content-length": "0",
      "x-amz-executed-version": "$LATEST",
      "x-amz-log-result":
 "U1RBUlQgUmVxdWVzdElkOiBkMzBkMjgwZi04MTg4LTQzNzItYmM5NC1jZTBmMTYwM2I2YmIgVmVyc2lvbjogJExBVEVTV
```

```
      "x-amzn-trace-id":
 "root=1-649091fc-771473c7778689627a6122b7;sampled=0;lineage=2301f8dc:0"
    },
    "RetryAttempts": 0
  },
  "StatusCode": 200,
  "LogResult":
 "U1RBUlQgUmVxdWVzdElkOiBkMzBkMjgwZi04MTg4LTQzNzItYmM5NC1jZTBmMTYwM2I2YmIgVmVyc2lvbjogJExBVEVTV
  "ExecutedVersion": "$LATEST",
  "Payload": "{\"statusCode\":200,\"body\":\"{\\\"message\\\":\\\"hello world\\\"}\"}"
}%
```

以下是text輸出的範例：

```
$ sam remote invoke --stack-name sam-app --output text

Invoking Lambda Function HelloWorldFunction


START RequestId: 4dbacc43-1ec6-47c2-982b-9dc4620144d6 Version: $LATEST
END RequestId: 4dbacc43-1ec6-47c2-982b-9dc4620144d6
REPORT RequestId: 4dbacc43-1ec6-47c2-982b-9dc4620144d6  Duration: 9.13 ms       Billed
 Duration: 10 ms  Memory Size: 128 MB     Max Memory Used: 67 MB  Init Duration: 165.50
 ms
{"statusCode":200,"body":"{\"message\":\"hello world\"}"}%
```

當您指定text輸出時，Lambda 函數執行階段輸出 (例如記錄檔) 會傳回至stderr。Lambda 函數有效負載會傳回至stdout。以下是範例：

```
$ sam remote invoke --stack-name sam-app --output text 2> stderr.log

{"statusCode":200,"body":"{\"message\":\"hello world\"}"}%



$ cat stderr.log

Invoking Lambda Function HelloWorldFunction
START RequestId: 82273c3b-aa3a-4d16-8f1c-1d2ad3ace891 Version: $LATEST
END RequestId: 82273c3b-aa3a-4d16-8f1c-1d2ad3ace891
REPORT RequestId: 82273c3b-aa3a-4d16-8f1c-1d2ad3ace891  Duration: 40.62 ms      Billed
 Duration: 41 ms  Memory Size: 128 MB     Max Memory Used: 68 MB
```

```
$ sam remote invoke --stack-name sam-app --output text 1> stdout.log

Invoking Lambda Function HelloWorldFunction


START RequestId: 74acaa9f-5b80-4a5c-b3b8-ffaccb84cbbd Version: $LATEST
END RequestId: 74acaa9f-5b80-4a5c-b3b8-ffaccb84cbbd
REPORT RequestId: 74acaa9f-5b80-4a5c-b3b8-ffaccb84cbbd  Duration: 2.31 ms       Billed
 Duration: 3 ms   Memory Size: 128 MB     Max Memory Used: 67 MB

$ cat stdout.log

{"statusCode":200,"body":"{\"message\":\"hello world\"}"}%
```

自訂Boto3參數

對於`sam remote invoke`，AWS SAM CLI利用適用 AWS SDK for Python (Boto3) 與您在雲中的資源進行交互。您可以使用`--parameter`此選項來自訂Boto3參數。如需可自訂的支援參數清單，請參閱`--parameter`。

範例

叫用 Lambda 函數來驗證參數值並驗證權限：

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --
parameter InvocationType="DryRun"
```

在單一命令中多次使用**`--parameter`**此選項可提供多個參數：

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --
parameter InvocationType="Event" --parameter LogType="None"
```

其他選項

如需選`sam remote invoke`項的完整清單，請參閱sam remote invoke。

配置您的項目配置文件

若要`sam remote invoke`在組態檔案中進行配置，請`remote_invoke`在表格中使用。以下是設定指令預設值的`samconfig.toml`檔案範例。`sam remote invoke`

```
...
version =0.1

[default]
...
[default.remote_invoke.parameters]
stack_name = "cloud-app"
event = '{"message": "Hello!"}'
```

範例

如需使用的基本範例`sam remote invoke`，請參閱 AWS Compute 部落格中的使用 AWS SAM 遠端測試 AWS Lambda 函數。

Kinesis Data Streams 範例

基本範例

將資料記錄從檔案傳送至 Kinesis Data Streams 應用程式。Kinesis 資料串流應用程式可透過為資源識別碼提供 ARN 來識別：

```
$ sam remote invoke arn:aws:kinesis:us-west-2:01234567890:stream/kinesis-example-KinesisStream-BgnLcAey4xUQ --event-file event.json
```

將命令列中提供的事件傳送至 Kinesis Data Streams 應用程式：

```
$ sam remote invoke KinesisStream --stack-name kinesis-example --event hello-world

Putting record to Kinesis data stream KinesisStream

Auto converting value 'hello-world' into JSON '"hello-world"'. If you don't want auto-conversion, please provide
a JSON string as event

{
  "ShardId": "shardId-000000000000",
  "SequenceNumber": "49646251411914806775980903986194508740483329854174920706"
}%
```

取得 Kinesis Data Streams 應用程式的實體 ID。然後，在命令行中提供一個事件：

```
$ sam list resources --stack-name kinesis-example --output json

[
  {
    "LogicalResourceId": "KinesisStream",
    "PhysicalResourceId": "kinesis-example-KinesisStream-ZgnLcQey4xUQ"
  }
]

$ sam remote invoke kinesis-example-KinesisStream-ZgnLcQey4xUQ --event hello

Putting record to Kinesis data stream KinesisStream

Auto converting value 'hello' into JSON '"hello"'. If you don't want auto-conversion,
 please provide a JSON
string as event

{
  "ShardId": "shardId-000000000000",
  "SequenceNumber": "49646251411914806775980904340716841045751814812900261890"
}%
```

在命令行中提供 JSON 字符串作為事件：

```
$ sam remote invoke KinesisStream --stack-name kinesis-example --event '{"method":
 "GET", "body": ""}'

Putting record to Kinesis data stream KinesisStream

{
  "ShardId": "shardId-000000000000",
  "SequenceNumber": "49646251411914806775980904492868617924990209230536441858"
}%
```

傳送空白事件至 Kinesis Data Streams 應用程式：

```
$ sam remote invoke KinesisStream --stack-name kinesis-example

Putting record to Kinesis data stream KinesisStream

{
  "ShardId": "shardId-000000000000",
```

```
  "SequenceNumber": "49646251411914806775980904866469008589597168190416224258"
}%
```

返回 JSON 格式的 AWS SAM CLI響應：

```
$ sam remote invoke KinesisStream --stack-name kinesis-example --event '{"hello":
 "world"}' --output json

Putting record to Kinesis data stream KinesisStream

{
  "ShardId": "shardId-000000000000",
  "SequenceNumber": "49646251411914806775980905078409420803696667195489648642",
  "ResponseMetadata": {
    "RequestId": "ebbbd307-3e9f-4431-b67c-f0715e9e353e",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "x-amzn-requestid": "ebbbd307-3e9f-4431-b67c-f0715e9e353e",
      "x-amz-id-2": "Q3yBcgTwtPaQTV26IKclbECmZikUYOzKY+CzcxA84ZHgCkc5T2N/
ITWg6RPOQcWw8Gn0tNPcEJBEHyVVqboJAPgCritqsvCu",
      "date": "Thu, 09 Nov 2023 18:13:10 GMT",
      "content-type": "application/x-amz-json-1.1",
      "content-length": "110"
    },
    "RetryAttempts": 0
  }
}%
```

返回 JSON 輸出到標準輸出：

```
$ sam remote invoke KinesisStream --stack-name kinesis-example --event '{"hello":
 "world"}' --output json 1> stdout.log

Putting record to Kinesis data stream KinesisStream


$ cat stdout.log
{
  "ShardId": "shardId-000000000000",
  "SequenceNumber": "49646251411914806775980906397777867595039988349006774274",
  "ResponseMetadata": {
    "RequestId": "f4290006-d84b-b1cd-a9ee-28306eeb2939",
    "HTTPStatusCode": 200,
```

```
    "HTTPHeaders": {
      "x-amzn-requestid": "f4290006-d84b-b1cd-a9ee-28306eeb2939",
      "x-amz-id-2": "npCqz
+IBKpoL4sQ1ClbUmxuJlbeA24Fx1UgpIrS6mm2NoIeV2qdZSN5AhNurdssykXajBrXaC9anMhj2eG/h7Hnbf
+bPuotU",
      "date": "Thu, 09 Nov 2023 18:33:26 GMT",
      "content-type": "application/x-amz-json-1.1",
      "content-length": "110"
    },
    "RetryAttempts": 0
  }
}%
```

Lambda 範例

基本範例

藉由提供 ARN 做為資源識別碼來叫用 Lambda 函數：

```
$ sam remote invoke arn:aws:lambda:us-west-2:012345678910:function:sam-app-
HelloWorldFunction-ohRFEn2RuAvp
```

藉由提供邏輯 ID 做為資源識別碼來叫用 Lambda 函數：

您也必須使用`--stack-name`選項提供 AWS CloudFormation 堆疊名稱。以下是範例：

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

如果您的應用程式包含單一 Lambda 函數，則不需要指定它的邏輯 ID。您只能提供`--stack-name`選項。以下是範例：

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

透過提供實體 ID 做為資源識別碼來叫用 Lambda 函數：

當您使用部署時，會建立實體 ID AWS CloudFormation。

```
$ sam remote invoke sam-app-HelloWorldFunction-TZvxQRFNv0k4
```

調用子堆棧的 Lambda 函數：

在此示例中，我們的應用程序包含以下目錄結構：

```
lambda-example
### childstack
#   ### function
#   #   ### __init__.py
#   #   ### app.py
#   #   ### requirements.txt
#   ### template.yaml
### events
#   ### event.json
### samconfig.toml
### template.yaml
```

若要叫用我們的 Lambda 函數childstack，我們執行下列命令：

```
$ sam remote invoke ChildStack/HelloWorldFunction --stack-name lambda-example

Invoking Lambda Function HelloWorldFunction

START RequestId: 207a864b-e67c-4307-8478-365b004d4bcd Version: $LATEST
END RequestId: 207a864b-e67c-4307-8478-365b004d4bcd
REPORT RequestId: 207a864b-e67c-4307-8478-365b004d4bcd  Duration: 1.27 ms       Billed
 Duration: 2 ms   Memory Size: 128 MB     Max Memory Used: 36 MB  Init Duration: 111.07
 ms
{"statusCode": 200, "body": "{\"message\": \"Hello\", \"received_event\": {}}"}%
```

叫用設定為串流回應的 Lambda 函數

在此範例中，我們使用初始化新的 AWS SAMCLI無伺服器應用程式，該應用程式包含設定為串流回應的 Lambda 函數。我們將應用程式部署到， AWS 雲端 並使用sam remote invoke與我們在雲端中的功能互動。

我們首先執行sam init命令來建立新的無伺服器應用程式。我們選取 Lambda 回應串流快速入門範本並命名我們的應用程式lambda-streaming-nodejs-app。

```
$ sam init

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

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

```
        2 - Custom Template Location
 Choice: 1

 Choose an AWS Quick Start application template
        1 - Hello World Example
        ...
        9 - Lambda Response Streaming
        ...
        15 - Machine Learning
 Template: 9

 Which runtime would you like to use?
        1 - go (provided.al2)
        2 - nodejs18.x
        3 - nodejs16.x
 Runtime: 2

 Based on your selections, the only Package type available is Zip.
 We will proceed to selecting the Package type as Zip.

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

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

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

 Project name [sam-app]: lambda-streaming-nodejs-app

    -----------------------
    Generating application:
    -----------------------
    Name: lambda-streaming-nodejs-app
    Runtime: nodejs18.x
    Architectures: x86_64
    Dependency Manager: npm
    Application Template: response-streaming
    Output Directory: .
    Configuration file: lambda-streaming-nodejs-app/samconfig.toml
```

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


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

該 AWS SAMCLI創建了我們的項目具有以下結構：

```
lambda-streaming-nodejs-app
 ### README.md
 ### __tests__
 #   ### unit
 #       ### index.test.js
 ### package.json
 ### samconfig.toml
 ### src
 #   ### index.js
 ### template.yaml
```

以下是我們 Lambda 函數程式碼的範例：

```
exports.handler = awslambda.streamifyResponse(
   async (event, responseStream, context) => {
     const httpResponseMetadata = {
       statusCode: 200,
       headers: {
         "Content-Type": "text/html",
         "X-Custom-Header": "Example-Custom-Header"
       }
     };

     responseStream = awslambda.HttpResponseStream.from(responseStream,
 httpResponseMetadata);
     // It's recommended to use a `pipeline` over the `write` method for more complex
 use cases.
     // Learn more: https://docs.aws.amazon.com/lambda/latest/dg/configuration-
response-streaming.html
     responseStream.write("<html>");
```

```
    responseStream.write("<p>First write!</p>");

    responseStream.write("<h1>Streaming h1</h1>");
    await new Promise(r => setTimeout(r, 1000));
    responseStream.write("<h2>Streaming h2</h2>");
    await new Promise(r => setTimeout(r, 1000));
    responseStream.write("<h3>Streaming h3</h3>");
    await new Promise(r => setTimeout(r, 1000));

    // Long strings will be streamed
    const loremIpsum1 = "Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Quisque vitae mi tincidunt tellus ultricies dignissim id et diam. Morbi pharetra eu
nisi et finibus. Vivamus diam nulla, vulputate et nisl cursus, pellentesque vehicula
libero. Cras imperdiet lorem ante, non posuere dolor sollicitudin a. Vestibulum ipsum
lacus, blandit nec augue id, lobortis dictum urna. Vestibulum ante ipsum primis in
faucibus orci luctus et ultrices posuere cubilia curae; Morbi auctor orci eget tellus
aliquam, non maximus massa porta. In diam ante, pulvinar aliquam nisl non, elementum
hendrerit sapien. Vestibulum massa nunc, mattis non congue vitae, placerat in quam.
Nam vulputate lectus metus, et dignissim erat varius a.";
    responseStream.write(`<p>${loremIpsum1}</p>`);
    await new Promise(r => setTimeout(r, 1000));

    responseStream.write("<p>DONE!</p>");
    responseStream.write("</html>");
    responseStream.end();
  }
);
```

以下是我們的`template.yaml`文件的一個例子。Lambda 函數的回應串流是使用`FunctionUrlConfig`屬性設定的。

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

 Description: >
   Sample SAM Template for lambda-streaming-nodejs-app

 Resources:
   StreamingFunction:
     Type: AWS::Serverless::Function
     Properties:
       CodeUri: src/
       Handler: index.handler
```

```
      Runtime: nodejs18.x
      Architectures:
        - x86_64
      Timeout: 10
      FunctionUrlConfig:
        AuthType: AWS_IAM
        InvokeMode: RESPONSE_STREAM

Outputs:
  StreamingFunction:
    Description: "Streaming Lambda Function ARN"
    Value: !GetAtt StreamingFunction.Arn
  StreamingFunctionURL:
    Description: "Streaming Lambda Function URL"
    Value: !GetAtt StreamingFunctionUrl.FunctionUrl
```

通常，您可以使用sam build和sam deploy --guided建置和部署生產應用程式。在這個例子中，我們將假設一個開發環境，並使用命sam sync令來構建和部署我們的應用程序。

> **Note**
>
> 建議在開發環境中使用此sam sync命令。如需進一步了解，請參閱使用同步sam sync到簡介 AWS 雲端。

在運行之前sam sync，我們驗證我們的項目在我們的samconfig.toml文件中配置正確。最重要的是，我們會驗證stack_name和的值watch。在我們的配置文件中指定了這些值，我們不必在命令行中提供它們。

```
version = 0.1

[default]
[default.global.parameters]
stack_name = "lambda-streaming-nodejs-app"

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

[default.validate.parameters]
lint = true
```

```
[default.deploy.parameters]
capabilities = "CAPABILITY_IAM"
confirm_changeset = true
resolve_s3 = true
s3_prefix = "lambda-streaming-nodejs-app"
region = "us-west-2"
image_repositories = []

[default.package.parameters]
resolve_s3 = true

[default.sync.parameters]
watch = true

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

[default.local_start_lambda.parameters]
warm_containers = "EAGER"
```

接下來，我們執行sam sync建置和部署我們的應用程式。由於該--watch選項是在我們的配置文件中配置的， AWS SAMCLI將構建我們的應用程序，部署我們的應用程序，並監視更改。

```
$ sam sync

The SAM CLI will use the AWS Lambda, Amazon API Gateway, and AWS StepFunctions APIs to
upload your code
without

performing a CloudFormation deployment. This will cause drift in your CloudFormation
stack.
**The sync command should only be used against a development stack**.



Queued infra sync. Waiting for in progress code syncs to complete...

Starting infra sync.

Building codeuri:

/Users/.../lambda-streaming-nodejs-app/src runtime: nodejs18.x metadata: {}
architecture: x86_64 functions: StreamingFunction
```

```
 package.json file not found. Continuing the build without dependencies.

 Running NodejsNpmBuilder:CopySource


 Build Succeeded

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


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

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


 2023-06-20 12:11:16 - Waiting for stack create/update to complete

 CloudFormation events from stack operations (refresh every 0.5 seconds)
 ---------------------------------------------------------------------------------------------
 ResourceStatus            ResourceType              LogicalResourceId
 ResourceStatusReason
 ---------------------------------------------------------------------------------------------
 CREATE_IN_PROGRESS        AWS::CloudFormation::St   lambda-streaming-
 Transformation
                           ack                       nodejs-app
 succeeded
 CREATE_IN_PROGRESS        AWS::IAM::Role            StreamingFunctionRole     -
```

```
CREATE_IN_PROGRESS        AWS::CloudFormation::St   AwsSamAutoDependencyLay   -
                          ack                       erNestedStack
CREATE_IN_PROGRESS        AWS::IAM::Role            StreamingFunctionRole     Resource
creation
Initiated
CREATE_IN_PROGRESS        AWS::CloudFormation::St   AwsSamAutoDependencyLay   Resource
creation
                          ack                       erNestedStack
Initiated
CREATE_COMPLETE           AWS::IAM::Role            StreamingFunctionRole     -
CREATE_COMPLETE           AWS::CloudFormation::St   AwsSamAutoDependencyLay   -
                          ack                       erNestedStack
CREATE_IN_PROGRESS        AWS::Lambda::Function     StreamingFunction         -
CREATE_IN_PROGRESS        AWS::Lambda::Function     StreamingFunction         Resource
creation
Initiated
CREATE_COMPLETE           AWS::Lambda::Function     StreamingFunction         -
CREATE_IN_PROGRESS        AWS::Lambda::Url          StreamingFunctionUrl      -
CREATE_IN_PROGRESS        AWS::Lambda::Url          StreamingFunctionUrl      Resource
creation
Initiated
CREATE_COMPLETE           AWS::Lambda::Url          StreamingFunctionUrl      -
CREATE_COMPLETE           AWS::CloudFormation::St   lambda-streaming-         -
                          ack                       nodejs-app
-----------------------------------------------------------------------------------------
CloudFormation outputs from deployed stack
-----------------------------------------------------------------------------------------
Outputs
```

```
 ---------------------------------------------------------------------------------------------------
 Key                 StreamingFunction

 Description         Streaming Lambda Function ARN

 Value               arn:aws:lambda:us-west-2:012345678910:function:lambda-streaming-
nodejs-app-
 StreamingFunction-gUmhO833A0vZ


 Key                 StreamingFunctionURL

 Description         Streaming Lambda Function URL

 Value               https://wxgkcc2dyntgtrwhf2dgdcvylu0rnnof.lambda-url.us-
west-2.on.aws/
 ---------------------------------------------------------------------------------------------------



 Stack creation succeeded. Sync infra completed.



 Infra sync completed.
```

現在，我們的功能已部署到雲端，我們可以用sam remote invoke來與我們的功能進行交互。 AWS SAMCLI自動檢測我們的功能是否已配置為響應流，並立即開始實時輸出我們函數的流式響應。

```
$ sam remote invoke StreamingFunction

 Invoking Lambda Function StreamingFunction

 {"statusCode":200,"headers":{"Content-Type":"text/html","X-Custom-Header":"Example-
Custom-Header"}}<html><p>First write!</p><h1>Streaming h1</h1><h2>Streaming h2</
h2><h3>Streaming h3</h3><p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
 Quisque vitae mi tincidunt tellus ultricies dignissim id et diam. Morbi pharetra eu
 nisi et finibus. Vivamus diam nulla, vulputate et nisl cursus, pellentesque vehicula
 libero. Cras imperdiet lorem ante, non posuere dolor sollicitudin a. Vestibulum ipsum
 lacus, blandit nec augue id, lobortis dictum urna. Vestibulum ante ipsum primis in
 faucibus orci luctus et ultrices posuere cubilia curae; Morbi auctor orci eget tellus
 aliquam, non maximus massa porta. In diam ante, pulvinar aliquam nisl non, elementum
 hendrerit sapien. Vestibulum massa nunc, mattis non congue vitae, placerat in quam.
```

```
 Nam vulputate lectus metus, et dignissim erat varius a.</p><p>DONE!</p></html>START
 RequestId: 1e4cdf04-60de-4769-b3a2-c1481982deb4 Version: $LATEST
 END RequestId: 1e4cdf04-60de-4769-b3a2-c1481982deb4
 REPORT RequestId: 1e4cdf04-60de-4769-b3a2-c1481982deb4  Duration: 4088.66 ms
 Billed Duration: 4089 ms        Memory Size: 128 MB     Max Memory Used: 68 MB  Init
 Duration: 168.45 ms
```

當我們修改函數代碼時，立 AWS SAMCLI即檢測並立即部署我們的更改。以下是對函數代碼進行更改後 AWS SAMCLI輸出的示例：

```
Syncing Lambda Function StreamingFunction...

 Building codeuri:

 /Users/.../lambda-streaming-nodejs-app/src runtime: nodejs18.x metadata: {}
 architecture:
 x86_64 functions: StreamingFunction

 package.json file not found. Continuing the build without dependencies.

 Running NodejsNpmBuilder:CopySource

 Finished syncing Lambda Function StreamingFunction.

 Syncing Layer StreamingFunctione9cfe924DepLayer...

 SyncFlow [Layer StreamingFunctione9cfe924DepLayer]: Skipping resource update as the

 content didn't change

 Finished syncing Layer StreamingFunctione9cfe924DepLayer.
```

現在，我們可以`sam remote invoke`再次使用與雲中的功能進行交互並測試我們的更改。

SQS 範例

基本範例

提供 ARN 做為資源識別碼，以叫用 Amazon SQS 佇列：

```
$ sam remote invoke arn:aws:sqs:us-west-2:01234567890:sqs-example-4DonhBsjsW1b --
event '{"hello": "world"}' --output json
```

```
Sending message to SQS queue MySqsQueue

{
  "MD5OfMessageBody": "49dfdd54b01cbcd2d2ab5e9e5ee6b9b9",
  "MessageId": "4f464cdd-15ef-4b57-bd72-3ad225d80adc",
  "ResponseMetadata": {
    "RequestId": "95d39377-8323-5ef0-9223-ceb198bd09bd",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "x-amzn-requestid": "95d39377-8323-5ef0-9223-ceb198bd09bd",
      "date": "Wed, 08 Nov 2023 23:27:26 GMT",
      "content-type": "application/x-amz-json-1.0",
      "content-length": "106",
      "connection": "keep-alive"
    },
    "RetryAttempts": 0
  }
}%
```

Step Functions 範例

基本範例

藉由提供其實體 ID 作為資源 ID 來叫用狀態機器：

首先，我們用`sam list resources`來獲取我們的物理 ID：

```
$ sam list resources --stack-name state-machine-example --output json

[
  {
    "LogicalResourceId": "HelloWorldStateMachine",
    "PhysicalResourceId": "arn:aws:states:us-
west-2:513423067560:stateMachine:HelloWorldStateMachine-z69tFEUx0F66"
  },
  {
    "LogicalResourceId": "HelloWorldStateMachineRole",
    "PhysicalResourceId": "simple-state-machine-HelloWorldStateMachineRole-
PduA0BDGuFXw"
  }
]
```

接下來，我們使用物理 ID 作為資源 ID 調用我們的狀態機。我們通過在命令行與--event選項的事件：

```
$ sam remote invoke arn:aws:states:us-
west-2:01234567890:stateMachine:HelloWorldStateMachine-z69tFEUx0F66 --
event '{"is_developer": true}'

Invoking Step Function arn:aws:states:us-
west-2:01234567890:stateMachine:HelloWorldStateMachine-z69tFEUx0F66
"Hello Developer World"%
```

通過傳遞一個空事件調用狀態機：

```
$ sam remote invoke HelloWorldStateMachine --stack-name state-machine-example

Invoking Step Function HelloWorldStateMachine

"Hello World"%
```

相關連結

如需與sam remote invoke和使用相關的說明文件 AWS SAMCLI，請參閱下列內容：

- sam remote invoke
- AWS SAMCLI疑難排

自動執行本機整合測試 AWS SAM

雖然您可以用測試簡介 sam local invoke來手動測試程式碼，但 AWS SAM 也可讓您使用自動化整合測試來測試程式碼。整合測試可協助您在開發週期的早期偵測問題、改善程式碼品質，並節省時間，同時降低成本。

若要在中編寫自動化整合測試 AWS SAM，您必須先對本機 Lambda 函數執行測試，然後再部署到 AWS 雲端。命測試簡介 sam local start-lambda令會啟動模擬 Lambda 叫用端點的本機端點。您可以從自動化測試中調用它。由於此端點會模擬 Lambda 叫用端點，因此您可以撰寫一次測試，然後針對本機 Lambda 函數或已部署的 Lambda 函數執行測試 (不進行任何修改)。您也可以針對 CI/CD 管道中已部署的 AWS SAM 堆疊執行相同的測試。

這是該過程的工作原理：

1. 啟動本機 Lambda 端點。

   在包含 AWS SAM 範本的目錄中執行下列命令，以啟動本機 Lambda 端點：

   ```
   sam local start-lambda
   ```

   此命令會在該端點處啟動模擬`http://127.0.0.1:3001` AWS Lambda的本機端點。您可以對這個本機 Lambda 端點執行自動化測試。當您使用 AWS CLI 或 SDK 叫用此端點時，它會在本機執行要求中指定的 Lambda 函數，並傳回回應。

2. 針對本機 Lambda 端點執行整合測試。

   在整合測試中，您可以使用 AWS SDK 來叫用含有測試資料的 Lambda 函數、等待回應，以及確認回應是否符合您的預期。若要在本機執行整合測試，您應該將 AWS SDK 設定為傳送 Lambda 叫用 API 呼叫，以叫用您在上一個步驟中啟動的本機 Lambda 端點。

   以下是 Python 示例（其他語言的 AWS SDK 具有類似的配置）：

   ```
   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=15,
               retries={'max_attempts': 0},
           )
       )
   else:
       lambda_client = boto3.client('lambda')
   ```

```
# Invoke your Lambda function as you normally usually do. The function will run
# locally if it is configured to do so
response = lambda_client.invoke(FunctionName="HelloWorldFunction")

# Verify the response
assert response == "Hello World"
```

您可以將設`running_locally`定為，使用此程式碼來測試已部署的 Lambda 函數`False`。這會將 AWS SDK 設定為在 AWS 雲端 AWS Lambda 中連線。

產生範例事件承載

若要測試 Lambda 函數，您可以產生和自訂範例事件承載，以模仿 Lambda 函數在其他 AWS 服務觸發時將接收的資料。這包括 API Gateway AWS CloudFormation、Amazon S3 等服務。

產生範例事件承載可協助您使用各種不同的輸入來測試 Lambda 函數的行為，而不需要在即時環境中工作。與手動建立 AWS 服務事件範例以測試函數相比，此方法也可以節省時間。

如需可產生範例事件承載之服務的完整清單，請使用以下命令：

```
sam local generate-event --help
```

對於可用於特定服務的選項列表，請使用以下命令：

```
sam local generate-event [SERVICE] --help
```

範例：

```
#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
```

偵錯您的無伺服器應用程式 AWS SAM

測試您的應用程式之後，您就可以準備偵錯您發現的任何問題。使用命 AWS SAM 令列介面 (CLI)，您可以在本機測試和偵錯無伺服器應用程式，然後再將其上傳到 AWS 雲端。偵錯應用程式可識別並修正應用程式中的問題或錯誤。

您可以使用 AWS SAM 來執行逐步偵錯，這是一次執行一行或指令的程式碼的方法。當您以偵錯模式在本機叫用 Lambda 函數時 AWS SAMCLI，您可以接著將除錯程式附加至該函數。使用偵錯工具，您可以逐行執行程式碼、查看不同變數的值，以及修正問題的方式，與其他任何應用程式相同的方式。在執行封裝和部署應用程式的步驟之前，您可以驗證應用程式是否如預期般運作、偵錯錯誤並修正任何問題。

> **Note**
>
> 如果您的應用程式包含一或多個層，則當您在本機執行和偵錯應用程式時，Layer 套件會下載並快取到您的本機主機上。如需更多詳細資訊，請參閱 如何在本機快取圖層。

主題

本地調試功能 AWS SAM

您可以 AWS SAM 搭配各種 AWS 工具組和除錯器使用，在本機測試和偵錯無伺服器應用程式。Lambda 函數的逐步偵錯可讓您在本機環境中一次識別並修正應用程式中的問題。

執行本機逐步偵錯的一些方法包括設定中斷點、檢查變數，以及一次執行一行函式程式碼。本機逐步偵錯可以讓您尋找並疑難排解可能在雲端中遇到的問題，藉此縮小回饋迴圈。

您可以使用 AWS 工具包進行調試，也可以 AWS SAM 在調試模式下運行。如需詳細資訊，請參閱本節中的主題。

使用 AWS 工具組

AWS Toolkit 是整合式開發環境 (IDE) 外掛程式，可讓您一次執行許多常見的偵錯工作，例如設定中斷點、檢查變數，以及一行執行函數程式碼。 AWS Toolkit 可讓您更輕鬆地開發、偵錯和部署使用. AWS SAM它們提供建置、測試、偵錯、部署和叫用已整合至 IDE 的 Lambda 函數的體驗。

如需可搭配使用之 AWS 「工具組」的詳細資訊 AWS SAM，請參閱下列內容：

- AWS Toolkit for Visual Studio Code
- AWS Cloud9
- AWS Toolkit for JetBrains

有各種各樣的工 AWS 具包可以與 IDE 和運行時的不同組合工作。下表列出支援應用程式逐步偵錯的常見 IDE/ 執行階段組合： AWS SAM

IDE	執行期	AWS 工具包	逐步偵錯的指示
Visual Studio 程式碼	• Node.js • Python • .NET • Java • Go	AWS Toolkit for Visual Studio Code	在《AWS Toolkit for Visual Studio Code 使用者指南》 AWS 無伺服器應用程式中使用
AWS Cloud9	• Node.js • Python	AWS Cloud9，啟用 AWS 工具包 [1]	使用使用AWS Cloud9者指南中的工 AWS 具組來處理 AWS 無伺服器應用程式。
WebStorm	Node.js	AWS Toolkit for JetBrains[2]	執行 (叫用) 或偵錯 AWS Toolkit for JetBrains
PyCharm	Python	AWS Toolkit for JetBrains[2]	執行 (叫用) 或偵錯 AWS Toolkit for JetBrains

IDE	執行期	AWS 工具包	逐步偵錯的指示
Rider	.NET	AWS Toolkit for JetBrains[2]	執行 (叫用) 或偵錯 AWS Toolkit for JetBrains
IntelliJ	Java	AWS Toolkit for JetBrains[2]	執行 (叫用) 或偵錯 AWS Toolkit for JetBrains
GoLand	Go	AWS Toolkit for JetBrains[2]	執行 (叫用) 或偵錯 AWS Toolkit for JetBrains

備註：

1. 若 AWS Cloud9 要使用逐步偵錯 AWS SAM 應用程式，必須啟用「 AWS 工具組」。如需詳細資訊，請參閱《使用指南》中的〈啟AWS Cloud9 用 AWS 工具組〉。
2. 若要使用 AWS Toolkit for JetBrains 逐步執行除錯 AWS SAM 應用程式，您必須先按照安裝中的指示進行安裝和設定。 AWS Toolkit for JetBrainsAWS Toolkit for JetBrains

以偵錯模式在 AWS SAM 本機執行

除了與 T AWS oolkit 集成之外，您還可以 AWS SAM 在「調試模式」下運行以附加到第三方調試器（如 pt vsd ）或進行深入研究。

若要 AWS SAM 在除錯模式下執行，請使用指令sam local invoke或sam local start-api搭配`--debug-port`或`-d`選項。

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**
>
> 如果您使用的是`sam local start-api`，本機 API Gateway 執行個體會公開您所有的 Lambda 函數。但是，由於您可以指定單一偵錯連接埠，因此一次只能偵錯一個函數。您需要在 AWS SAMCLI綁定到端口之前調用 API，這允許調試器連接。

調試時傳遞多個運行時參數 AWS SAM

您可以選擇通過其他運行時參數 AWS SAM 來檢查問題並更有效地對變量進行故障排除。這樣做可為您的除錯程序提供額外的控制與彈性，協助您自訂執行階段設定和環境。

若要在偵錯函數時傳遞其他執行階段引數，請使用環境變數DEBUGGER_ARGS。這將一串引數直接傳遞到 run 命令中， AWS SAMCLI用於啟動函數。

例如，如果您想在 Python 函數的運行時加載像 iKPDB 這樣的調試器，則可以通過以下方式傳遞以下內容。`DEBUGGER_ARGS: -m ikpdb --ikpdb-port=5858 --ikpdb-working-directory=/var/task/ --ikpdb-client-working-directory=/myApp --ikpdb-address=0.0.0.0`這將在運行時使用您指定的其他參數加載 iKPDB。

在這種情況下，您的完整 AWS SAMCLI命令將是：

```
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
```

您可以將調試器參數傳遞給所有運行時的函數。

使用 AWS CloudFormation Linter 驗證您的 AWS SAM 應用程式

AWS CloudFormation 林特（cfn-lint）是一種開源工具，可用於對模板執行詳細驗證。 AWS CloudFormation CFN-lint 包含由資源規範引導的規則 AWS CloudFormation 。使用 cfn-lint 將您的資源與這些規則進行比較，以接收有關錯誤、警告或資訊性建議的詳細訊息。或者，建立您自己的自訂規則以進行驗證。要了解有關 cfn-lint 的更多信息，請參閱存儲庫中的 cfn- lint。AWS CloudFormation GitHub

您可以使用 cfn-lint 透過 AWS SAM 指令行介面 AWS Serverless Application Model (AWS SAM) 驗證您的 () 範本，方法是使sam validate用選項執行。AWS SAMCLI --lint

```
sam validate --lint
```

若要自訂 cfn-lint 行為 (例如建立自訂規則或指定驗證選項)，您可以定義規劃檔。要了解更多信息，請參閱 cfn-lint AWS CloudFormation GitHub 存儲庫中的 Config 文件。運行時sam validate --lint，將應用配置文件中定義的 cfn-lint 行為。

範例

在模板上執行 cfn-lint 驗證 AWS SAM

```
sam validate --lint --template myTemplate.yaml
```

進一步了解

欲進一步了解 sam validate 命令，請參閱 sam validate。

部署您的應用程式和資源 AWS SAM

在雲端部署您的應用程式佈建和設定您的 AWS 資源，讓您的應用程式在 AWS 雲端中執行。 AWS SAM 用AWS CloudFormation作其基礎部署機制。 AWS SAM 使用您在執行sam build命令時建立的組建成品，作為部署無伺服器應用程式的標準輸入。

您可以使 AWS SAM用手動部署無伺服器應用程式，也可以自動化部署。若要自動化部署，您可以使用 AWS SAM 管道搭配您選擇的持續整合和持續部署 (CI/CD) 系統。您的部署管道是一系列自動化的步驟，可執行以發行新版本的無伺服器應用程式。

本節中的主題提供有關自動和手動部署的指導。若要手動部署應用程式，請使用 AWS SAMCLI指令。若要自動化部署，請參閱本節中的主題。他們特別提供有關使用管道和 CI/CD 系統自動化部署的深入內容。這包括產生入門管道、設定自動化、疑難排解部署、使用 OpenID Connect (OIDC) 使用者驗證，以及在部署時上傳本機檔案。

主題

- 使用sam deploy指令部署簡介
- 用於部署應用程式的選項 AWS SAM
- 使用 CI/CD 系統和管道進行部署 AWS SAM
- 使用同步sam sync到簡介 AWS 雲端

使用sam deploy指令部署簡介

使用指 AWS Serverless Application Model 令行介面 (AWS SAMCLI) `sam deploy` 指令將無伺服器應用程式部署到 AWS 雲端.

- 如需「」的簡介 AWS SAMCLI，請參閱什麼是 AWS SAMCLI？。
- 如需指`sam deploy`令選項的清單，請參閱sam deploy。
- 如需在典型開發工作流程`sam deploy`期間使用的範例，請參閱步驟 3：將您的應用程式部署到 AWS 雲端。

主題

- 必要條件
- 使用 sam 部署部署應用程式

- 最佳實務
- 山姆部署的選項
- 故障診斷
- 範例
- 進一步了解

必要條件

若要使用sam deploy，請完成下列 AWS SAMCLI步驟來安裝：

- AWS SAM 前提.
- 安裝 AWS SAMCLI.

使用前sam deploy，我們建議對以下內容進行基本了解：

- 配置 AWS SAMCLI.
- 使用sam init指令建立您的應用程式.
- 使用sam build指令建立簡介.

使用 sam 部署部署應用程式

第一次部署無伺服器應用程式時，請使用選--guided項。 AWS SAMCLI將引導您完成互動式流程，以配置應用程式的部署設定。

使用互動式流程部署應用程式

1. 轉到項目的根目錄。這與您的 AWS SAM 範本位置相同。

```
$ cd sam-app
```

2. 執行以下命令：

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

3. 在互動式流程期間， AWS SAMCLI會提示您設定應用程式部署設定的選項。

   括號 ([ ]) 表示預設值。將您的答案留空，以選取預設值。預設值可從下列組態檔案取得：

- ~/.aws/config— 您的一般 AWS 帳戶設置。
- ~/.aws/credentials— 您的 AWS 帳戶憑據。
- *<project>*/samconfig.toml-您的項目的配置文件。

透過回答提示來 AWS SAMCLI提供值。例如，您可以輸入 **y** yes、**n** no 或字串值。

會 AWS SAMCLI將您的回應寫入專samconfig.toml案的檔案。對於後續部署，您可以使sam deploy用這些設定的值進行部署。若要重新設定這些值，請sam deploy --guided再次使用或直接修改您的組態檔案。

下面是一個示例輸出：

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

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

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

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

4. 接下來，將您的應用程式 AWS SAMCLI部署到 AWS 雲端. 在部署期間，進度會顯示在命令提示字元中。以下是部署的主要階段：

   - 對於 AWS Lambda 功能封裝為 .zip 檔案存檔的應用程式，會將套件 AWS SAMCLI壓縮並上傳到 Amazon Simple Storage Service (Amazon S3) 儲存貯體。如有必要， AWS SAMCLI將創建一個新存儲桶。
   - 對於將 Lambda 函數封裝為容器映像的應用程式，會將映像 AWS SAMCLI上傳至亞馬遜彈性容器登錄 (Amazon ECR)。如有必要， AWS SAMCLI將建立新的存放庫。
   - 會 AWS SAMCLI建立 AWS CloudFormation 變更集，並將您的應用程式部署到 AWS CloudFormation 堆疊中。
   - 會使用 Lambda 函數的新`CodeUri`值來 AWS SAMCLI修改已部署的 AWS SAM 範本。

   以下是部 AWS SAMCLI署輸出的範例：

```
        Looking for resources needed for deployment:

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

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

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

        Uploading to sam-app-zip/da3c598813f1c2151579b73ad788cac8  262144 / 619839
 (42.29%)Uploading to sam-app-zip/da3c598813f1c2151579b73ad788cac8  524288 / 619839
  (84.58%)Uploading to sam-app-zip/da3c598813f1c2151579b73ad788cac8  619839 /
 619839  (100.00%)

        Deploying with following values
```

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

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

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


Waiting for changeset to be created..

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

                         oWorldPermissionProd     n

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

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

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

                         yment47fc2d5f9d          yment

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

                         tage

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

```
                                                pi

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

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


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

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

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

CREATE_IN_PROGRESS       AWS::IAM::Role           HelloWorldFunctionRole   Resource
 creation

 Initiated
CREATE_COMPLETE          AWS::IAM::Role           HelloWorldFunctionRole   -

CREATE_IN_PROGRESS       AWS::Lambda::Function    HelloWorldFunction       -

CREATE_IN_PROGRESS       AWS::Lambda::Function    HelloWorldFunction       Resource
 creation

 Initiated
CREATE_COMPLETE          AWS::Lambda::Function    HelloWorldFunction       -

CREATE_IN_PROGRESS       AWS::ApiGateway::RestA   ServerlessRestApi        -

                         pi

CREATE_IN_PROGRESS       AWS::ApiGateway::RestA   ServerlessRestApi        Resource
 creation
```

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

```
                          tack

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

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

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

Description         Implicit IAM Role created for Hello World function

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

HelloWorldFunctionRole-11ZOGSCG28H0M


Key                 HelloWorldApi

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

Key                 HelloWorldFunction

Description         Hello World Lambda Function ARN

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

HelloWorldFunction-XPqNX4TBu7qn

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


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

5. 若要檢視已部署的應用程式，請執行下列動作：

   1. 直接使用網址打開 AWS CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
   2. 選取「堆疊」。

3. 依應用程式名稱識別您的堆疊，然後選取它。

在部署之前驗證變更

您可以將配置 AWS SAMCLI為顯示 AWS CloudFormation 變更集並在部署之前要求確認。

部署前確認變更的步驟

1. 在期間`sam deploy --guided`，輸入**Y**以在部署之前確認變更。

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

或者，您也可以使用下列方式修改`samconfig.toml`檔案：

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

2. 在部署期間， AWS SAMCLI會要求您在部署之前確認變更。以下是範例：

```
Waiting for changeset to be created..

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

                         oWorldPermissionProd     n

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

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

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

                         yment47fc2d5f9d          yment
```

```
+ Add                     ServerlessRestApiProdS   AWS::ApiGateway::Stage   N/A
                          tage
+ Add                     ServerlessRestApi        AWS::ApiGateway::RestA   N/A
                                                   pi
-----------------------------------------------------------------------------------------

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

在部署期間指定其他參數

您可以指定部署時要設定的其他參數值。您可以在部署期間修改 AWS SAM 範本並設定參數值來執行此操作。

若要指定其他參數

1. 修改 AWS SAM 範本的Parameters區段。以下是範例：

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

2. 執行 sam deploy --guided。下面是一個示例輸出：

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

Configuring SAM deploy
```

```
======================

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

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

設定 Lambda 函數的程式碼簽章

您可以在部署時為 Lambda 函數設定程式碼簽章。您可以在部署期間修改 AWS SAM 範本並設定程式碼簽章來達成此目的。

若要設定程式碼簽章

1. 在 AWS SAM 範本CodeSigningConfigArn中指定。以下是範例：

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

2. 執行 `sam deploy --guided`。 AWS SAMCLI會提示您設定程式碼簽章。下面是一個示例輸出：

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

```
#Signing profile details for layer 'MyLayer', which is used by functions
 {'HelloWorld'}
Signing Profile Name:
Signing Profile Owner Account ID (optional):
```

最佳實務

- 使用時sam deploy， AWS SAMCLI會部署位於.aws-sam目錄中的應用程式組建成品。當您對應用程式的原始檔案進行變更時，請執行sam build以在部署之前更新.aws-sam目錄。
- 第一次部署應用程式時，請使用sam deploy --guided來設定部署設定。對於後續部署，您可以使sam deploy用配置的設定進行部署。

山姆部署的選項

以下是的常用選項sam deploy。如需所有選項的清單，請參閱sam deploy。

使用引導式互動流程部署您的應用程式

使用--guided此選項可透過互動式流程設定應用程式的部署設定。以下是範例：

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

應用程式的部署設定會儲存在專samconfig.toml案的檔案中。如需進一步了解，請參閱設定專案設定。

故障診斷

若要疑難排解 AWS SAMCLI，請參閱AWS SAMCLI疑難排。

範例

部署包含封裝為 .zip 檔案封存的 Lambda 函數的 Hello World 應用程式

如需範例，請參閱 Hello World 應用程式教學課程步驟 3：將您的應用程式部署到 AWS 雲端中的。

部署包含封裝為容器映像檔的 Lambda 函數的 Hello World 應用程式

首先，我們用sam init來創建我們的 Hello World 應用程序。在互動式流程中，我們選擇Python3.9執行階段和Image套件類型。

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

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

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

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

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

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

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

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

接下來，我們cd到項目的根目錄並運行sam build。使用本機 AWS SAMCLI建置我們的 Lambda 函數Docker。

```
sam-app $ sam build
Building codeuri: /Users/.../sam-app runtime: None metadata: {'Dockerfile':
 'Dockerfile', 'DockerContext': '/Users/.../sam-app/hello_world', 'DockerTag':
 'python3.9-v1'} architecture: x86_64 functions: HelloWorldFunction
Building image for HelloWorldFunction function
Setting DockerBuildArgs: {} for HelloWorldFunction function
Step 1/5 : FROM public.ecr.aws/lambda/python:3.9
 ---> 0a5e3da309aa
Step 2/5 : COPY requirements.txt ./
 ---> abc4e82e85f9
Step 3/5 : RUN python3.9 -m pip install -r requirements.txt -t .
 ---> [Warning] The requested image's platform (linux/amd64) does not match the
 detected host platform (linux/arm64/v8) and no specific platform was requested
 ---> Running in 43845e7aa22d
Collecting requests
  Downloading requests-2.28.2-py3-none-any.whl (62 kB)
     ######################################## 62.8/62.8 KB 829.5 kB/s eta 0:00:00
Collecting idna<4,>=2.5
  Downloading idna-3.4-py3-none-any.whl (61 kB)
     ######################################## 61.5/61.5 KB 2.4 MB/s eta 0:00:00
Collecting charset-normalizer<4,>=2
  Downloading charset_normalizer-3.1.0-cp39-cp39-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl (199 kB)
     ######################################## 199.2/199.2 KB 2.1 MB/s eta 0:00:00
Collecting certifi>=2017.4.17
  Downloading certifi-2022.12.7-py3-none-any.whl (155 kB)
     ######################################## 155.3/155.3 KB 10.2 MB/s eta 0:00:00
Collecting urllib3<1.27,>=1.21.1
  Downloading urllib3-1.26.15-py2.py3-none-any.whl (140 kB)
     ######################################## 140.9/140.9 KB 9.1 MB/s eta 0:00:00
Installing collected packages: urllib3, idna, charset-normalizer, certifi, requests
Successfully installed certifi-2022.12.7 charset-normalizer-3.1.0 idna-3.4
 requests-2.28.2 urllib3-1.26.15
Removing intermediate container 43845e7aa22d
 ---> cab8ace899ce
Step 4/5 : COPY app.py ./
```

```
 ---> 4146f3cd69f2
Step 5/5 : CMD ["app.lambda_handler"]
 ---> [Warning] The requested image's platform (linux/amd64) does not match the
 detected host platform (linux/arm64/v8) and no specific platform was requested
 ---> Running in f4131ddffb31
Removing intermediate container f4131ddffb31
 ---> d2f5180b2154
Successfully built d2f5180b2154
Successfully tagged helloworldfunction:python3.9-v1


Build Succeeded

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

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

接下來，我們執行`sam deploy --guided`部署我們的應用程式。這會 AWS SAMCLI引導我們完成設定部署設定。然後，將我們的應用程式 AWS SAMCLI部署到 AWS 雲端.

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

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

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

        Setting default arguments for 'sam deploy'
        =========================================
        Stack Name [sam-app]: ENTER
        AWS Region [us-west-2]: ENTER
        #Shows you resources changes to be deployed and require a 'Y' to initiate
 deploy
        Confirm changes before deploy [Y/n]: ENTER
        #SAM needs permission to be able to create roles to connect to the resources in
 your template
```

```
        Allow SAM CLI IAM role creation [Y/n]: ENTER
        #Preserves the state of previously provisioned resources when an operation
 fails
        Disable rollback [y/N]: ENTER
        HelloWorldFunction may not have authorization defined, Is this okay? [y/N]: y
        Save arguments to configuration file [Y/n]: ENTER
        SAM configuration file [samconfig.toml]: ENTER
        SAM configuration environment [default]: ENTER

        Looking for resources needed for deployment:

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

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

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

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

        Deploying with following values
        ===============================
        Stack name                   : sam-app
        Region                       : us-west-2
```

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

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

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


Waiting for changeset to be created..

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

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

                         oWorldPermissionProd     n

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

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

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

                         yment47fc2d5f9d          yment

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

```
                           tage

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

                                                    pi

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


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


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

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

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

CREATE_IN_PROGRESS       AWS::IAM::Role           HelloWorldFunctionRole   -

CREATE_IN_PROGRESS       AWS::IAM::Role           HelloWorldFunctionRole   Resource
 creation
                                                                            Initiated

CREATE_COMPLETE          AWS::IAM::Role           HelloWorldFunctionRole   -

CREATE_IN_PROGRESS       AWS::Lambda::Function    HelloWorldFunction       -

CREATE_IN_PROGRESS       AWS::Lambda::Function    HelloWorldFunction       Resource
 creation
                                                                            Initiated
```

```
CREATE_COMPLETE          AWS::Lambda::Function    HelloWorldFunction       -

CREATE_IN_PROGRESS       AWS::ApiGateway::RestA   ServerlessRestApi        -
                         pi

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

CREATE_COMPLETE          AWS::ApiGateway::RestA   ServerlessRestApi        -
                         pi

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

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

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

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

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

CREATE_IN_PROGRESS       AWS::ApiGateway::Stage   ServerlessRestApiProdS   -
                                                  tage

CREATE_IN_PROGRESS       AWS::ApiGateway::Stage   ServerlessRestApiProdS   Resource
 creation
                                                  tage                     Initiated

CREATE_COMPLETE          AWS::ApiGateway::Stage   ServerlessRestApiProdS   -
```

```
                                                tage

CREATE_COMPLETE          AWS::Lambda::Permissio   HelloWorldFunctionHell   -

                         n                        oWorldPermissionProd

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

                         tack

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

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

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

Description         Implicit IAM Role created for Hello World function

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

Key                 HelloWorldApi

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

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


Key                 HelloWorldFunction

Description         Hello World Lambda Function ARN

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

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


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

進一步了解

若要進一步瞭解如何使用 AWS SAMCLIsam deploy指令，請參閱下列內容：

- 完整 AWS SAM 研討會：單元 3-手動部署 — 瞭解如何使用 AWS SAMCLI.

用於部署應用程式的選項 AWS SAM

您可以使用手動部署應用程式 AWS SAM，也可以自動化部署。使用手動部署您的應用程式。 AWS SAMCLI若要自動化部署，請使用管道和持續整合與持續部署 (CI/CD) 系統。本節中的主題提供有關這兩種方法的資訊。

主題

- 如何使用手 AWS SAMCLI動部署
- 使用 CI/CD 系統和管線進行部署
- 逐步部署
- 疑難排解使用的部署 AWS SAMCLI
- 進一步了解

如何使用手 AWS SAMCLI動部署

在本機開發和測試無伺服器應用程式之後，您可以使用sam deploy指令來部署應用程式。

若要透過提示 AWS SAM 引導您完成部署，請指定--guided旗標。當您指定此旗標時，sam deploy命令會壓縮您的應用程式成品、將它們上傳到 Amazon Simple Storage Service (Amazon S3) (針對 .zip 檔案存檔) 或 Amazon Elastic Container Registry (Amazon ECR) (適用於容器映像)。然後，命令會將您的應用程式部署到 AWS 雲端。

範例：

```
# Deploy an application using prompts:
sam deploy --guided
```

使用 CI/CD 系統和管線進行部署

AWS SAM 協助您使用管線和持續整合與持續部署 (CI/CD) 系統自動化部署。 AWS SAM 可用來建立管道，並簡化無伺服器應用程式的 CI/CD 工作。多個 CI/CD 系統支援 AWS SAM 建置容器映像，同時 AWS SAM 也為多個 CI/CD 系統提供一組預設管線範本，以封裝 AWS最佳部署作法。

如需詳細資訊，請參閱 使用 CI/CD 系統和管道進行部署 AWS SAM。

逐步部署

如果您想要逐步部署 AWS SAM 應用程式，而不是一次全部部署，您可以指定 AWS CodeDeploy 提供的部署組態。若要取得更多資訊，請參閱《使用指南》 CodeDeploy中的〈AWS CodeDeploy 使用部署規劃〉。

如需設定 AWS SAM 應用程式以逐步部署的相關資訊，請參閱逐步部署無伺服器應用程。

疑難排解使用的部署 AWS SAMCLI

AWS SAMCLI錯誤：「安全約束不滿意」

執行時sam deploy --guided，系統會提示您提供問題`HelloWorldFunction may not have authorization defined, Is this okay? [y/N]`。如果您使用 **N** (預設回應) 回應此提示，您會看到下列錯誤：

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

提示會通知您即將部署的應用程式可能已在未經授權的情況下設定 Amazon API Gateway API。通過響**N**應此提示，您說這是不確定的。

若要修正此問題，您有下列選項：

- 使用授權配置您的應用程序。如需有關配置授權的資訊，請參閱使用 AWS SAM 範本控制 API 存取。
- 回應此問題，指**Y**出您可以部署具有未經授權設定 API Gateway API 的應用程式。

進一步了解

如需部署無伺服器應用程式的實作範例，請參閱完整 AWS SAM 研討會的下列內容：

- 單元 3-手動部署 — 瞭解如何使用 AWS SAMCLI.
- 單元 4-CI/CD — 瞭解如何透過建立持續整合與交付 (CI/CD) 管道，將建置、封裝和部署階段自動化。

使用 CI/CD 系統和管道進行部署 AWS SAM

AWS SAM 協助組織針對偏好的 CI/CD 系統建立管道，以便他們以最少的努力實現 CI/CD 的優點，例如加速部署頻率、縮短變更的前置時間，以及減少部署錯誤。

AWS SAM 透過建置容器映像檔的協助，簡化無伺服器應用程式的 CI/CD 工作。 AWS SAM 提供的映像檔包括許多受支援的 AWS Lambda 執行階段的 AWS SAMCLI和建置工具。這可讓您更輕鬆地使用 AWS SAMCLI. 這些映像檔也減輕了團隊為 CI/CD 系統建立和管理自己的映像檔的需求。如需有關 AWS SAM 組建容器映像檔的詳細資訊，請參閱映像儲存庫。

多個 CI/CD 系統支持 AWS SAM 構建容器映像。您應該使用哪種 CI/CD 系統取決於幾個因素。其中包括您的應用程式是使用單一執行階段還是多個執行階段，或是要在容器映像檔內建置應用程式，還是直接在主機 (虛擬機器 (VM) 或裸機主機上建置應用程式。

AWS SAM 也為多個 CI/CD 系統提供一組預設管線範本，這些範本封裝了部署 AWS最佳實務。這些預設管線範本使用標準 JSON/YAML 管線組態格式，而內建的最佳實務可協助執行多帳戶和多區域部署，並確認管道無法對基礎結構進行非預期的變更。

您有兩個主要選項可用 AWS SAM 來部署無伺服器應用程式：1) 修改現有的管線組態以使用 AWS SAMCLI指令，或 2) 產生範例 CI/CD 管線組態，以作為自己應用程式的起點。

主題

- 什麼是管道？
- 產生起動器 CI/CD 管線
- 如何自訂入門管道
- 自動化 AWS SAM 應用程式的部署
- 如何搭配管線使用 OIDC 驗證 AWS SAM
- 如何在部署時上傳本地文件 AWS SAMCLI

什麼是管道？

管線是執行以發行新版應用程式的自動化步驟序列。有了 AWS SAM，您可以使用許多常見的 CI/CD 系統來部署您的應用程式，包括詹金斯AWS CodePipeline、GitLab CI/ CD 和動作。GitHub

管道範本包括 AWS 部署最佳實務，可協助進行多帳戶和多區域部署。 AWS 開發和生產等環境通常存在於不同的 AWS 帳戶中。這可讓開發團隊設定安全的部署管線，而不會對基礎結構進行意外變更。

您也可以提供自己的自訂管道範本，以協助跨開發團隊標準化管道。

產生起動器 CI/CD 管線

當您準備好自動化部署時，您可以使用其中 AWS SAM一個入門管道範本，為您選擇使用的 CI/CD 系統產生部署管線。部署管道是您設定和用來自動化無伺服器應用程式部署的管道。入門管道範本已預先設定，可協助您快速設定無伺服器應用程式的部署管道。

使用入門管線範本，您可以使用sam pipeline init指令在幾分鐘內產生管線。

入門管線範本使用 CI/CD 系統熟悉的 JSON/YAML 語法，並整合了最佳實務，例如跨多個帳戶和區域管理成品，以及使用部署應用程式所需的最低權限量。目前， AWS SAM CLI 支持生成詹金斯，CI/CD AWS CodePipeline，GitHub 操作和比特桶管道的入門 GitLab CI/CD 管道配置。

以下是產生入門管線組態所需執行的高階工作：

1. 建立基礎設施資源 — 您的管道需要特定 AWS 資源，例如 IAM 使用者和具有必要許可的角色、Amazon S3 儲存貯體，以及選擇性使用 Amazon ECR 儲存庫。
2. 將 Git 儲存庫與 CI/CD 系統 Connect — 您的 CI/CD 系統需要知道哪個 Git 儲存庫會觸發管道執行。請注意，此步驟可能不是必要的，具體取決於您使用的 Git 存儲庫和 CI/CD 系統的組合。
3. 產生管線組態 — 此步驟會產生包含兩個部署階段的入門管線組態。
4. 將您的管道配置提交到 Git 存儲庫 — 此步驟對於確保 CI/CD 系統知道您的管道配置，並在提交更改時運行是必要的。

在您產生入門管線組態並將其提交至 Git 儲存庫之後，每當有人對該儲存庫提交程式碼變更時，您的管道就會觸發自動執行。

這些步驟的順序以及每個步驟的詳細資訊，會根據您的 CI/CD 系統而有所不同：

- 如果您正在使用 AWS CodePipeline，請參閱生成啟動管道 AWS CodePipeline。
- 如果您正在使用詹金斯， GitLab CI/CD， GitHub 操作或比特桶管道，請參閱。為詹金斯， GitLab CI/CD， GitHub 操作或比特桶管道生成入門管道

生成啟動管道 AWS CodePipeline

欲產生的入門配管組態AWS CodePipeline，請依此順序執行下列工作：

1. 建立基礎結構資
2. 產生管線組態
3. 將您的管道配置提交給 Git
4. 將您的 Git 儲存庫與您的 CI/CD 系統 Connect

> **Note**
>
> 下列程序會使用兩個AWS SAMCLI指令`sam pipeline bootstrap`和`sam pipeline init`。有兩個命令的原因是要處理這種使用案例：系統管理員 (也就是說，需要設定基礎架構AWS資源 (例如 IAM 使用者和角色之類的權限的使用者) 擁有更多權限的開發人員 (也就是說，只需要設定個別管道權限的使用者，但不需要所需基礎設施AWS資源的使用者)。

步驟 1：建立基礎結構資源

使用的管道AWS SAM需要特定AWS資源，例如 IAM 使用者和具有必要許可的角色、Amazon S3 儲存貯體，以及選擇性使用 Amazon ECR 儲存庫。管道的每個部署階段都必須有一組基礎結構資源。

您可以執行下列命令來協助進行此設定：

```
sam pipeline bootstrap
```

> **Note**
>
> 針對管線的每個部署階段執行上一個命令。

步驟 2：產生管線組態

欲產生配管組態，請執行下列命令：

```
sam pipeline init
```

第 3 步：將您的管道配置提交到 Git 存儲庫

為了確保您的 CI/CD 系統知道您的管線組態，並在確認變更時執行此步驟是必要的。

步驟 4：將您的 Git 儲存庫與您的 CI/CD 系統 Connect

因為AWS CodePipeline您現在可以通過運行以下命令來創建連接：

```
sam deploy -t codepipeline.yaml --stack-name <pipeline-stack-name> --
capabilities=CAPABILITY_IAM --region <region-X>
```

如果您正在使用 GitHub 或 Bitbucket，在先前執行sam deploy命令之後，請遵循開發人員工具主控台使用者指南中更新擱置連線主題中「若要完成連線」下的步驟，完成連線。此外，存儲sam deploy命令輸出CodeStarConnectionArn中的副本，因為如果您想AWS CodePipeline與另一個分支一起使用，則需要它main。

配置其他分支

預設情況下，AWS CodePipeline會將main分支與AWS SAM. 如果您想要使用以外的分支main，則必須再次執行該sam deploy命令。請注意，根據您使用的是哪個 Git 存儲庫，您可能還需要提供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 AWS CodeCommit
sam deploy -t codepipeline.yaml --stack-name <feature-pipeline-stack-name> --
capabilities=CAPABILITY_IAM --parameter-overrides="FeatureGitBranch=<branch-name>"
```

進一步了解

如需設定 CI/CD 管線的實際操作範例，請參閱完整研討會AWS CodePipeline中的 CI/CD。AWS SAM

為詹金斯， GitLab CI/CD， GitHub 操作或比特桶管道生成入門管道

若要產生 Jenkins、 GitLab CI/CD、 GitHub 動作或 Bitbucket 管道的入門管線組態，請依此順序執行下列工作：

1. 建立基礎結構資
2. 將您的 Git 儲存庫與您的 CI/CD 系統 Connect
3. 建立認證物件
4. 產生管線組態

5. 將您的管道配置提交到 Git 存儲庫

> **Note**
>
> 下列程序會使用兩個 AWS SAMCLI指令`sam pipeline bootstrap`和`sam pipeline init`。有兩個命令的原因是要處理這種使用案例：管理員 (也就是說，需要設定基礎架構 AWS 資源 (例如 IAM 使用者和角色) 的使用者擁有更多權限，開發人員 (也就是說，只需要設定個別管道權限的使用者，但不需要所需基礎設施 AWS 資源的使用者)。

步驟 1：建立基礎結構資源

使用的管道 AWS SAM 需要特定 AWS 資源，例如 IAM 使用者和具有必要許可的角色、Amazon S3 儲存貯體，以及選擇性使用 Amazon ECR 儲存庫。管道的每個部署階段都必須有一組基礎結構資源。

您可以執行下列命令來協助進行此設定：

```
sam pipeline bootstrap
```

> **Note**
>
> 針對管線的每個部署階段執行上一個命令。

您必須為管線的每個部署階段擷取管道使用者的 AWS 認證 (金鑰 ID 和秘密金鑰)，因為後續步驟需要這些認證。

步驟 2：將您的 Git 儲存庫與您的 CI/CD 系統 Connect

您必須將 Git 儲存庫連接到 CI/CD 系統，以便 CI/CD 系統能夠存取您的應用程式原始程式碼以進行建置和部署。

> **Note**
>
> 如果您使用下列其中一種組合，則可略過此步驟，因為連接會自動為您完成：
>
> 1. GitHub GitHub 存放庫的動作
> 2. GitLab 含儲存庫的 CI/CD GitLab

3. 具有比特桶存儲庫的比特桶管道

若要將 Git 儲存庫與 CI/CD 系統連線，請執行下列其中一個動作：

- 如果您使用的是詹金斯，請參閱詹金斯文檔「添加分支源」。
- 如果您使用的是 GitLab CI/CD 和 Git 儲存庫 GitLab，請參閱「連接外部儲存庫」的GitLab文件。

步驟 3：創建憑據對象

每個 CI/CD 系統都有自己的方式來管理 CI/CD 系統存取 Git 儲存庫所需的認證。

若要建立必要的認證物件，請執行下列其中一個動作：

- 如果您使用的是 Jenkins，請創建一個存儲密鑰 ID 和密鑰的單個「憑據」。按照在構建詹金斯管道與 AWS SAM博客的說明，在配置詹金斯部分。您將需要「憑據 ID」進行下一步。
- 如果您使用的是 GitLab CI/CD，請創建兩個「受保護的變量」，每個密鑰 ID 和密鑰一個。按照GitLab 文檔中的說明進行操作-下一步需要兩個「可變密鑰」。
- 如果您使用的是 GitHub 動作，請建立兩個「加密密碼」，每個金鑰和私密金鑰各一個。按照GitHub文檔中的說明進行操作-下一步需要兩個「秘密名稱」。
- 如果您使用的是 Bitbucket 管道，請創建兩個「安全變量」，每個密鑰 ID 和密鑰一個。按照變量和秘密中的說明進行操作-下一步需要兩個「秘密名稱」。

步驟 4：產生管線組態

若要產生管線組態，請執行下列命令。您將需要輸入您在上一個步驟中創建的憑據對象：

```
sam pipeline init
```

第 5 步：將您的管道配置提交到 Git 存儲庫

為了確保您的 CI/CD 系統知道您的管線組態，並在確認變更時執行此步驟是必要的。

進一步了解

如需使用設定 CI/CD 管線的實際操作範例GitHub Actions，請參閱完整研討會GitHub中的 CI/CD 與。AWS SAM

如何自訂入門管道

身為 CI/CD 管理員，您可能想要自訂入門管線範本和相關的引導式提示，讓組織中的開發人員可以用來建立管線組態。

建立初學者範本時， AWS SAMCLI會使用「曲奇」範本。有關餅乾切割器模板的詳細信息，餅乾。

您也可以自訂使用`sam pipeline init`指令建立管線組態時向使用者 AWS SAMCLI顯示的提示。若要自訂使用者提示，請執行下列動作：

1. 建立**`questions.json`**檔案 — 檔`questions.json`案必須位於專案存放庫的根目錄中。這是與`cookiecutter.json`檔案相同的目錄。若要檢視`questions.json`檔案的結構描述，請參閱問題 .json. schema。若要檢視範例`questions.json`檔案，請參閱問題 .json。
2. 使用 Cookie 執行器名稱對應問題索引鍵 — `questions.json` 檔案中的每個物件都需要一個與 Cookie 執行器範本中名稱相符的金鑰。此鍵匹配是如何將用戶提示響應 AWS SAMCLI映射到 cookie 切割器模板。若要查看此金鑰比對的範例，請參閱本主題稍後的範例檔案章節。
3. 建立**`metadata.json`**檔案 — 宣告管線在檔`metadata.json`案中將具有的階段數。階段數會指示`sam pipeline init`命令要提示有多少個階段，或在`--bootstrap`選項的情況下，要建立基礎結構資源的階段數目。若要檢視宣告具有兩個階段之管線的範例`metadata.json`檔案，請參閱 metadata.json。

範例專案

以下是範例專案，每個專案都包含一個 Cookiecutter 範本、一個檔案和一個`questions.json`檔案：`metadata.json`

- 詹金斯例如：兩階段詹金斯管道模板
- CodePipeline 範例：兩階段 CodePipeline 管線範本

範例檔案

下列檔案集顯示檔案中的問題如何與 Cookiecutter 範本`questions.json`檔案中的項目相關聯。請注意，這些範例是檔案片段，而非完整檔案。若要查看完整檔案的範例，請參範例專案閱本主題稍早的章節。

範例：**`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"
  }, {
    ...
```

範例：**cookiecutter.json**

```
{
  "outputDir": "aws-sam-pipeline",
  "pipeline_user_jenkins_credential_id": "",
  "sam_template": "",
    ...
```

範例：**Jenkinsfile**

```
pipeline {
  agent any
  environment {
    PIPELINE_USER_CREDENTIAL_ID =
 '{{cookiecutter.pipeline_user_jenkins_credential_id}}'
    SAM_TEMPLATE = '{{cookiecutter.sam_template}}'
    ...
```

自動化 AWS SAM 應用程式的部署

在中 AWS SAM，您自動化應用程式部署的方 AWS SAM 式視您使用的 CI/CD 系統而有所不同。因此，本節中的範例說明如何設定各種 CI/CD 系統，以便在組建容器映像中自動 AWS SAM 建置無伺服器應用程式。這些建置容器映像檔可讓您更輕鬆地使用 AWS SAMCLI.

根據您使用的 CI/CD 系統，現有 CI/CD 管道部署無伺服器應 AWS SAM 用程式的程序會略有不同。

下列主題提供設定 CI/CD 系統以在組建容器映像 AWS SAM 中建置無伺服器應用程式的範例：

主題

- 部署使用 AWS CodePipeline
- 使用比特桶管道進行部署
- 使用詹金斯部署
- 使用 GitLab CI/CD 進行部署
- 使用 GitHub 動作部署

部署使用 AWS CodePipeline

若要將AWS CodePipeline管線設定為自動化 AWS SAM 應用程式的建置和部署，您的 AWS CloudFormation 範本和`buildspec.yml`檔案必須包含執行下列動作的行：

1. 使用可用映像中的必要運行時引用構建容器映像。下列範例會使用組`public.ecr.aws/sam/build-nodejs20.x`建容器映像檔。
2. 設定管線階段以執行必要的 AWS SAM 命令列介面 (CLI) 命令。下列範例會執行兩個 AWS SAMCLI 命令：sam build和 sam deploy (使用必要的選項)。

此範例假設您已使用宣告 AWS SAM 範本檔案中的所有函數和圖層`runtime: nodejs20.x`。

AWS CloudFormation 模板片段：

```
  CodeBuildProject:
    Type: AWS::CodeBuild::Project
    Properties:
      Environment:
        ComputeType: BUILD_GENERAL1_SMALL
        Image: public.ecr.aws/sam/build-nodejs20.x
        Type: LINUX_CONTAINER
      ...
```

`buildspec.yml`片段：

```
version: 0.2
phases:
  build:
```

```
    commands:
      - sam build
      - sam deploy --no-confirm-changeset --no-fail-on-empty-changeset
```

如需可用的 Amazon Elastic Container Registry (Amazon ECR) 清單，請參閱。映像儲存庫

使用比特桶管道進行部署

若要將 Bitbucket 管線設定為自動化 AWS SAM 應用程式的建置和部署，您的`bitbucket-pipelines.yml`檔案必須包含執行下列動作的行：

1. 使用可用映像中的必要運行時引用構建容器映像。下列範例會使用組`public.ecr.aws/sam/build-nodejs20.x`建容器映像檔。
2. 設定管線階段以執行必要的 AWS SAM 命令列介面 (CLI) 命令。下列範例會執行兩個 AWS SAMCLI 命令：sam build和 sam deploy (使用必要的選項)。

此範例假設您已使用宣告 AWS SAM 範本檔案中的所有函數和圖層`runtime: nodejs20.x`。

```
image: public.ecr.aws/sam/build-nodejs20.x

pipelines:
  branches:
    main: # branch name
      - step:
          name: Build and Package
          script:
            - sam build
            - sam deploy --no-confirm-changeset --no-fail-on-empty-changeset
```

如需可用的 Amazon Elastic Container Registry (Amazon ECR) 清單，請參閱〈〉建置不同執行階段的容器映像。映像儲存庫

使用詹金斯部署

要配置 Jenkins 管道以自動化 AWS SAM 應用程序的構建和部署，您`Jenkinsfile`必須包含執行以下操作的行：

1. 使用可用映像中的必要運行時引用構建容器映像。下列範例會使用組`public.ecr.aws/sam/build-nodejs20.x`建容器映像檔。

2. 設定管線階段以執行必要的 AWS SAM 命令列介面 (CLI) 命令。下列範例會執行兩個 AWS SAMCLI 命令：sam build和 sam deploy (使用必要的選項)。

此範例假設您已使用宣告 AWS SAM 範本檔案中的所有函數和圖層`runtime: nodejs20.x`。

```
pipeline {
    agent { docker { image 'public.ecr.aws/sam/build-nodejs20.x' } }
    stages {
        stage('build') {
            steps {
                sh 'sam build'
                sh 'sam deploy --no-confirm-changeset --no-fail-on-empty-changeset'
            }
        }
    }
}
```

如需可用的 Amazon 彈性容器登錄檔 (Amazon ECR) 清單，請參閱〈〉建置不同執行階段的容器映像檔，請參閱。映像儲存庫

使用 GitLab CI/CD 進行部署

若要將GitLab管線設定為自動化 AWS SAM 應用程式的建置和部署，您的`gitlab-ci.yml`檔案必須包含執行下列動作的行：

1. 使用可用映像中的必要運行時引用構建容器映像。下列範例會使用組`public.ecr.aws/sam/build-nodejs20.x`建容器映像檔。
2. 設定管線階段以執行必要的 AWS SAM 命令列介面 (CLI) 命令。下列範例會執行兩個 AWS SAMCLI 命令：sam build和 sam deploy (使用必要的選項)。

此範例假設您已使用宣告 AWS SAM 範本檔案中的所有函數和圖層`runtime: nodejs20.x`。

```
image: public.ecr.aws/sam/build-nodejs20.x
deploy:
  script:
    - sam build
    - sam deploy --no-confirm-changeset --no-fail-on-empty-changeset
```

如需可用的 Amazon Elastic Container Registry (Amazon ECR) 清單，請參閱。映像儲存庫

使用 GitHub 動作部署

若要將GitHub管線設定為自動化AWS SAM應用程式的建置和部署，您必須先在主機上安裝AWS SAM命令列介面 (CLI)。您可以在工GitHub 作 GitHub 流程中使用「動作」 來協助進行此設定。

下列範例 GitHub 工作流程使用一系列 GitHub動作來設定 Ubuntu 主機，然後執行AWS SAMCLI命令來建置和部署AWS SAM應用程式：

```
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
```

如需可用的 Amazon Elastic Container Registry (Amazon ECR) 清單，請參閱〈〉建置不同執行階段的容器映像。映像儲存庫

如何搭配管線使用 OIDC 驗證 AWS SAM

AWS Serverless Application Model （ AWS SAM ）支持 OpenID Connect （ OIDC ）用戶身份驗證，用於比特存儲桶， GitHub操作以及持 GitLab 續集成和持續交付（ CI/CD ）平台。透過此支援，您可以使用這些平台的授權 CI/CD 使用者帳戶來管理無伺服器應用程式管道。否則，您需要建立和管理多個 AWS Identity and Access Management (IAM) 使用者，以控制對 AWS SAM 管道的存取。

使用管道設定 OIDC AWS SAM

在`sam pipeline bootstrap`組態程序期間，請執行下列動作以使用管線設定 OIDC。 AWS SAM

1. 當系統提示您選擇身分識別提供者時，請選取 OIDC。

2. 接下來，選取支援的 OIDC 提供者。
3. 輸入 OIDC 提供者 URL，開頭為。**`https://`**

> **Note**
>
> AWS SAM 生成`AWS::IAM::OIDCProvider`資源類型時引用此 URL。

4. 接下來，依照提示輸入存取所選平台所需的 CI/CD 平台資訊。這些詳細資料因平台而異，可能包括：
   - OIDC 用戶端識別碼。
   - 程式碼儲存庫名稱或通用唯一識別碼 (UUID)。
   - 與存放庫相關聯的群組或組織名稱。
   - GitHub 程式碼儲存庫所屬的組織。
   - GitHub 儲存庫名稱。
   - 將發生部署的分支。
5. AWS SAM 顯示輸入 OIDC 組態的摘要。輸入要編輯的設定數字，或按Enter繼續。
6. 當系統提示您確認建立支援輸入的 OIDC 連線所需的資源時，按下Y以繼續。

AWS SAM 使用提供的配置生成具有承擔管線執行角色的`AWS::IAM::OIDCProvider` AWS CloudFormation 資源。若要進一步了解此 AWS CloudFormation 資源類型，請參閱使用者指南中的 AWS:: IAM:: OIDC 供應商。AWS CloudFormation

> **Note**
>
> 如果身分識別提供者 (IdP) 資源已存在於您的中 AWS 帳戶，請 AWS SAM 參考該資源，而不是建立新資源。

範例

以下是使 AWS SAM 用管線設定 OIDC 的範例。

```
Select a permissions provider:
    1 - IAM (default)
    2 - OpenID Connect (OIDC)
Choice (1, 2): 2
```

```
Select an OIDC provider:
    1 - GitHub Actions
    2 - GitLab
    3 - Bitbucket
Choice (1, 2, 3): 1
Enter the URL of the OIDC provider [https://token.actions.githubusercontent.com]:
Enter the OIDC client ID (sometimes called audience) [sts.amazonaws.com]:
Enter the GitHub organization that the code repository belongs to. If there is no
 organization enter your username instead: my-org
Enter GitHub repository name: testing
Enter the name of the branch that deployments will occur from [main]:

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

[4] Summary
Below is the summary of the answers:
    1 - Account: 123456
    2 - Stage configuration name: dev
    3 - Region: us-east-1
    4 - OIDC identity provider URL: https://token.actions.githubusercontent.com
    5 - OIDC client ID: sts.amazonaws.com
    6 - GitHub organization: my-org
    7 - GitHub repository: testing
    8 - Deployment branch: main
    9 - Pipeline execution role: [to be created]
    10 - CloudFormation execution role: [to be created]
    11 - Artifacts bucket: [to be created]
    12 - ECR image repository: [skipped]
Press enter to confirm the values above, or select an item to edit the value:

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

進一步了解

如需將 OIDC 與 AWS SAM 管線搭配使用的詳細資訊，請參閱。sam pipeline bootstrap

如何在部署時上傳本地文件 AWS SAMCLI

在開發時，您通常會發現將應用程式程式碼分解為不同的檔案，以便更好地組織和管理您的應用程式是有益的。這方面的一個基本示例是將 AWS Lambda 函數代碼與基礎結構代碼分開。您可以在專案的子目錄中組織 Lambda 函數程式碼，並在 AWS Serverless Application Model (AWS SAM) 範本中參考其本機路徑，以達到此目的。

將應用程式部署到時 AWS 雲端， AWS CloudFormation 需要先將本機檔案上傳到可存取的 AWS 服務，例如 Amazon Simple Storage Service (Amazon S3)。您可以使用 AWS SAMCLI來自動執行此程序。使用`sam deploy`或`sam package`指令執行下列作業：

1. 自動將本機檔案上傳至可存取的 AWS 服務。
2. 自動更新您的應用程式範本以參照新的檔案路徑。

主題

- 示範：使用上 AWS SAMCLI傳 Lambda 函數程式碼
- 支援的使用案例
- 進一步了解

示範：使用上 AWS SAMCLI傳 Lambda 函數程式碼

在此示範中，我們使用 Lambda 函數的 .zip 套件類型，初始化範例 Hello World 應用程式。我們使用自動 AWS SAMCLI將 Lambda 函數程式碼上傳到 Amazon S3，並在應用程式範本中參考其新路徑。

首先，我們運行`sam init`初始化我們的 Hello World 應用程序。

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

Choose an AWS Quick Start application template
        1 - Hello World Example
```

```
        2 - Multi-step workflow
        ...
Template: 1

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

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

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

Project name [sam-app]: demo

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

...
```

我們的 Lambda 函數程式碼會組織在專案的`hello_world`子目錄中。

```
demo
### README.md
### hello_world
#   ### __init__.py
#   ### app.py
#   ### requirements.txt
### template.yaml
### tests
```

在 AWS SAM 範本中，我們使用`CodeUri`屬性參考 Lambda 函數程式碼的本機路徑。

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

```
...
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function # More info about Function Resource:
 https://github.com/awslabs/serverless-application-model/blob/master/
versions/2016-10-31.md#awsserverlessfunction
    Properties:
      CodeUri: hello_world/
      Handler: app.lambda_handler
      Runtime: python3.9
      ...
```

接下來，我們執sam build行建置應用程式並準備部署。

```
$ sam build
Starting Build use cache
Manifest file is changed (new hash: 3298f13049d19cffaa37ca931dd4d421) or dependency
 folder (.aws-sam/deps/7896875f-9bcc-4350-8adb-2c1d543627a1) is missing for
 (HelloWorldFunction), downloading dependencies and copying/building source
Building codeuri: /Users/.../demo/hello_world runtime: python3.9 metadata: {}
 architecture: x86_64 functions: HelloWorldFunction
Running PythonPipBuilder:CleanUp
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource
Running PythonPipBuilder:CopySource

Build Succeeded

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

接下來，我們執行sam deploy --guided部署我們的應用程式。

```
$ sam deploy --guided

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

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

        Setting default arguments for 'sam deploy'
```

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

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

File with same data already exists at demo/da3c598813f1c2151579b73ad788cac8, skipping
 upload

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

Initiating deployment
=====================
```

```
...
Waiting for changeset to be created..
CloudFormation stack changeset
---------------------------------------------------------------------------------------------
Operation                LogicalResourceId        ResourceType             Replacement

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

                         oWorldPermissionProd     n

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

...
---------------------------------------------------------------------------------------------

Changeset created successfully. arn:aws:cloudformation:us-
west-2:012345678910:changeSet/samcli-deploy1680906292/1164338d-72e7-4593-a372-
f2b3e67f542f

2023-04-07 12:24:58 - Waiting for stack create/update to complete

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

CREATE_IN_PROGRESS       AWS::IAM::Role           HelloWorldFunctionRole   Resource
 creation
                                                                           Initiated

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

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

Description         Implicit IAM Role created for Hello World function
```

```
Value                arn:aws:iam::012345678910:role/demo-HelloWorldFunctionRole-
VQ4CU7UY7S2K

Key                  HelloWorldApi

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

Value                https://satnon55e9.execute-api.us-west-2.amazonaws.com/Prod/hello/


Key                  HelloWorldFunction

Description          Hello World Lambda Function ARN

Value                arn:aws:lambda:us-west-2:012345678910:function:demo-

HelloWorldFunction-G14inKTmSQvK

-----------------------------------------------------------------------------------------
Successfully created/updated stack - demo in us-west-2
```

在部署期間，會 AWS SAMCLI自動將 Lambda 函數程式碼上傳到 Amazon S3，並更新我們的範本。我們在 AWS CloudFormation 主控台中修改過的範本會反映 Amazon S3 儲存貯體路徑。

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: s3://aws-sam-cli-managed-default-samclisourcebucket-1a4x26zbcdkqr/demo/
da3c598813f1c2151579b73ad788cac8
      Handler: app.lambda_handler
      ...
```

支援的使用案例

AWS SAMCLI可以自動促進許多檔案類型、 AWS CloudFormation 資源類型和 AWS CloudFormation 巨集的此程序。

檔案類型

支持應用程序文件和Docker圖像。

AWS CloudFormation 資源類型

以下是支援的資源類型及其屬性的清單：

資源	屬性
AWS::ApiGateway::RestApi	BodyS3Location
AWS::ApiGatewayV2::Api	BodyS3Location
AWS::AppSync:FunctionConfiguration	CodeS3Location RequestMappingTemplateS3Location ResponseMappingTemplateS3Location
AWS::AppSync::GraphQLSchema	DefinitionS3Location
AWS::AppSync::Resolver	CodeS3Location RequestMappingTemplateS3Location ResponseMappingTemplateS3Location
AWS::CloudFormation::ModuleVersion	ModulePackage
AWS::CloudFormation::ResourceVersion	SchemaHandlerPackage
AWS::ECR::Repository	RepositoryName
AWS::ElasticBeanstalk::ApplicationVersion	SourceBundle

資源	屬性
AWS::Glue::Job	Command.ScriptLocation
AWS::Lambda::Function	Code Code.ImageUri
AWS::Lambda::LayerVersion	Content
AWS::Serverless::Api	DefinitionUri
AWS::Serverless::Function	CodeUri ImageUri
AWS::Serverless::GraphQLApi	SchemaUri Function.CodeUri Resolver.CodeUri
AWS::Serverless::HttpApi	DefinitionUri
AWS::Serverless::LayerVersion	ContentUri
AWS::Serverless::StateMachine	DefinitionUri
AWS::StepFunctions::StateMachine	DefinitionS3Location

AWS CloudFormation 巨集

支援使用AWS::Include轉換巨集參照的檔案。

進一步了解

若要進一步瞭解AWS::Include轉換，請參閱AWS CloudFormation 使用指南中的 AWS::Include 轉換。

若要查看在範 AWS SAM 本中使用AWS::Include轉換的範例，請參閱無伺服器 Land 上的 API Gateway HTTP API 至 SQS 模式。

使用同步sam sync到簡介 AWS 雲端

指 AWS Serverless Application Model 令行介面 (AWS SAMCLI) sam sync 指令提供可快速將本機應用程式變更同步至的選項 AWS 雲端。sam sync在開發應用程式時使用以下目的：

1. 自動偵測並將本機變更同步至 AWS 雲端.
2. 自訂要將哪些本機變更同步至 AWS 雲端.
3. 在雲端準備您的應用程式以進行測試和驗證。

您可以使用sam sync建立快速開發工作流程，縮短將本機變更同步至雲端以進行測試和驗證所需的時間。

> **Note**
>
> 建議在開發環境中使用此sam sync命令。對於生產環境，我們建議使用sam deploy或設定持續整合與傳遞 (CI/CD) 管線。如需進一步了解，請參閱部署您的應用程式和資源 AWS SAM。

該sam sync命令是的一部分 AWS SAM Accelerate。 AWS SAM Accelerate提供的工具可用來加速在中開發和測試無伺服器應用程式的體驗。 AWS 雲端

主題

- 自動偵測並將本機變更同步至 AWS 雲端
- 自訂要將哪些本機變更同步至 AWS 雲端
- 在雲端準備您的應用程式以進行測試和驗證
- sam 同步指令的選項
- 故障診斷
- 範例
- 進一步了解

自動偵測並將本機變更同步至 AWS 雲端

`sam sync`使用`--watch`選項運行以開始將應用程序同步到 AWS 雲端. 這會執行以下操作：

1. 建置您的應用程式 — 此程序類似於使用`sam build`指令。
2. 部署您的應用程式 — AWS SAMCLI 將您的應用程式部署到 AWS CloudFormation 使用預設設定。會使用下列預設值：
   a. AWS 在您的`.aws`使用者資料夾中找到認證和一般組態設定。
   b. 應用程式`samconfig.toml`檔案中的應用程式部署設定。

   如果找不到預設值， AWS SAMCLI會通知您並結束同步處理程序。
3. 注意本地變化 — AWS SAMCLI 仍在運行並監視應用程序的本地更改。這是該`--watch`選項提供的內容。

   依預設，此選項可能會開啟。如需預設值，請參閱應用程式的`samconfig.toml`檔案。以下是範例檔案：

   ```
   ...
   [default.sync]
   [default.sync.parameters]
   watch = true
   ...
   ```
4. 將本機變更同步至 AWS 雲端 — 當您進行本機變更時，會 AWS 雲端 透過最快速的方法 AWS SAMCLI偵測這些變更並同步至。視變更類型而定，可能會發生下列情況：
   a. 如果您更新的資源支援 AWS 服務 API，則 AWS SAMCLI會使用它來部署您的變更。這會導致快速同步以更新中的資源 AWS 雲端。
   b. 如果您更新的資源不支援 AWS 服務 API，則 AWS SAMCLI會執行 AWS CloudFormation 部署。這會更新您在中的整個應用程式 AWS 雲端。雖然沒有那麼快，但它確實可以防止您手動初始化部署。

由於該`sam sync`命令會自動更新中的應用程式 AWS 雲端，因此建議僅用於開發環境。當您執行時`sam sync`，系統會要求您確認：

```
**The sync command should only be used against a development stack**.

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

```
Enter Y to proceed with the command, or enter N to cancel:
 [Y/n]: ENTER
```

自訂要將哪些本機變更同步至 AWS 雲端

提供自訂要同步到的本機變更的選項 AWS 雲端。這可以加快在雲中查看本地更改以進行測試和驗證所需的時間。

例如，提供僅同步程式碼變更的`--code`選項，例如 AWS Lambda 函數程式碼。在開發過程中，如果您專注於 Lambda 程式碼，這樣可以快速將您的變更導入雲端以進行測試和驗證。以下是範例：

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

若只要同步特定 Lambda 函數或層的程式碼變更，請使用`--resource-id`選項。以下是範例：

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

在雲端準備您的應用程式以進行測試和驗證

指`sam sync`令會自動尋找可用於在中更新應用程式的最快方法。 AWS 雲端這可以加快您的開發和雲端測試工作流程。透過使用 AWS 服務 API，您可以快速開發、同步處理和測試支援的資源。如需實際操作範例，請參閱單元 6-完整 AWS SAM 研討會中的 AWS SAM 加速。

sam 同步指令的選項

以下是您可以用來修改`sam sync`指令的一些主要選項。如需所有選項的清單，請參閱sam sync。

執行一次性 AWS CloudFormation 部署

使用`--no-watch`此選項可關閉自動同步。以下是範例：

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

AWS SAMCLI將執行一次性 AWS CloudFormation 部署。此`sam deploy`指令會將`sam build`和指令執行的動作群組在一起。

略過初始 AWS CloudFormation 部署

您可以自訂每次執行時`sam sync`是否需要 AWS CloudFormation 部署。

- 提供`--no-skip-deploy-sync`以在每次執行時`sam sync`都需要 AWS CloudFormation 部署。這可確保您的本機基礎結構同步到 AWS CloudFormation，防止漂移。使用此選項確實會為您的開發和測試工作流程增加額外的時間。
- 提供`--skip-deploy-sync`以使 AWS CloudFormation 部署可選。 AWS SAMCLI會將您的本機 AWS SAM 範本與已部署的 AWS CloudFormation 範本進行比較，如果未偵測到變更，則會略過初始 AWS CloudFormation 部署。將本機變更同步至時，略過 AWS CloudFormation 部署可以節省時間 AWS 雲端。

  如果未偵測到任何變更，仍 AWS SAMCLI會在下列情況下執行 AWS CloudFormation 部署：

  - 如果自上次 AWS CloudFormation 部署以來已有 7 天或更長時間。
  - 如果偵測到大量 Lambda 函數程式碼變更，讓 AWS CloudFormation 部署成為更新應用程式的最快速方法。

以下是範例：

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

從嵌套堆棧同步資源

從巢狀堆疊同步資源的步驟

1. 使用提供根堆疊`--stack-name`。
2. 使用下列格式識別巢狀堆疊中的資源：*nestedStackId/resourceId*
3. 使用提供嵌套堆棧中的資源`--resource-id`。

   以下是範例：

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

如需建立巢狀應用程式的詳細資訊，請參閱使用嵌套應用程序重複使用代碼和資源 AWS SAM。

指定要更新的特定 AWS CloudFormation 堆疊

若要指定要更新的特定 AWS CloudFormation 堆疊，請提供選`--stack-name`項。以下是範例：

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

通過在源文件夾中構建項目來加快構建時間

對於支持的運行時和構建方法，您可以使用該--build-in-source選項直接在源文件夾中構建項目。默認情況下， AWS SAM CLI構建在臨時目錄中，其中涉及複製源代碼和項目文件。使用--build-in-source，直接在源文件夾中 AWS SAM CLI構建，通過消除將文件複製到臨時目錄的需要來加快構建過程。

如需支援的執行階段和建置方法的清單，請參閱 --build-in-source。

指定不會啟動同步的檔案和資料夾

使用此選--watch-exclude項可指定任何在更新時不會啟動同步的檔案或資料夾。如需有關此選項的詳細資訊，請參閱 --watch-exclude。

以下是排除與我們HelloWorldFunction函數關聯的package-lock.json文件的示例：

```
$ sam sync --watch --watch-exclude HelloWorldFunction=package-lock.json
```

執行此命令時， AWS SAM CLI將啟動同步處理程序。這包含下列項目：

- 執行sam build以建置您的函數，並準備應用程式以進行部署。
- 執行sam deploy以部署您的應用程式。
- 注意應用程式的變更。

當我們修改package-lock.json文件時， AWS SAM CLI不會啟動同步。當另一個檔案更新時， AWS SAM CLI將啟動同步，其中將包含package-lock.json檔案。

以下是指定子堆疊的 Lambda 函數的範例：

```
$ sam sync --watch --watch-exclude ChildStackA/MyFunction=database.sqlite3
```

故障診斷

若要疑難排解 AWS SAMCLI，請參閱AWS SAMCLI疑難排。

範例

使用山姆同步來更新你好世界應用程序

在這個例子中，我們通過初始化示例 Hello World 應用程序開始。若要進一步瞭解此應用程式，請參閱教學課程：部署 Hello World 應用程式。

執行會sam sync開始建置和部署程序。

```
$ sam sync

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

Confirm that you are synchronizing a development stack.

Enter Y to proceed with the command, or enter N to cancel:
 [Y/n]:
Queued infra sync. Waiting for in progress code syncs to complete...
Starting infra sync.
Manifest file is changed (new hash: 3298f13049d19cffaa37ca931dd4d421) or dependency
 folder (.aws-sam/deps/0663e6fe-a888-4efb-b908-e2344261e9c7) is missing for
 (HelloWorldFunction), downloading dependencies and copying/building source
Building codeuri: /Users/.../Demo/sync/sam-app/hello_world runtime: python3.9 metadata:
 {} architecture: x86_64 functions: HelloWorldFunction
Running PythonPipBuilder:CleanUp
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource

Build Succeeded

Successfully packaged artifacts and wrote output template to file /var/
folders/45/5ct135bx3fn2551_ptl5g6_80000gr/T/tmpx_5t4u3f.
Execute the following command to deploy the packaged template
sam deploy --template-file /var/folders/45/5ct135bx3fn2551_ptl5g6_80000gr/T/tmpx_5t4u3f
 --stack-name <YOUR STACK NAME>


    Deploying with following values
    ===============================
```

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

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


2023-03-17 11:17:19 - Waiting for stack create/update to complete

CloudFormation events from stack operations (refresh every 0.5 seconds)
---------------------------------------------------------------------------------------------
ResourceStatus                     ResourceType
 LogicalResourceId                  ResourceStatusReason
---------------------------------------------------------------------------------------------
CREATE_IN_PROGRESS                 AWS::CloudFormation::Stack          sam-app
                     Transformation succeeded
CREATE_IN_PROGRESS                 AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt   -
                                                                       ack
CREATE_IN_PROGRESS                 AWS::IAM::Role
 HelloWorldFunctionRole              -
CREATE_IN_PROGRESS                 AWS::IAM::Role
 HelloWorldFunctionRole              Resource creation Initiated
CREATE_IN_PROGRESS                 AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt   Resource creation Initiated
                                                                       ack
CREATE_COMPLETE                    AWS::IAM::Role
 HelloWorldFunctionRole              -
CREATE_COMPLETE                    AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt   -
                                                                       ack
CREATE_IN_PROGRESS                 AWS::Lambda::Function
 HelloWorldFunction                  -
CREATE_IN_PROGRESS                 AWS::Lambda::Function
 HelloWorldFunction                  Resource creation Initiated
CREATE_COMPLETE                    AWS::Lambda::Function
 HelloWorldFunction                  -
```

```
CREATE_IN_PROGRESS                  AWS::ApiGateway::RestApi
 ServerlessRestApi                   -
CREATE_IN_PROGRESS                  AWS::ApiGateway::RestApi
 ServerlessRestApi                   Resource creation Initiated
CREATE_COMPLETE                     AWS::ApiGateway::RestApi
 ServerlessRestApi                   -
CREATE_IN_PROGRESS                  AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment47fc2d   -
                                                                    5f9d
CREATE_IN_PROGRESS                  AWS::Lambda::Permission
 HelloWorldFunctionHelloWorldPermi   -
                                                                    ssionProd
CREATE_IN_PROGRESS                  AWS::Lambda::Permission
 HelloWorldFunctionHelloWorldPermi   Resource creation Initiated
                                                                    ssionProd
CREATE_IN_PROGRESS                  AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment47fc2d   Resource creation Initiated
                                                                    5f9d
CREATE_COMPLETE                     AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment47fc2d   -
                                                                    5f9d
CREATE_IN_PROGRESS                  AWS::ApiGateway::Stage
 ServerlessRestApiProdStage          -
CREATE_IN_PROGRESS                  AWS::ApiGateway::Stage
 ServerlessRestApiProdStage          Resource creation Initiated
CREATE_COMPLETE                     AWS::ApiGateway::Stage
 ServerlessRestApiProdStage          -
CREATE_COMPLETE                     AWS::Lambda::Permission
 HelloWorldFunctionHelloWorldPermi   -
                                                                    ssionProd
CREATE_COMPLETE                     AWS::CloudFormation::Stack      sam-app
                      -
-----------------------------------------------------------------------------------------

CloudFormation outputs from deployed stack
-----------------------------------------------------------------------------------------
Outputs
-----------------------------------------------------------------------------------------
Key                 HelloWorldFunctionIamRole
Description         Implicit IAM Role created for Hello World function
Value               arn:aws:iam::012345678910:role/sam-app-HelloWorldFunctionRole-
BUFVMO2PJIYF

Key                 HelloWorldApi
```

```
Description         API Gateway endpoint URL for Prod stage for Hello World function
Value               https://pcrx5gdaof.execute-api.us-west-2.amazonaws.com/Prod/hello/

Key                 HelloWorldFunction
Description         Hello World Lambda Function ARN
Value               arn:aws:lambda:us-west-2:012345678910:function:sam-app-
HelloWorldFunction-2PlN6TPTQoco
---------------------------------------------------------------------------------------------------
Stack creation succeeded. Sync infra completed.

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

部署完成後，我們會修改程HelloWorldFunction式碼。會 AWS SAMCLI偵測到此變更，並將我們的應用程式同步到 AWS 雲端. 由於 AWS Lambda 支援 AWS 服務 API，因此會執行快速同步。

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

接下來，我們在應用程序的 AWS SAM 模板中修改我們的 API 端點。我們變更/hello為/helloworld。

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

由於 Amazon API Gateway 資源不支援 AWS 服務 API，因此會 AWS SAMCLI自動執行 AWS CloudFormation 部署。下面是一個示例輸出：

```
Queued infra sync. Waiting for in progress code syncs to complete...
Starting infra sync.
Manifest is not changed for (HelloWorldFunction), running incremental build
Building codeuri: /Users/.../Demo/sync/sam-app/hello_world runtime: python3.9 metadata:
 {} architecture: x86_64 functions: HelloWorldFunction
Running PythonPipBuilder:CopySource

Build Succeeded

Successfully packaged artifacts and wrote output template to file /var/
folders/45/5ct135bx3fn2551_ptl5g6_80000gr/T/tmpuabo0jb9.
Execute the following command to deploy the packaged template
sam deploy --template-file /var/folders/45/5ct135bx3fn2551_ptl5g6_80000gr/T/tmpuabo0jb9
 --stack-name <YOUR STACK NAME>


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

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


2023-03-17 14:41:18 - Waiting for stack create/update to complete

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

```
UPDATE_IN_PROGRESS                  AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt   -
                                                                    ack
UPDATE_COMPLETE                     AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt   -
                                                                    ack
UPDATE_IN_PROGRESS                  AWS::ApiGateway::RestApi
 ServerlessRestApi                   -
UPDATE_COMPLETE                     AWS::ApiGateway::RestApi
 ServerlessRestApi                   -
CREATE_IN_PROGRESS                  AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment8cf30e   -
                                                                    d3cd
UPDATE_IN_PROGRESS                  AWS::Lambda::Permission
 HelloWorldFunctionHelloWorldPermi   Requested update requires the
                                                                    ssionProd
                     creation of a new physical

                     resource; hence creating one.
UPDATE_IN_PROGRESS                  AWS::Lambda::Permission
 HelloWorldFunctionHelloWorldPermi   Resource creation Initiated
                                                                    ssionProd
CREATE_IN_PROGRESS                  AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment8cf30e   Resource creation Initiated
                                                                    d3cd
CREATE_COMPLETE                     AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment8cf30e   -
                                                                    d3cd
UPDATE_IN_PROGRESS                  AWS::ApiGateway::Stage
 ServerlessRestApiProdStage          -
UPDATE_COMPLETE                     AWS::ApiGateway::Stage
 ServerlessRestApiProdStage          -
UPDATE_COMPLETE                     AWS::Lambda::Permission
 HelloWorldFunctionHelloWorldPermi   -
                                                                    ssionProd
UPDATE_COMPLETE_CLEANUP_IN_PROGRE   AWS::CloudFormation::Stack      sam-app
                     -
SS
DELETE_IN_PROGRESS                  AWS::Lambda::Permission
 HelloWorldFunctionHelloWorldPermi   -
                                                                    ssionProd
DELETE_IN_PROGRESS                  AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment47fc2d   -
                                                                    5f9d
```

```
DELETE_COMPLETE                     AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment47fc2d   -
                                                                    5f9d
UPDATE_COMPLETE                     AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt   -
                                                                    ack
DELETE_COMPLETE                     AWS::Lambda::Permission
 HelloWorldFunctionHelloWorldPermi   -
                                                                    ssionProd
UPDATE_COMPLETE                     AWS::CloudFormation::Stack          sam-app
                      -
-----------------------------------------------------------------------------------------------

CloudFormation outputs from deployed stack
-----------------------------------------------------------------------------------------------
Outputs
-----------------------------------------------------------------------------------------------
Key                 HelloWorldFunctionIamRole
Description         Implicit IAM Role created for Hello World function
Value               arn:aws:iam::012345678910:role/sam-app-HelloWorldFunctionRole-
BUFVMO2PJIYF

Key                 HelloWorldApi
Description         API Gateway endpoint URL for Prod stage for Hello World function
Value               https://pcrx5gdaof.execute-api.us-west-2.amazonaws.com/Prod/hello/

Key                 HelloWorldFunction
Description         Hello World Lambda Function ARN
Value               arn:aws:lambda:us-west-2:012345678910:function:sam-app-
HelloWorldFunction-2PlN6TPTQoco
-----------------------------------------------------------------------------------------------


Stack update succeeded. Sync infra completed.

Infra sync completed.
```

進一步了解

如需所有`sam sync`選項的描述，請參閱sam sync。

監控您的無伺服器應用程式 AWS SAM

部署無伺服器應用程式後，您可以對其進行監控，以提供有關其作業的深入解析並偵測異常情況，這有助於進行疑難排解。本節提供監控無伺服器應用程式的詳細資訊。這包括如何設定 Amazon 在偵測 CloudWatch 到異常時通知您的資訊。它也提供使用記錄的相關資訊，包括錯誤反白顯示，以及檢視、篩選、擷取和追蹤記錄檔的提示。

主題

使用應用程式洞察來監控無伺服器 CloudWatch 應用

Amazon CloudWatch 應用程式深入解析可協助您監控應用程式中的 AWS 資源，以協助識別潛在問題。它可以分析 AWS 資源數據以查找問題跡象，並構建自動化儀表板以將其視覺化。您可以設定應用 CloudWatch 程式深入解析，以搭配 AWS Serverless Application Model (AWS SAM) 應用程式使用。若要進一步了解 CloudWatch 應用程式洞察，請參閱 Amazon CloudWatch 使用者指南中的 Amazon CloudWatch 應用程式深入解析。

主題

設定 CloudWatch 應用程式見解 AWS SAM

透過 AWS SAM 命令列介面 (AWS SAMCLI) 或透過 AWS SAM 範本，為您的 AWS SAM 應用程式設 CloudWatch 定應用程式深入解析。

透過設定 AWS SAMCLI

使用初始化應用程式時sam init，請透過互動式流 CloudWatch 程或使用--application-insights選項啟動應用程式深入解析。

若要透過 AWS SAMCLI互動式流 CloudWatch 程啟用應用程式見解，請在出現提示**y**時輸入

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

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

若要使用--application-insights選項啟 CloudWatch 用應用程式深入解析，請執行下列動作。

```
sam init --application-insights
```

若要瞭解有關使用sam init指令的更多資訊，請參閱sam init。

透過 AWS SAM 範本設定

透過在 AWS SAM 範本中定義AWS::ResourceGroups::Group和AWS::ApplicationInsights::Application資源來啟動 CloudWatch 應用程式深入解析。

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  ApplicationResourceGroup:
    Type: AWS::ResourceGroups::Group
    Properties:
      Name:
        Fn::Join:
        - ''
        - - ApplicationInsights-SAM-
        - Ref: AWS::StackName
      ResourceQuery:
        Type: CLOUDFORMATION_STACK_1_0
  ApplicationInsightsMonitoring:
    Type: AWS::ApplicationInsights::Application
    Properties:
      ResourceGroupName:
        Fn::Join:
          - ''
          - - ApplicationInsights-SAM-
          - Ref: AWS::StackName
        AutoConfigurationEnabled: 'true'
    DependsOn: ApplicationResourceGroup
```

- AWS::ResourceGroups::Group— 建立群組來組織您的 AWS 資源，以便一次管理和自動化大量資源上的工作。您可以在此建立資源群組，以與應用 CloudWatch 程式深

入解析搭配使用。如需此資源類型的詳細資訊，請參閱《AWS CloudFormation 使用指南》*AWS::ResourceGroups::Group*中的〈〉。

- `AWS::ApplicationInsights::Application`— 設定資源群組的 CloudWatch 應用程式深入解析。如需此資源類型的詳細資訊，請參閱《AWS CloudFormation 使用指南》*AWS::ApplicationInsights::Application*中的〈〉。

這兩種資源都會 AWS CloudFormation 在應用程式部署時自動傳遞至。您可以使用 AWS SAM 範本中的 AWS CloudFormation 語法進一步設定 CloudWatch 應用程式深入解析。如需詳細資訊，請參閱 Amazon 使用 CloudWatch 者指南中的使用 AWS CloudFormation 範本。

使用指sam init --application-insights令時，這兩個資源都會自動在 AWS SAM 範本中產生。下面是一個生成的模板的例子。

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: >
  sam-app-test

  Sample SAM Template for sam-app-test

# More info about Globals: https://github.com/awslabs/serverless-application-model/
blob/master/docs/globals.rst
Globals:
  Function:
    Timeout: 3
    MemorySize: 128

Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function # More info about Function Resource:
 https://github.com/awslabs/serverless-application-model/blob/master/
versions/2016-10-31.md#awsserverlessfunction
    Properties:
      CodeUri: hello_world/
      Handler: app.lambda_handler
      Runtime: python3.9
      Architectures:
      - x86_64
      Events:
        HelloWorld:
```

```
          Type: Api # More info about API Event Source: https://github.com/awslabs/
serverless-application-model/blob/master/versions/2016-10-31.md#api
          Properties:
            Path: /hello
            Method: get

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

Outputs:
  # ServerlessRestApi is an implicit API created out of Events key under
 Serverless::Function
  # Find out more about other implicit resources you can reference within SAM
  # https://github.com/awslabs/serverless-application-model/blob/master/docs/internals/
generated_resources.rst#api
  HelloWorldApi:
    Description: API Gateway endpoint URL for Prod stage for Hello World function
    Value: !Sub "https://${ServerlessRestApi}.execute-api.${AWS::Region}.amazonaws.com/
Prod/hello/"
  HelloWorldFunction:
    Description: Hello World Lambda Function ARN
    Value: !GetAtt HelloWorldFunction.Arn
  HelloWorldFunctionIamRole:
    Description: Implicit IAM Role created for Hello World function
    Value: !GetAtt HelloWorldFunctionRole.Arn
```

後續步驟

設定 CloudWatch 應用程式深入解析之後，可用sam build來建置您的應用程sam deploy式和部署應用程式。所有 CloudWatch 應用程式見解支援的資源都會設定進行監視。

- 如需支援資源的清單，請參閱 Amazon CloudWatch 使用者指南中的支援日誌和指標。
- 若要了解如何存取 CloudWatch 應用程式洞見，請參閱 Amazon CloudWatch 使用者指南中的存取 CloudWatch 應用程式深入解析。

使用記錄

為了簡化疑難排解， AWS SAMCLI有一個名為的命令`sam logs`。此命令可讓您從命令列擷取 Lambda 函數產生的記錄。

> **Note**
>
> 該`sam logs`命令適用於所有 AWS Lambda 功能，而不僅僅是您使用部署的功能 AWS SAM。

通 AWS CloudFormation 過堆棧獲取日誌

當您的函數是 AWS CloudFormation 堆棧的一部分時，您可以使用函數的邏輯 ID 來獲取日誌：

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

通過 Lambda 函數名稱獲取日誌

或者，您可以使用函數的名稱獲取日誌：

```
sam logs -n mystack-HelloWorldFunction-1FJ8PD
```

拖尾日誌

添加選`--tail`項以等待新日誌並在到達時查看它們。這在部署期間或疑難排解生產問題時很有幫助。

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

檢視特定時間範圍的記錄

您可以使用-s和-e選項來檢視特定時間範圍的記錄檔：

```
sam logs -n HelloWorldFunction --stack-name mystack -s '10min ago' -e '2min ago'
```

過濾記錄檔

使用此選--filter項可快速尋找符合日誌事件中字詞、片語或值的記錄檔：

```
sam logs -n HelloWorldFunction --stack-name mystack --filter "error"
```

在輸出中， AWS SAMCLI強調所有出現的單詞「error」，以便您可以輕鬆地在日誌輸出中找到 filter 關鍵字。

亮顯錯誤

當 Lambda 函數當機或逾時時，會以紅色 AWS SAMCLI反白顯示逾時訊息。這有助於您輕鬆找到在巨大的日誌輸出流中超時的特定執行。

漂亮的印刷

如果您的日誌消息打印 JSON 字符串，則 AWS SAMCLI會自動打印 JSON 以幫助您直觀地解析和理解 JSON。

AWS SAM 參考

本節包含 AWS SAM 參考材料。這包括 AWS SAMCLI參考資料，例如 AWS SAMCLI指令的參考資訊和其他 AWS SAMCLI資訊，例如組態、版本控制和疑難排解資訊。此外， AWS SAM 本節還包括有關 AWS SAM 規格和範本的參考資訊，例如連接器、映像儲存庫和部署的參考資訊。

AWS SAM 規格和模 AWS SAM 板

該 AWS SAM 規範是 Apache 2.0 許可證下的開源規範。 AWS SAM 規格的目前版本可在中找到項 AWS SAM 目和 AWS SAM 模板。 AWS SAM 規格隨附簡化的簡短語法，可用來定義無伺服器應用程式的函數、事件、API、組態和權限。

您可以透過 AWS SAM 應用程式專案目錄 (執行sam init命令時建立的資料夾和檔案) 與 AWS SAM 規格互動。該目錄包括 AWS SAM 模板，這是定義 AWS 資源的重要文件。該 AWS SAM 模板是模 AWS CloudFormation 板的擴展。如需 AWS CloudFormation 範本的完整參考資料，請參閱《AWS CloudFormation 使用指南》中的〈範本參考〉。

AWS SAMCLI指令參考

命 AWS Serverless Application Model 令列介面 (AWS SAMCLI) 是命令列工具，可搭配 AWS SAM 範本和支援的第三方整合使用，以建置和執行無伺服器應用程式。

您可以使用這些 AWS SAMCLI命令來開發、測試和部署無伺服器應用程式到 AWS 雲端. 以下是 AWS SAMCLI指令的一些範例：

- `sam init`— 如果您是初次 AWS SAMCLI使用的使用者，您可以執行不含任何參數的`sam init`命令來建立 Hello World 應用程式。此命令會以您選擇的語言產生預先設定的範 AWS SAM 本和應用程式範例程式碼。
- `sam local invoke`和 `sam local start-api` — 使用這些命令在本機測試您的應用程式程式碼，然後再將它部署到 AWS 雲端.
- `sam logs`— 使用此命令可擷取 Lambda 函數產生的記錄。這可以協助您在將應用程式部署到 AWS 雲端.
- `sam package`— 使用此命令將您的應用程式程式碼和相依性捆綁到部署套件中。您需要部署套件才能將應用程式上傳至 AWS 雲端.
- `sam deploy`— 使用此命令將無伺服器應用程式部署到 AWS 雲端. 它會建立資 AWS 源，並設定 AWS SAM 範本中定義的權限和其他組態。

如需有關安裝的指示 AWS SAMCLI，請參閱安裝 AWS SAMCLI。

AWS SAM 策略範本

使用時 AWS SAM，您可以從政策範本清單中選擇，將 AWS Lambda 函數的權限限定為應用程式使用的資源。

主題

- 項 AWS SAM 目和 AWS SAM 模板
- AWS SAMCLI指令參考
- AWS SAMCLI配置文件
- AWS SAM 連接器參考
- AWS SAM策略範本
- 映像儲存庫
- 遙測中的 AWS SAMCLI
- 在 AWS SAM 範本中設定和管理資源存取

AWS SAMCLI指令參考

本節包括有關 AWS SAMCLI指令的參考資訊。其中包括使用方式的詳細資訊、每個命令可用之不同選項的完整清單，以及其他資訊。適用時，其他資訊包括引數、環境變數和事件等詳細資訊。有關詳細信息，請參見每個命 如需有關安裝的指示 AWS SAMCLI，請參閱安裝 AWS SAMCLI。

主題

- sam build
- sam delete
- sam deploy
- sam init
- sam list
- sam local generate-event
- sam local invoke

- sam local start-api
- sam local start-lambda
- sam logs
- sam package
- sam pipeline bootstrap
- sam pipeline init
- sam publish
- sam remote invoke
- sam remote test-event
- sam sync
- sam traces
- sam validate

sam build

本頁提供「 AWS Serverless Application Model 指令行介面」(AWS SAMCLI) `sam build` 指令的參考資訊。

- 如需「」的簡介 AWS SAMCLI，請參閱什麼是 AWS SAMCLI？。
- 如需使用 AWS SAMCLI`sam build`指令的文件，請參閱使用sam build指令建立簡介。

此命`sam build`令會為開發人員工作流程中的後續步驟準備應用程式，例如本機測試或部署到 AWS 雲端.

用量

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

引數

資源 ID

選用。指示建 AWS SAM 立在AWS SAM 範本中宣告的單一資源。指定資源的構建工件將是唯一可用於工作流程中後續命令的構建工件，即`sam package`和`sam deploy`。

選項

`--base-dir, -s` *DIRECTORY*

解析與此目錄相關的函數或圖層原始程式碼的相對路徑。如果您想要變更原始程式碼資料夾的相對路徑的解析方式，請使用此選項。依預設，相對路徑會根據 AWS SAM 樣板的位置進行解析。

除了您正在建置的根應用程式或堆疊中的資源之外，此選項也會套用巢狀應用程式或堆疊。

此選項適用於下列資源類型和屬性：

- 資源類型:AWS::Serverless::Function屬性:CodeUri
- 資源類型:AWS::Serverless::Function資源屬性:Metadata項目:DockerContext
- 資源類型:AWS::Serverless::LayerVersion屬性:ContentUri
- 資源類型:AWS::Lambda::Function屬性:Code
- 資源類型:AWS::Lambda::LayerVersion屬性:Content

`--beta-features | --no-beta-features`

允許或拒絕測試版功能。

`--build-dir, -b` *DIRECTORY*

儲存已建構成品之目錄的路徑。此目錄及其所有內容都會使用此選項移除。

`--build-image` *TEXT*

您要為組建提取的容器映像檔的 URI。默認情況下， AWS SAM 從 Amazon ECR 公共提取容器映像。使用此選項可從其他位置提取影像。

您可以多次指定此選項。此選項的每個執行個體都可以使用字串或索引鍵值配對。如果您指定字串，它就是應用程式中所有資源使用的容器映像檔的 URI。例如 `sam build --use-container --build-image amazon/aws-sam-cli-build-image-python3.8`。如果您指定索引鍵值配對，則索引鍵是資源名稱，而該值是要用於該資源的容器映像檔的 URI。例如：`sam build --use-container --build-image Function1=amazon/aws-sam-cli-build-image-python3.8`。使用鍵值配對，您可以為不同的資源指定不同的容器映像檔。

此選項僅在指定選`--use-container`項時適用，否則將導致錯誤。

`--build-in-source | --no-build-in-source`

提供`--build-in-source`直接在源文件夾中構建項目。

--build-in-source此選項支援下列執行階段和建置方法：

- 執行階段 — sam init --runtime 選項支援的任何Node.js執行階段。
- 建置方法 — Makefile、esbuild.

此選--build-in-source項與下列選項不相容：

- --hook-name
- --use-container

預設：--no-build-in-source

--cached | --no-cached

啟用或停用快取的組建。使用此選項可重複使用先前組建未變更的組建成品。 AWS SAM 評估您是否已更改項目目錄中的任何文件。根據預設，不會快取組建。如果調用該--no-cached選項，它將覆蓋 samcofig.toml 中的cached = true設置。

> **Note**
>
> AWS SAM 不會評估您是否已更改項目所依賴的第三方模塊，以及尚未提供特定版本的地方。例如，如果您的 Python 函數包含一個帶有條目的requirements.txt文件requests=1.x，並且最新的請求模塊版本從更改1.1為1.2，那麼在運行非緩存構建之前 AWS SAM 不會提取最新版本。

--cache-dir

指定快取人工因素的儲存--cached目錄。預設快取目錄為.aws-sam/cache。

--config-env *TEXT*

指定組態檔案中要使用的預設參數值的環境名稱。預設值為「預設值」。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

--config-file *PATH*

包含要使用的預設參數值的組態檔案的路徑和檔案名稱。在專案目錄的根目錄中，預設值為samconfig.toml ""。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

--container-env-var, -e *TEXT*

要傳遞給組建容器的環境變數。您可以多次指定此選項。這個選項的每個執行個體都有一個機碼-值配對，其中索引鍵是資源和環境變數，而值是環境變數的值。例如：--

container-env-var Function1.GITHUB_TOKEN=TOKEN1 --container-env-var Function2.GITHUB_TOKEN=TOKEN2。

此選項僅在指定選--use-container項時適用，否則將導致錯誤。

--container-env-var-file, -ef *PATH*

JSON 檔案的路徑和檔案名稱，其中包含容器環境變數的值。如需容器環境變數檔案的詳細資訊，請參閱容器環境變量文件。

此選項僅在指定選--use-container項時適用，否則將導致錯誤。

--debug

開啟偵錯記錄以列印 AWS SAMCLI產生的偵錯訊息，並顯示時間戳記。

--docker-network *TEXT*

指定 Lambda Docker 容器應連線到的現有Docker網路名稱或 ID，以及預設橋接網路。如果未指定，Lambda 容器只會連線至預設的橋接Docker網路。

--exclude, -x

要從中排除的資源名稱sam build。例如，如果您的範本包含Function1Function2、和Function3且您執行sam build --exclude Function2，則只會建置Function1且Function3將會建置。

--help

顯示此訊息並結束。

--hook-name *TEXT*

用來擴充 AWS SAMCLI功能的掛接名稱。

接受的值：terraform。

--manifest , -m *PATH*

要使用的自訂相依性資訊清單檔案 (例如 package.json) 的路徑，而非預設值。

--parallel

啟用 parallel 組建。使用此選項可 parallel 建置 AWS SAM 範本的函數和圖層。默認情況下，功能和圖層是按順序構建的。

`--parameter-overrides`

(選擇性) 包含編碼為索引鍵值配對之 AWS CloudFormation 參數覆寫的字串。使用與 AWS Command Line Interface (AWS CLI) 相同的格式。例如：'ParameterKey=KeyPairName，ParameterValue= MyKey ParameterKey =InstanceType，ParameterValue=t1.micro'。此選項與不相容--hook-name。

`--profile` *TEXT*

從您的認證檔案取得 AWS 認證的特定設定檔。

`--region` *TEXT*

AWS 區域 要部署到. 例如 us-east-1。

`--save-params`

將您在命令列中提供的參數儲存到 AWS SAM 組態檔案中。

`--skip-prepare-infra`

如果未進行基礎結構變更，則略過準備階段。搭配--hook-name選項使用。

`--skip-pull-image`

指定命令是否應略過向下拉 Lambda 執行階段的最新 Docker 映像檔。

`--template-file, --template, -t` *PATH*

樣板檔案的路徑和檔 AWS SAM 案名稱[default: template.[yaml|yml]]。此選項與不相容--hook-name。

`--terraform-project-root-path`

包含配置文件或函數源代碼的頂級目錄的Terraform相對或絕對路徑。如果這些檔案位於包含Terraform根模組的目錄之外，請使用此選項來指定其絕對或相對路徑。此選項需要--hook-name將設定為terraform。

`--use-container, -u`

如果您的函數依賴於具有原生編譯依賴關係的軟件包，請使用此選項在類似 Lambda 的 Docker 容器中構建函數。

sam delete

本頁提供「 AWS Serverless Application Model 指令行介面」(AWS SAMCLI) `sam delete` 指令的參考資訊。

如需「」的簡介 AWS SAMCLI，請參閱什麼是 AWS SAMCLI？。

該sam delete命令會刪除 AWS SAM 應用程式，刪除 AWS CloudFormation 堆疊、封裝並部署到 Amazon S3 和 Amazon ECR 的成品，以及 AWS SAM 範本檔案。

此命令還會檢查是否已部署 Amazon ECR 配套堆疊，如果有的話會提示使用者刪除該堆疊和 Amazon ECR 儲存庫。如果--no-prompts已指定，則依預設會刪除隨附堆疊和 Amazon ECR 儲存庫。

用量

```
$ sam delete <options>
```

選項

`--config-env` *TEXT*

指定組態檔案中要使用的預設參數值的環境名稱。預設值為 default。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--config-file` *PATH*

包含要使用的預設參數值的組態檔案的路徑和檔案名稱。預設值samconfig.toml位於專案目錄的根目錄中。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--debug`

開啟偵錯記錄以列印 AWS SAMCLI產生的偵錯訊息，並顯示時間戳記。

`--help`

顯示此訊息並結束。

`--no-prompts`

指定此選項可在非互動模式下 AWS SAM 進行操作。堆疊名稱必須與--stack-name選項一起提供，或在組態toml檔案中提供。

`--profile` *TEXT*

從您的認證檔案取得 AWS 認證的特定設定檔。

`--region` *TEXT*

要部署的 AWS 區域。例如 us-east-1。

`--s3-bucket`

您要刪除的 Amazon S3 儲存貯體的路徑。

`--s3-prefix`

您要刪除的 Amazon S3 儲存貯體的前置詞。

`--save-params`

將您在命令列中提供的參數儲存到 AWS SAM 組態檔案中。

`--stack-name` *TEXT*

您要刪除的 AWS CloudFormation 堆疊名稱。

sam deploy

本頁提供「 AWS Serverless Application Model 指令行介面」(AWS SAMCLI) `sam deploy` 指令的參考資訊。

- 如需「」的簡介 AWS SAMCLI，請參閱什麼是 AWS SAMCLI？。
- 如需使用 AWS SAMCLI`sam deploy`指令的文件，請參閱使用sam deploy指令部署簡介。

該`sam deploy`命令將應用程序部署到 AWS 雲端 使用 AWS CloudFormation。

用量

```
$ <environment variables> sam deploy <options>
```

環境變數

`SAM_CLI_POLL_DELAY`

將`SAM_CLI_POLL_DELAY`環境變數設定為秒，以設定 AWS SAM CLI 檢查 AWS CloudFormation 堆疊狀態的頻率，這在查看節流時非常有用。 AWS CloudFormation這個 env 變量用於輪詢 `describe_stack` API 調用，這是在運行時進行`sam deploy`。

以下是此變數的範例：

```
$ SAM_CLI_POLL_DELAY=5 sam deploy
```

選項

`--capabilities` *`LIST`*

您必須指定以允許 AWS CloudFormation 建立特定堆疊的功能清單。某些堆疊範本可能包含影響您的許可的資源 AWS 帳戶，例如建立新的 AWS Identity and Access Management (IAM) 使用者。對於這些堆疊，您必須指定此選項來明確認其功能。唯一有效的值為 CAPABILITY_IAM 和 CAPABILITY_NAMED_IAM。如果您有 IAM 資源，則可以指定任一功能。如果您擁有具有自訂名稱的 IAM 資源，則必須指定CAPABILITY_NAMED_IAM。如果未指定此選項，則作業會傳回錯InsufficientCapabilities誤。

`--config-env` *`TEXT`*

指定組態檔案中要使用的預設參數值的環境名稱。預設值為 default。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--config-file` *`PATH`*

包含要使用的預設參數值的組態檔案的路徑和檔案名稱。預設值samconfig.toml位於專案目錄的根目錄中。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--confirm-changeset | --no-confirm-changeset`

提示確認是否部 AWS SAMCLI署計算的變更集。

`--debug`

開啟偵錯記錄以列印 AWS SAMCLI產生的偵錯訊息，並顯示時間戳記。

`--disable-rollback | --no-disable-rollback`

指定是否在部署期間發生錯誤時復原 AWS CloudFormation 堆疊。根據預設，如果在部署期間發生錯誤，您的 AWS CloudFormation 堆疊會回復到上一個穩定狀態。如果您指定--disable-rollback並且在部署期間發生錯誤，則在錯誤發生之前建立或更新的資源不會復原。

`--fail-on-empty-changeset | --no-fail-on-empty-changeset`

指定如果沒有要對堆疊進行任何變更，是否傳回非零結束代碼。默認行為是返回一個非零的退出代碼。

`--force-upload`

指定此選項以上傳成品，即使它們符合 Amazon S3 儲存貯體中的現有成品。相符的人工因素會被覆寫

`--guided, -g`

指定此選項可讓 AWS SAMCLI使用提示引導您完成部署。

`--help`

顯示此訊息並退出。

`--image-repositories` *TEXT*

將函數對應至其 Amazon ECR 儲存庫 URI。通過它們的邏輯 ID 引用函數。以下是範例：

```
$ sam deploy --image-repositories Function1=123456789012.dkr.ecr.us-
east-1.amazonaws.com/my-repo
```

您可以在單一指令中多次指定此選項。

`--image-repository` *TEXT*

此命令上傳函數映像的 Amazon ECR 儲存庫的名稱。使用Image封裝類型宣告的函數需要此選項。

`--kms-key-id` *TEXT*

用於加密 Amazon S3 儲存貯體中靜態成品的 AWS Key Management Service (AWS KMS) 金鑰識別碼。如果未指定此選項，則 AWS SAM 使用 Amazon S3 受管加密金鑰。

`--metadata`

要貼附至範本中參考之所有人工因素的詮釋資料對映。

`--no-execute-changeset`

指出是否套用變更集。如果您要在套用變更集之前檢視堆疊變更，請指定此選項。此命令會建立 AWS CloudFormation 變更集，然後結束而不套用變更集。若要套用變更集，請在不使用此選項的情況下執行相同的命令。

`--no-progressbar`

將成品上傳到 Amazon S3 時，請勿顯示進度列。

`--notification-arns` *LIST*

與堆疊 AWS CloudFormation 相關聯的亞馬遜簡單通知服務 (Amazon SNS) 主題 ARN 清單。

`--on-failure [ROLLBACK | DELETE | DO_NOTHING]`

指定堆疊建立失敗時要採取的動作。

以下是可用的選項：

- ROLLBACK— 將堆疊回復至先前已知的良好狀態。
- DELETE— 將堆疊回復至先前已知的良好狀態 (如果存在)。否則，會刪除堆疊。
- DO_NOTHING-既不回滾也不刪除堆棧。效果與的效果相同`--disable-rollback`。

預設行為是 ROLLBACK。

> **Note**
>
> 您可以指定`--disable-rollback`選項或選`--on-failure`項，但不能同時指定兩者。

`--parameter-overrides`

包含編碼為索引鍵值配對之 AWS CloudFormation 參數覆寫的字串。使用與 AWS Command Line Interface (AWS CLI) 相同的格式。例如 `ParameterKey=ParameterValue InstanceType=t1.micro`。

`--profile` *TEXT*

從您的認證檔案取得 AWS 認證的特定設定檔。

`--region` *TEXT*

AWS 區域 要部署到. 例如 us-east-1。

`--resolve-image-repos`

自動建立 Amazon ECR 儲存庫，以用於封裝和部署非引導式部署。此選項僅適用於`PackageType: Image`指定的函數和圖層。如果您指定`--guided`選項，則 AWS SAMCLI忽略`--resolve-image-repos`。

> **Note**
>
> 如果使用此選項 AWS SAM 自動為函數或圖層建立任何 Amazon ECR 儲存庫，而您稍後從 AWS SAM 範本中刪除這些函數或圖層，則會自動刪除對應的 Amazon ECR 儲存庫。

`--resolve-s3`

自動建立 Amazon S3 儲存貯體，用於包裝和部署非引導式部署。如果您指定`--guided`選項，則 AWS SAM CLI 將忽略`--resolve-s3`。如果同時指定`--s3-bucket`和`--resolve-s3`選項，則會發生錯誤。

`--role-arn` *TEXT*

應用變更集時 AWS CloudFormation 假設的 IAM 角色的 Amazon 資源名稱 (ARN)。

`--s3-bucket` *TEXT*

此命令會上傳 AWS CloudFormation 範本的 Amazon S3 儲存貯體的名稱。如果您的範本大於 51,200 位元組，則需要`--s3-bucket`選項或`--resolve-s3`選項。如果同時指定`--s3-bucket`和`--resolve-s3`選項，則會發生錯誤。

`--s3-prefix` *TEXT*

新增至上傳至 Amazon S3 儲存貯體之成品名稱的前置詞。前置詞名稱是 Amazon S3 儲存貯體的路徑名稱 (資料夾名稱)。

`--save-params`

將您在命令列中提供的參數儲存到 AWS SAM 組態檔案中。

`--signing-profiles` *LIST*

用來簽署您的部署套件的簽署設定檔清單。此選項會取得索引鍵值配對的清單，其中索引鍵是要簽署的函數或層的名稱，而值是簽署設定檔，而選擇性的設定檔擁有者則以分隔。`:`例如 `FunctionNameToSign=SigningProfileName1 LayerNameToSign=SigningProfileName2:SigningProfileOwner`。

`--stack-name` *TEXT*

(必要) 您要部署到的 AWS CloudFormation 堆疊名稱。如果您指定既有堆疊，則指令會更新堆疊。如果您指定新堆疊，則指令會建立它。

`--tags` *LIST*

要與建立或更新之堆疊相關聯的標籤清單。 AWS CloudFormation 還將這些標籤傳播到支持它的堆棧中的資源。

`--template-file, --template, -t` *PATH*

AWS SAM 樣板所在的路徑和檔案名稱。

> **Note**
>
> 如果您指定此選項，則只 AWS SAM 會部署範本及其指向的本機資源。

`--use-json`

為 AWS CloudFormation 範本輸出 JSON。預設輸出為 YAML。

sam init

本頁提供「 AWS Serverless Application Model 指令行介面」(AWS SAMCLI) `sam init` 指令的參考資訊。

- 如需「」的簡介 AWS SAMCLI，請參閱什麼是 AWS SAMCLI？。
- 如需使用 AWS SAMCLI`sam init`指令的文件，請參閱使用sam init指令建立您的應用程式。

此指`sam init`令提供初始化新無伺服器應用程式的選項。

用量

```
$ sam init <options>
```

選項

`--app-template` *TEXT*

您要使用之受管理應用程式範本的識別碼。如果您不確定，請在`sam init`沒有選項的情況下呼叫互動式工作流程。

如果已指定且`--location`未提供此參數，則`--no-interactive`需要此參數。

此參數僅適用於 0.30.0 及更 AWS SAMCLI新版本。使用舊版指定此參數會導致錯誤。

`--application-insights | --no-application-insights`

為您的 CloudWatch 應用程式啟用 Amazon 應用程式洞見監控。如需進一步了解，請參閱使用應用程式洞察來監控無伺服器 CloudWatch 應用。

預設選項為 `--no-application-insights`。

`--architecture, -a [ x86_64 | arm64 ]`

應用程式 Lambda 函數的指令集架構。指定其中一個x86_64或arm64。

`--base-image [amazon/dotnet8-base | amazon/dotnet6-base | amazon/dotnetcore3.1-base | amazon/go1.x-base | amazon/java21-base | amazon/java17-base | amazon/java11-base | amazon/java8.al2-base | amazon/java8-base | amazon/nodejs20.x-base | amazon/nodejs18.x-base | amazon/nodejs16.x-base | | amazon/python3.12-base | amazon/python3.11-base | amazon/python3.10-base | amazon/python3.9-base | amazon/python3.8-base | amazon/ruby3.3-base | amazon/ruby3.2-base]`

您的應用程序的基本映像。只有當封裝類型為時，此選項才適用Image。

如果已指定、指定--no-interactive為、--package-type且未指定Image，則此參數為必要參數。--location

`--config-env TEXT`

指定組態檔案中要使用的預設參數值的環境名稱。預設值為「預設值」。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--config-file PATH`

包含要使用的預設參數值的組態檔案的路徑和檔案名稱。默認值是項目目錄的根目錄中的「samconfig.toml」。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--debug`

開啟偵錯記錄以列印 AWS SAMCLI產生的偵錯訊息，並顯示時間戳記。

`--dependency-manager, -d [ gradle | mod | maven | bundler | npm | cli-package | pip ]`

Lambda 執行階段的相依性管理員。

`--extra-content`

覆寫範本cookiecutter.json組態中的任何自訂參數，例如{"customParam1": "customValue1", "customParam2":"customValue2"}。

`--help, -h`

顯示此訊息並結束。

`--location, -l TEXT`

模板或應用程序位置（Git，水銀，HTTP /HTTPS，.zip 文件，路徑）。

如果指定了和`--runtime`、和 `--name``--app-template`，則此參數`--no-interactive`為必要參數。

對於 Git 儲存庫，您必須使用儲存庫根目錄的位置。

對於本機路徑，範本必須是 .zip 檔案或餅乾取決方塊格式。

`--name, -n TEXT`

要產生為目錄的專案名稱。

如果已指定且`--location`未提供此參數，則`--no-interactive`需要此參數。

`--no-input`

停用 Cookie 提示，並接受範本組態中定義的 vcfdefault 值。

`--no-interactive`

停用 init 參數的互動式提示，如果缺少任何必要的值，則會失敗。

`--output-dir, -o PATH`

輸出初始化應用程式的位置。

`--package-type [ Zip | Image ]`

範例應用程式的套件類型。 `Zip`會建立 .zip 檔案封存，並`Image`建立容器映像檔。

`--runtime, -r [ dotnet8 | dotnet6 | dotnetcore3.1 | go1.x | java21 | java17 | java11 | java8 | java8.al2 | nodejs20.x | nodejs18.x | nodejs16.x | python3.12 | python3.11 | python3.10 | python3.9 | python3.8 | ruby3.3 | ruby3.2 ]`

應用程式的 Lambda 執行階段。只有當封裝類型為時，此選項才適用`Zip`。

如果已指定、指定`--no-interactive`為、`--package-type`且未指定`Zip`，則此參數為必要參數。`--location`

`--save-params`

將您在命令列中提供的參數儲存到 AWS SAM 組態檔案中。

`--tracing | --no-tracing`

啟用 Lambda 函數的 AWS X-Ray 追蹤功能。

sam list

本頁提供「 AWS Serverless Application Model 指令行介面」(AWS SAMCLI) `sam list` 指令的參考資訊。

如需「」的簡介 AWS SAMCLI，請參閱什麼是 AWS SAMCLI？。

此命`sam list`令會輸出無伺服器應用程式中的資源和無伺服器應用程式狀態的重要資訊。在部署sam list之前和之後使用以協助本地和雲端開發。

用量

```
$ sam list <options> <subcommand>
```

選項

`--help, -h`

顯示此訊息並退出。

子命令

`endpoints`

顯示堆疊中的雲端和本機 AWS CloudFormation 端點清單。如需詳細資訊，請參閱 sam list endpoints。

`resources`

顯示 AWS Serverless Application Model (AWS SAM) 範本中建立的資源，這些資源是在部署 AWS CloudFormation 時建立的。如需詳細資訊，請參閱 sam list resources。

`stack-outputs`

顯示 AWS SAM 或 AWS CloudFormation 範本中 AWS CloudFormation 堆疊的輸出。如需更多詳細資訊，請參閱 sam list stack-outputs。

sam list endpoints

本頁提供「 AWS Serverless Application Model 指令行介面」(AWS SAMCLI) `sam list endpoints` 子指令的參考資訊。

如需「」的簡介 AWS SAMCLI，請參閱什麼是 AWS SAMCLI？。

`sam list endpoints`子命令會顯示來自 AWS CloudFormation 堆疊的雲端和本機端點清單。您可以透過sam local和sam sync指令與這些資源互動。

AWS Lambda 此命令支援 Amazon API Gateway 資源類型。

> **Note**
>
> 為您的 Amazon API Gateway 資源設定時，會支援自訂網域。此命令將輸出自訂網域，而不是預設端點。

用量

```
$ sam list endpoints <options>
```

選項

`--config-env` *TEXT*

指定組態檔案中要使用的預設參數值的環境名稱。

預設值：`default`

如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--config-file` *TEXT*

包含要使用的預設參數值的組態檔案的路徑和檔案名稱。

預設值：`samconfig.toml`在目前工作目錄中。

如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--debug`

開啟偵錯記錄以列印時間戳記所產生 AWS SAMCLI的偵錯訊息。

`--help, -h`

顯示此訊息並退出。

`--output [json|table]`

指定輸出結果的格式。

預設值：`table`

`--profile` *`TEXT`*

從您的認證檔案中選取特定的設定檔以取得認 AWS 證。

`--region` *`TEXT`*

設定服務的 AWS 區域。例如 `us-east-1`。

`--save-params`

將您在命令列中提供的參數儲存到 AWS SAM 組態檔案中。

`--stack-name` *`TEXT`*

已部署 AWS CloudFormation 堆疊的名稱。堆棧名稱可以在應用程序的`samconfig.toml`文件或指定的配置文件中找到。

如果未指定此選項，則會顯示範本中定義的本機資源。

`--template-file, --template, -t` *`PATH`*

AWS SAM 範本檔案。

預設值：`template.[yaml|yml|json]`

範例

以 json 格式顯示名為的 AWS CloudFormation 堆疊中已部署資源端點的輸出`test-stack`。

```
$ sam list endpoints --stack-name test-stack --output json

[
  {
    "LogicalResourceId": "HelloWorldFunction",
    "PhysicalResourceId": "sam-app-test-list-HelloWorldFunction-H85Y7yIV7ZLq",
    "CloudEndpoint": "https://zt55oi7kbljxjmcoahsj3cknwu0rposq.lambda-url.us-
east-1.on.aws/",
    "Methods": "-"
```

```
  },
  {
    "LogicalResourceId": "ServerlessRestApi",
    "PhysicalResourceId": "uj80uoe2o2",
    "CloudEndpoint": [
      "https://uj80uoe2o2.execute-api.us-east-1.amazonaws.com/Prod",
      "https://uj80uoe2o2.execute-api.us-east-1.amazonaws.com/Stage"
    ],
    "Methods": [
      "/hello['get']"
    ]
  }
]
```

sam list resources

本頁提供「 AWS Serverless Application Model 指令行介面」(AWS SAMCLI) `sam list resources` 子指令的參考資訊。

如需「」的簡介 AWS SAMCLI，請參閱什麼是 AWS SAMCLI？。

`sam list resources`子指令會顯示 AWS Serverless Application Model (AWS SAM) 範本中由部署 AWS SAM 轉換 AWS CloudFormation 所建立的資源。

在部署之前sam list resources與 AWS SAM 範本搭配使用，以查看將要建立的資源。提供 AWS CloudFormation 堆疊名稱以檢視包含已部署資源的合併清單。

> **Note**
>
> 若要從 AWS SAM 範本產生資源清單，則會執行範本的本機轉換。此清單中包含要部署條件的資源 (例如特定區域內)。

用量

```
$ sam list resources <options>
```

選項

`--config-env` *TEXT*

指定組態檔案中要使用的預設參數值的環境名稱。

預設值：default

如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

--config-file *TEXT*

包含要使用的預設參數值的組態檔案的路徑和檔案名稱。

預設值：samconfig.toml在目前工作目錄中。

如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

--debug

開啟偵錯記錄以列印時間戳記所產生 AWS SAMCLI的偵錯訊息。

--help, -h

顯示此訊息並退出。

--output [json|table]

指定輸出結果的格式。

預設值：table

--profile *TEXT*

從您的認證檔案中選取特定的設定檔以取得認 AWS 證。

--region *TEXT*

設定服務的 AWS 區域。例如 us-east-1。

--save-params

將您在命令列中提供的參數儲存到 AWS SAM 組態檔案中。

--stack-name *TEXT*

已部署 AWS CloudFormation 堆疊的名稱。堆棧名稱可以在應用程序的samconfig.toml文件或指定的配置文件中找到。

提供時，範本中的資源邏輯 ID 將對應至中的對應實體 ID AWS CloudFormation。若要進一步瞭解實體 ID，請參閱AWS CloudFormation 使用者指南中的資源欄位。

如果未指定此選項，則會顯示範本中定義的本機資源。

`--template-file, --template, -t` *PATH*

AWS SAM 範本檔案。

預設值：`template.[yaml|yml|json]`

範例

以表格格式顯示 AWS SAM 範本機資源的輸出，以及名為的 AWS CloudFormation 堆疊中已部署的資源`test-stack`。從與本機範本相同的目錄執行。

```
$ sam list resources --stack-name test-stack --output table

-------------------------------------------------------------------------------------------------
Logical ID                                                      Physical ID
-------------------------------------------------------------------------------------------------
HelloWorldFunction                                              sam-app-test-list-
HelloWorldFunction-H85Y7yIV7ZLq
HelloWorldFunctionHelloWorldPermissionProd                      sam-app-test-list-

 HelloWorldFunctionHelloWorldPermissionProd-1QH7CPOCBL2IK
HelloWorldFunctionRole                                          sam-app-test-list-
HelloWorldFunctionRole-SRJDMJ6F7F41
ServerlessRestApi                                               uj80uoe2o2
ServerlessRestApiDeployment47fc2d5f9d                           pncw5f
ServerlessRestApiProdStage                                      Prod
ServerlessRestApiDeploymentf5716dc08b                           -
-------------------------------------------------------------------------------------------------
```

sam list stack-outputs

本頁提供「 AWS Serverless Application Model 指令行介面」(AWS SAMCLI) `sam list stack-outputs` 子指令的參考資訊。

如需「」的簡介 AWS SAMCLI，請參閱什麼是 AWS SAMCLI？。

`sam list stack-outputs`子命令會顯示來自 AWS Serverless Application Model (AWS SAM) 或 AWS CloudFormation 範本的 AWS CloudFormation 堆疊輸出。若要取得有關的詳細資訊`Outputs`，請參閱AWS CloudFormation 使用指南中的輸出。

用量

```
$ sam list stack-outputs <options>
```

選項

`--config-env` *TEXT*

指定組態檔案中要使用的預設參數值的環境名稱。

預設值：`default`

如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--config-file` *TEXT*

包含要使用的預設參數值的組態檔案的路徑和檔案名稱。

預設值：`samconfig.toml`在目前工作目錄中。

如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--debug`

開啟偵錯記錄以列印時間戳記所產生 AWS SAMCLI的偵錯訊息。

`--help, -h`

顯示此訊息並退出。

`--output [json|table]`

指定輸出結果的格式。

預設值：`table`

`--profile` *TEXT*

從您的認證檔案中選取特定的設定檔以取得認 AWS 證。

`--region` *TEXT*

設定服務的 AWS 區域。例如 `us-east-1`。

`--save-params`

將您在命令列中提供的參數儲存到 AWS SAM 組態檔案中。

`--stack-name` *TEXT*

已部署 AWS CloudFormation 堆疊的名稱。堆棧名稱可以在應用程序的`samconfig.toml`文件或指定的配置文件中找到。

此選項為必要。

範例

以表格格式顯示名為的 AWS CloudFormation 堆疊中資源的輸出`test-stack`。

```
$ sam list stack-outputs --stack-name test-stack --output table

-------------------------------------------------------------------------------------------------
OutputKey                                 OutputValue
 Description
-------------------------------------------------------------------------------------------------
HelloWorldFunctionIamRole                 arn:aws:iam::account-number:role/sam-
 Implicit IAM Role created for Hello
                                          app-test-list-HelloWorldFunctionRole-    World
 function
                                          SRJDMJ6F7F41
HelloWorldApi                             https://uj80uoe2o2.execute-api.us-       API
 Gateway endpoint URL for Prod
                                          east-1.amazonaws.com/Prod/hello/         stage
 for Hello World function
HelloWorldFunction                        arn:aws:lambda:us-                       Hello
 World Lambda Function ARN
                                          east-1:account-number:function:sam-app-
                                          test-list-
                                          HelloWorldFunction-H85Y7yIV7ZLq
-------------------------------------------------------------------------------------------------
```

sam local generate-event

本頁提供「 AWS Serverless Application Model 指令行介面」(AWS SAMCLI) `sam local generate-event` 子指令的參考資訊。

- 如需「」的簡介 AWS SAMCLI，請參閱什麼是 AWS SAMCLI？。
- 如需使用 AWS SAMCLI`sam local generate-event`指令的文件，請參閱測試簡介 sam local generate-event。

`sam local generate-event`子命令會產生支援 AWS 服務的事件裝載範例。

用量

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

選項

`--config-env TEXT`

指定組態檔案中要使用的預設參數值的環境名稱。預設值為「預設值」。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--config-file PATH`

包含要使用的預設參數值的組態檔案的路徑和檔案名稱。預設值`samconfig.toml`位於專案目錄的根目錄中。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--help`

顯示此訊息並結束。

服務

若要查看支援的服務清單，請執行下列命令：

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

事件

若要查看可針對每個服務產生的支援事件清單，請執行下列命令：

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

事件選項

若要查看您可以修改的支援事件選項清單，請執行下列命令：

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

sam local invoke

本頁提供「 AWS Serverless Application Model 指令行介面」(AWS SAMCLI) `sam local invoke` 子指令的參考資訊。

- 如需「」的簡介 AWS SAMCLI，請參閱什麼是 AWS SAMCLI？。
- 如需使用 AWS SAMCLI`sam local invoke`子指令的文件，請參閱測試簡介 sam local invoke。

`sam local invoke`子命令會在本機啟動函數的一次性叫用。 AWS Lambda

用量

```
$ sam local invoke <arguments> <options>
```

> **Note**
>
> 如果您在 AWS SAM 範本中定義了多個函數，請提供您要叫用的函數邏輯 ID。

引數

資源 ID

要叫用的 Lambda 函數識別碼。

此為選用引數。如果您的應用程式包含單一 Lambda 函數， AWS SAM CLI 會叫用它。如果您的應用程式包含多個函數，請提供要叫用之函數的 ID。

有效值：資源的邏輯 ID 或資源 ARN。

選項

`--add-host` *`LIST`*

將主機名稱傳遞給 IP 地址映射到 Docker 容器的主機文件。此參數可以多次傳遞。

Example

範例：`--add-host` *`example.com:127.0.0.1`*

`--beta-features | --no-beta-features`

允許或拒絕測試版功能。

`--config-env` *TEXT*

指定組態檔案中要使用的預設參數值的環境名稱。預設值為「預設值」。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--config-file` *PATH*

包含要使用的預設參數值的組態檔案的路徑和檔案名稱。在專案目錄的根目錄中，預設值為 `samconfig.toml ""`。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--container-env-vars`

(選擇性) 在本機偵錯時，將環境變數傳遞至 Lambda 函數映像容器。

`--container-host` *TEXT*

本機模擬 Lambda 容器的主機。預設值為 `localhost`。如果要在 macOS 上的 Docker 容器 AWS SAMCLI中運行，則可以指定`host.docker.internal`。如果您想要在不同的主機上執行容器 AWS SAMCLI，您可以指定遠端主機的 IP 位址。

`--container-host-interface` *TEXT*

容器連接埠應繫結之主機網路介面的 IP 位址。預設值為 `127.0.0.1`。用`0.0.0.0`於繫結至所有介面。

`--debug`

開啟偵錯記錄以列印 AWS SAMCLI產生的偵錯訊息，並顯示時間戳記。

`--debug-args` *TEXT*

要傳遞給偵錯工具的其他引數。

`--debug-port, -d` *TEXT*

如果有指定，系統會以偵錯模式啟動 Lambda 函數容器，並在本機主機上公開此連接埠。

`--debugger-path` *TEXT*

掛接至 Lambda 容器之除錯程式的主機路徑。

`--docker-network` *TEXT*

Lambda Docker 容器應連線到的現有 Docker 網路名稱或識別碼，以及預設橋接網路。如果未指定，Lambda 容器只會連線到預設的橋接器 Docker 網路。

`--docker-volume-basedir, -v` *TEXT*

AWS SAM 檔案所在之基本目錄的位置。如果 Docker 在遠端機器上執行，您必須掛載 Docker 機器上 AWS SAM 檔案所在的路徑，並修改此值以符合遠端機器。

`--env-vars, -n` *PATH*

包含 Lambda 函數環境變數值的 JSON 檔案。如需環境變數檔案的更多資訊，請參閱環境變數檔案。

`--event, -e` *PATH*

包含在叫用 Lambda 函數時傳遞至 Lambda 函數的事件資料的 JSON 檔案。如果您未指定此選項，則不會假設任何事件。要從中輸入 JSON`stdin`，您必須傳入值 '-'。如需有關不同 AWS 服務之事件訊息格式的詳細資訊，請參閱開AWS Lambda 發人員指南中的使用其他服務。

`--force-image-build`

指定是否 AWS SAMCLI應重建用來呼叫 Lambda 函數與圖層的影像。

`--help`

顯示此訊息並結束。

`--hook-name TEXT`

用來擴充 AWS SAMCLI功能的掛接名稱。

接受的值：`terraform`。

`--invoke-image` *TEXT*

要用於本機函數叫用之容器映像檔的 URI。根據預設，會從 Amazon ECR 公開 (列於映像儲存庫) AWS SAM 提取容器映像檔。使用此選項可從其他位置提取影像。

例如 `sam local invoke MyFunction --invoke-image amazon/aws-sam-cli-emulation-image-python3.8`。

`--layer-cache-basedir` *DIRECTORY*

指定將範本使用的圖層下載到的基本目錄的位置。

`--log-file, -l` *TEXT*

要將執行階段記錄檔傳送至的記錄檔。

`--no-event`

叫用具有空事件的函數。

`--parameter-overrides`

(選擇性) 包含編碼為索引鍵值配對之 AWS CloudFormation 參數覆寫的字串。使用與 AWS Command Line Interface (AWS CLI) 相同的格式。例如：'ParameterKey=KeyPairName，ParameterValue= MyKey ParameterKey =InstanceType，ParameterValue=t1.micro'。

此選項與不相容`--hook-name`。

`--profile` *TEXT*

從您的認證檔案取得 AWS 認證的特定設定檔。

`--region` *TEXT*

要部署的 AWS 區域。例如 us-east-1。

`--save-params`

將您在命令列中提供的參數儲存到 AWS SAM 組態檔案中。

`--shutdown`

在叫用完成後模擬 shutdown 事件，以測試關機行為的延伸處理。

`--skip-prepare-infra`

如果未進行基礎結構變更，則略過準備階段。搭配`--hook-name`選項使用。

`--skip-pull-image`

依預設，會 AWS SAMCLI檢查 Lambda 最新的遠端執行階段環境，並自動更新本機映像以保持同步。

指定此選項可略過下拉 Lambda 執行階段環境的最新Docker影像。

`--template, -t` *PATH*

AWS SAM 範本檔案。

此選項與不相容`--hook-name`。

> **Note**
>
> 如果指定此選項，則僅 AWS SAM 載入範本及其指向的本機資源。

`--terraform-plan-file`

AWS SAMCLI搭配使用時，本端Terraform平面檔的相對或絕對路徑Terraform Cloud。此選項需要`--hook-name`將設定為`terraform`。

sam local start-api

本頁提供「 AWS Serverless Application Model 指令行介面」(AWS SAMCLI) `sam local start-api` 子指令的參考資訊。

- 如需「」的簡介 AWS SAMCLI，請參閱什麼是 AWS SAMCLI？。
- 如需使用 AWS SAMCLI`sam local start-api`子指令的文件，請參閱測試簡介 sam local start-api。

`sam local start-api`子命令會在本機執行您的 AWS Lambda 函數，以透過本機 HTTP 伺服器主機進行測試。

用量

```
$ sam local start-api <options>
```

選項

`--add-host` *`LIST`*

將主機名稱傳遞給 IP 地址映射到 Docker 容器的主機文件。此參數可以多次傳遞。

Example

範例：`--add-host` *`example.com:127.0.0.1`*

`--beta-features | --no-beta-features`

允許或拒絕測試版功能。

`--config-env` *TEXT*

指定組態檔案中要使用的預設參數值的環境名稱。預設值為「預設值」。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--config-file` *PATH*

包含要使用的預設參數值的組態檔案的路徑和檔案名稱。默認值是項目目錄的根目錄中的「samconfig.toml」。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--container-env-vars`

選用。在本地調試時將環境變量傳遞給映像容器。

`--container-host` *TEXT*

本機模擬 Lambda 容器的主機。預設值為 `localhost`。如果要在 macOS 上的 Docker 容器 AWS SAMCLI中運行，則可以指定`host.docker.internal`。如果您想要在不同的主機上執行容器 AWS SAMCLI，您可以指定遠端主機的 IP 位址。

`--container-host-interface` *TEXT*

容器連接埠應繫結之主機網路介面的 IP 位址。預設值為 `127.0.0.1`。用`0.0.0.0`於繫結至所有介面。

`--debug`

開啟偵錯記錄以列印由產生的偵錯訊息， AWS SAMCLI並顯示時間戳記。

`--debug-args` *TEXT*

要傳遞給調試器的其他參數。

`--debug-function`

選用。指定 Lambda 函數，以在指定時`--warm-containers`套用偵錯選項。此參數適用於`--debug-port``--debugger-path`、和`--debug-args`。

`--debug-port, -d` *TEXT*

如果有指定，系統會以偵錯模式啟動 Lambda 函數容器，並在本機主機上公開此連接埠。

`--debugger-path` *TEXT*

將掛載至 Lambda 容器之除錯器的主機路徑。

`--docker-network` *TEXT*

Lambda Docker 容器應連線到的現有 Docker 網路名稱或識別碼，以及預設橋接網路。如果未指定，Lambda 容器只會連線到預設的橋接器 Docker 網路。

`--docker-volume-basedir, -v` *TEXT*

AWS SAM 檔案所在之基本目錄的位置。如果 Docker 在遠端機器上執行，您必須掛載 Docker 機器上 AWS SAM 檔案所在的路徑，並修改此值以符合遠端機器。

`--env-vars, -n` *PATH*

包含 Lambda 函數環境變數值的 JSON 檔案。

`--force-image-build`

指定是否 AWS SAM CLI應重建用於使用圖層呼叫函數的影像。

`--help`

顯示此訊息並結束。

`--hook-name` *TEXT*

用來擴充 AWS SAMCLI功能的掛接名稱。

接受的值：`terraform`。

`--host` *TEXT*

要綁定的本地主機名稱或 IP 地址（默認值：'127.0.0.1'）。

`--invoke-image` *TEXT*

您要用於 Lambda 函數之容器映像檔的 URI。默認情況下， AWS SAM 從 Amazon ECR 公共提取容器映像。使用此選項可從其他位置提取影像。

您可以多次指定此選項。此選項的每個執行個體都可以使用字串或索引鍵值配對。如果您指定字串，它就是應用程式中所有函式使用的容器映像檔的 URI。例如 `sam local start-api --invoke-image public.ecr.aws/sam/emu-python3.8`。如果您指定索引鍵值配對，則索引鍵是資源名稱，而該值是要用於該資源的容器映像檔的 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` 。使用鍵值配對，您可以為不同的資源指定不同的容器映像檔。

`--layer-cache-basedir` *DIRECTORY*

指定範本使用的圖層下載到哪個位置。

`--log-file, -l TEXT`

要將執行階段記錄檔傳送至的記錄檔。

`--parameter-overrides`

選用。包含編碼為索引鍵值配對之 AWS CloudFormation 參數覆寫的字串。使用與 — 相同的格式 AWS CLI— 例如，'ParameterKey=，ParameterValueMyKey ParameterKey= KeyPairNameInstanceType，ParameterValue=t1.micro'。

`--port, -p INTEGER`

要監聽的本地端口號（默認值：'3000'）。

`--profile TEXT`

從您的認證檔案取得 AWS 認證的特定設定檔。

`--region TEXT`

要部署的 AWS 區域。例如 us-east-1。

`--save-params`

將您在命令列中提供的參數儲存到 AWS SAM 組態檔案中。

`--shutdown`

在調用完成後模擬關閉事件，以測試關閉行為的擴展處理。

`--skip-prepare-infra`

如果未進行基礎結構變更，則略過準備階段。搭配`--hook-name`選項使用。

`--skip-pull-image`

指定 CLI 是否應略過向下拉 Lambda 執行階段的最新 Docker 映像檔。

`--ssl-cert-file PATH`

SSL 憑證檔案的路徑 (預設值：無)。使用此選項時，也必須使用該`--ssl-key-file`選項。

`--ssl-key-file PATH`

SSL 金鑰檔案的路徑 (預設值：無)。使用此選項時，也必須使用該`--ssl-cert-file`選項。

`--static-dir, -s TEXT`

位於此目錄中的任何靜態資產 (例如 CSS JavaScript /HTML) 檔案都會顯示在。/

`--template, -t PATH`

AWS SAM 範本檔案。

> **Note**
>
> 如果指定此選項，則僅 AWS SAM 載入範本及其指向的本機資源。

`--terraform-plan-file`

AWS SAMCLI搭配使用時，本端Terraform平面檔的相對或絕對路徑Terraform Cloud。此選項需要`--hook-name`將設定為`terraform`。

`--warm-containers [EAGER | LAZY]`

選用。指定如何 AWS SAMCLI管理每個函數的容器。

有兩個選項可供選擇：

`EAGER`：所有函數的容器都在啟動時加載，並在調用之間保留。

`LAZY`：只有在第一次呼叫每個函數時，才會載入容器。這些容器會持續存在，以進行其他調用。

sam local start-lambda

本頁提供「 AWS Serverless Application Model 指令行介面」(AWS SAMCLI) `sam local start-lambda` 子指令的參考資訊。

- 如需「」的簡介 AWS SAMCLI，請參閱什麼是 AWS SAMCLI？。
- 如需使用 AWS SAMCLI`sam local start-lambda`子指令的文件，請參閱測試簡介 sam local start-lambda。

`sam local start-lambda`子命令啟動要模擬 AWS Lambda的本地端點。

用量

```
$ sam local start-lambda <options>
```

選項

`--add-host` *`LIST`*

將主機名稱傳遞給 IP 地址映射到 Docker 容器的主機文件。此參數可以多次傳遞。

Example

範例：`--add-host` *`example.com:127.0.0.1`*

`--beta-features | --no-beta-features`

允許或拒絕測試版功能。

`--config-env` *`TEXT`*

指定組態檔案中要使用的預設參數值的環境名稱。預設值為「預設值」。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--config-file` *`PATH`*

包含要使用的預設參數值的組態檔案的路徑和檔案名稱。默認值是項目目錄的根目錄中的「samconfig.toml」。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--container-env-vars`

選用。在本地調試時將環境變量傳遞給映像容器。

`--container-host` *`TEXT`*

本機模擬 Lambda 容器的主機。預設值為 `localhost`。如果要在 macOS 上的 Docker 容器 AWS SAMCLI中運行，則可以指定`host.docker.internal`。如果您想要在不同的主機上執行容器 AWS SAMCLI，您可以指定遠端主機的 IP 位址。

`--container-host-interface` *`TEXT`*

容器連接埠應繫結之主機網路介面的 IP 位址。預設值為 `127.0.0.1`。用`0.0.0.0`於繫結至所有介面。

`--debug`

開啟偵錯記錄以列印由產生的偵錯訊息， AWS SAMCLI並顯示時間戳記。

`--debug-args` *`TEXT`*

要傳遞給調試器的其他參數。

`--debug-function`

選用。指定 Lambda 函數，以在指定時`--warm-containers`套用偵錯選項。此參數適用於`--debug-port`、`--debugger-path`、和`--debug-args`。

`--debug-port, -d` *TEXT*

如果有指定，系統會以偵錯模式啟動 Lambda 函數容器，並在本機主機上公開此連接埠。

`--debugger-path` *TEXT*

要掛接至 Lambda 容器的除錯程式的主機路徑。

`--docker-network` *TEXT*

Lambda Docker 容器應連線到的現有 Docker 網路名稱或識別碼，以及預設橋接網路。如果指定此選項，Lambda 容器只會連線到預設的橋接器 Docker 網路。

`--docker-volume-basedir, -v` *TEXT*

AWS SAM 檔案所在之基本目錄的位置。如果 Docker 在遠端機器上執行，您必須掛載 Docker 機器上 AWS SAM 檔案所在的路徑，並修改此值以符合遠端機器。

`--env-vars, -n` *PATH*

包含 Lambda 函數環境變數值的 JSON 檔案。

`--force-image-build`

指定是否CLI應該重建用於使用圖層調用函數的映像。

`--help`

顯示此訊息並結束。

`--hook-name` *TEXT*

用來擴充 AWS SAMCLI功能的掛接名稱。

接受的值：`terraform`。

`--host` *TEXT*

要綁定的本地主機名稱或 IP 地址（默認值：'127.0.0.1'）。

`--invoke-image` *TEXT*

要用於本機函數叫用之容器映像檔的 URI。默認情況下， AWS SAM 從 Amazon ECR 公共提取容器映像。使用此選項可從其他位置提取影像。

例如 `sam local start-lambda MyFunction --invoke-image amazon/aws-sam-cli-emulation-image-python3.8`。

`--layer-cache-basedir` *DIRECTORY*

指定範本使用的圖層下載到哪個位置。

`--log-file, -l` *TEXT*

要將執行階段記錄檔傳送至的記錄檔。

`--parameter-overrides`

選用。包含編碼為索引鍵值配對之 AWS CloudFormation 參數覆寫的字串。使用與 — 相同的格式 AWS CLI— 例如，'ParameterKey=，ParameterValueMyKey ParameterKey=KeyPairNameInstanceType，ParameterValue=t1.micro'。此選項與不相容`--hook-name`。

`--port, -p` *INTEGER*

要監聽的本地端口號（默認值：'3001'）。

`--profile` *TEXT*

從您的認證檔案取得 AWS 認證的特定設定檔。

`--region` *TEXT*

要部署的 AWS 區域。例如 us-east-1。

`--save-params`

將您在命令列中提供的參數儲存到 AWS SAM 組態檔案中。

`--shutdown`

在叫用完成後模擬 shutdown 事件，以測試關機行為的延伸處理。

`--skip-prepare-infra`

如果未進行基礎結構變更，則略過準備階段。搭配`--hook-name`選項使用。

`--skip-pull-image`

指定是否CLI應略過向下拉 Lambda 執行階段的最新 Docker 映像檔。

`--template, -t` *PATH*

AWS SAM 範本檔案。

> **Note**
>
> 如果指定此選項，則僅 AWS SAM 載入範本及其指向的本機資源。此選項與不相容--hook-name。

`--terraform-plan-file`

AWS SAMCLI搭配使用時，本端Terraform平面檔的相對或絕對路徑Terraform Cloud。此選項需要--hook-name將設定為terraform。

`--warm-containers` *`[EAGER | LAZY]`*

選用。指定如何 AWS SAMCLI管理每個函數的容器。

有兩個選項可供選擇：

- EAGER：所有函數的容器都在啟動時加載，並在調用之間保留。
- LAZY：只有在第一次呼叫每個函數時，才會載入容器。這些容器會持續存在，以進行其他調用。

sam logs

本頁提供「 AWS Serverless Application Model 指令行介面」(AWS SAMCLI) `sam logs` 指令的參考資訊。

如需「」的簡介 AWS SAMCLI，請參閱什麼是 AWS SAMCLI？。

該sam logs命令獲取由您的 AWS Lambda 函數生成的日誌。

用量

```
$ sam logs <options>
```

選項

`--config-env` *`TEXT`*

指定組態檔案中要使用的預設參數值的環境名稱。預設值為「預設值」。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--config-file PATH`

包含要使用的預設參數值的組態檔案的路徑和檔案名稱。默認值是項目目錄的根目錄中的「samconfig.toml」。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--cw-log-group LIST`

包含來自您指定之 CloudWatch 記錄檔記錄群組的記錄檔。如果您同時指定此選項name，則除了來自指定資源的記錄檔外，還會 AWS SAM 包含來自指定記錄群組的記錄。

`--debug`

開啟偵錯記錄以列印由產生的偵錯訊息， AWS SAMCLI並顯示時間戳記。

`---end-time, e TEXT`

截至目前為止擷取記錄檔。時間可以是相對值，例如「5 分鐘前」，「明天」或「2018-01-01 10：10」等格式化的時間戳。

`--filter TEXT`

可讓您指定運算式，以快速尋找符合記錄事件中字詞、片語或值的記錄。這可以是簡單的關鍵字 (例如「錯誤」)，也可以是 Amazon CloudWatch Logs 支援的模式。如需語法的相關資訊，請參閱 Amazon CloudWatch 日誌文件。

`--help`

顯示此訊息並結束。

`--include-traces`

在記錄輸出中包含 X-Ray 軌跡。

`--name, -n TEXT`

要擷取記錄檔的資源名稱。如果此資源是 AWS CloudFormation 堆棧的一部分，則可以是 AWS CloudFormation/AWS SAM 模板中函數資源的邏輯 ID。再次重複參數可以提供多個名稱。如果資源位於嵌套堆棧中，則可以在名稱前面加上嵌套堆棧名稱，以從該資源（NestedStackLogicalId/ResourceLogicalId）中提取日誌。如果沒有給出資源名稱，則會掃描給定的堆棧，並提取所有支持資源的日誌信息。如果未指定此選項，請 AWS SAM 擷取您指定之堆疊中所有資源的記錄檔。支援下列資源類型：

- `AWS::Serverless::Function`
- `AWS::Lambda::Function`
- `AWS::Serverless::Api`

- AWS::ApiGateway::RestApi
- AWS::Serverless::HttpApi
- AWS::ApiGatewayV2::Api
- AWS::Serverless::StateMachine
- AWS::StepFunctions::StateMachine

`--output` *TEXT*

指定記錄檔的輸出格式。若要列印格式化記錄檔，請指定`text`。若要將記錄檔列印為 JSON，請指定`json`。

`--profile` *TEXT*

從您的認證檔案取得 AWS 認證的特定設定檔。

`--region` *TEXT*

要部署的 AWS 區域。例如 us-east-1。

`--save-params`

將您在命令列中提供的參數儲存到 AWS SAM 組態檔案中。

`--stack-name` *TEXT*

該資源所屬的 AWS CloudFormation 堆棧的名稱。

`--start-time, -s` *TEXT*

從這個時候開始擷取記錄檔。時間可以是相對值，例如「5 分鐘前」，「昨天」，或像「2018-01-01 10：10」這樣的格式化時間戳。它默認為「10 分鐘前」。

`--tail, -t`

尾隨日誌輸出。這會忽略結束時間引數，並在記錄檔可用時繼續擷取記錄。

範例

當您的函數是 AWS CloudFormation 堆棧的一部分時，您可以在指定堆棧名稱時使用函數的邏輯 ID 來獲取日誌。

```
$ sam logs -n HelloWorldFunction --stack-name myStack
```

使用-s (-開始時間) 和-e (-結束時間) 選項檢視特定時間範圍的記錄檔。

```
$ sam logs -n HelloWorldFunction --stack-name myStack -s '10min ago' -e '2min ago'
```

您還可以添加選--tail項以等待新日誌並在到達時查看它們。

```
$ sam logs -n HelloWorldFunction --stack-name myStack --tail
```

使用此選--filter項可快速尋找符合日誌事件中字詞、片語或值的記錄檔。

```
$ sam logs -n HelloWorldFunction --stack-name myStack --filter "error"
```

檢視子堆疊中資源的記錄檔。

```
$ sam logs --stack-name myStack -n childStack/HelloWorldFunction
```

應用程序中所有支持的資源的尾部日誌。

```
$ sam logs --stack-name sam-app --tail
```

擷取應用程式中特定 Lambda 函數和 API Gateway API 的記錄檔。

```
$ sam logs --stack-name sam-app --name HelloWorldFunction --name HelloWorldRestApi
```

擷取應用程式中所有支援資源的記錄，以及從指定的記錄群組擷取記錄檔。

```
$ sam logs --cw-log-group /aws/lambda/myfunction-123 --cw-log-group /aws/lambda/
myfunction-456
```

sam package

命 AWS Serverless Application Model 令列介面 (AWS SAM CLI) 封裝 AWS SAM 應用程式。

此命令會建立程式碼和相依性的.zip檔案，並將檔案上傳到 Amazon Simple Storage Service (Amazon S3)。 AWS SAM 為 Amazon S3 中存放的所有檔案啟用加密功能。接著會傳回 AWS SAM 範本的副本，將本機成品的參考取代為命令上傳成品的 Amazon S3 位置。

依預設，當您使用此命令時， AWS SAMCLI會假設您目前的工作目錄是專案的根目錄。 AWS SAMCLI第一個嘗試尋找使用sam build命令建置的範本檔案，位於.aws-sam子資料夾中，並命

名為template.yaml。接下來， AWS SAMCLI會嘗試尋找名為template.yaml或目前工作目錄template.yml中的樣板檔案。如果您指定--template選項， AWS SAMCLI則會覆寫預設行為，而且只會封裝該 AWS SAM 範本及其指向的本機資源。

> **Note**
>
> sam deploy現在會以隱含方式執行的sam package功能。您可以直接使用sam deploy命令來封裝和部署應用程式。

用量

```
$ sam package <arguments> <options>
```

引數

資源 ID

要封裝的 Lambda 函數識別碼。

此為選用引數。如果您的應用程式包含單一 Lambda 函數， AWS SAM CLI 將會封裝它。如果您的應用程式包含多個函數，請提供函數的 ID 以封裝單一函數。

有效值：資源的邏輯 ID 或資源 ARN。

選項

`--config-env` *TEXT*

指定組態檔案中要使用的預設參數值的環境名稱。預設值為「預設值」。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--config-file` *PATH*

包含要使用的預設參數值的組態檔案的路徑和檔案名稱。默認值是項目目錄的根目錄中的「samconfig.toml」。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--debug`

開啟偵錯記錄以列印由產生的偵錯訊息， AWS SAMCLI並顯示時間戳記。

`--force-upload`

覆寫 Amazon S3 儲存貯體中的現有檔案。指定此旗標以上傳成品，即使它們符合 Amazon S3 儲存貯體中的現有成品。

`--help`

顯示此訊息並結束。

`--image-repository` *TEXT*

Amazon Elastic Container Registry (Amazon ECR) 存儲庫的 URI，此命令可以在其中上傳您的函數的映像。對於使用`Image`封裝類型宣告的函數是必要的。

`--kms-key-id` *TEXT*

用於加密 Amazon S3 儲存貯體中靜態成品的 AWS Key Management Service (AWS KMS) 金鑰識別碼。如果未指定此選項，則 AWS SAM 使用 Amazon S3 受管加密金鑰。

`--metadata`

(選擇性) 要附加至範本中參照之所有人工因素的中繼資料對映。

`--no-progressbar`

將成品上傳到 Amazon S3 時，請勿顯示進度列。

`--output-template-file` *PATH*

指令寫入封裝範本的檔案路徑。如果您未指定路徑，命令會將範本寫入標準輸出。

`--profile` *TEXT*

從您的認證檔案取得 AWS 認證的特定設定檔。

`--region` *TEXT*

要部署的 AWS 區域。例如 us-east-1。

`--resolve-s3`

自動建立用於包裝的 Amazon S3 儲存貯體。如果同時指定`--s3-bucket`和`--resolve-s3`選項，則會導致錯誤。

`--s3-bucket` *TEXT*

此命令會上傳您的成品的 Amazon S3 儲存貯體的名稱。如果您的成品大於 51,200 個位元組，則需要使用`--s3-bucket`或`--resolve-s3`選項。如果同時指定`--s3-bucket`和`--resolve-s3`選項，則會導致錯誤。

`--s3-prefix` *TEXT*

新增至上傳至 Amazon S3 儲存貯體的成品名稱的前置詞。前置詞名稱是 Amazon S3 儲存貯體的路徑名稱 (資料夾名稱)。這僅適用於使用`Zip`包類型聲明的函數。

`--save-params`

將您在命令列中提供的參數儲存到 AWS SAM 組態檔案中。

`--signing-profiles` *LIST*

(選擇性) 用來簽署部署套件的簽署設定檔清單。此參數會取得索引鍵值配對的清單，其中索引鍵是要簽署的函數或層的名稱，而該值是簽署設定檔，其中選用的設定檔擁有者以分隔。`:`例如 `FunctionNameToSign=SigningProfileName1 LayerNameToSign=SigningProfileName2:SigningProfileOwner`。

`--template-file, --template, -t` *PATH*

AWS SAM 樣板所在的路徑和檔案名稱。

> **Note**
>
> 如果您指定此選項，則只會 AWS SAM 封裝範本及其指向的本機資源。

`--use-json`

為 AWS CloudFormation 範本輸出 JSON。依預設會使用 YAML。

sam pipeline bootstrap

本頁提供「 AWS Serverless Application Model 指令行介面」(AWS SAMCLI) `sam local pipeline bootstrap` 子指令的參考資訊。

如需「」的簡介 AWS SAMCLI，請參閱什麼是 AWS SAMCLI？。

`sam pipeline bootstrap`子指令會產生連線到 CI/CD 系統所需的 AWS 基礎結構資源。在執行命令之前，必須先針對管線中的每個部署階段執行此步sam pipeline init驟。

此子命令會設定下列 AWS 基礎結構資源：

- 透過下列方式設定管線權限的選項

- 具有存取金鑰 ID 和秘密金鑰存取登入資料的管線 IAM 使用者，可與 CI/CD 系統共用。

  > **Note**
  >
  > 我們建議您定期旋轉存取鍵。如需詳細資訊，請參閱《IAM 使用者指南》中針對需要長期登入資料的使用案例定期輪換存取金鑰。

- 透過 OIDC 支援的 CI/CD 平台。如需搭 AWS SAM 配管線使用 OIDC 的簡介，請移至。如何搭配管線使用 OIDC 驗證 AWS SAM

- 部署 AWS SAM 應用程式所 AWS CloudFormation 承擔的 AWS CloudFormation 執行 IAM 角色。
- 用於保存 AWS SAM 工件的 Amazon S3 存儲桶。
- 選擇性地使用 Amazon ECR 映像儲存庫來保存容器映像 Lambda 部署套件 (如果您的資源屬於套件類型`Image`)。

用量

```
$ sam pipeline bootstrap <options>
```

選項

`--bitbucket-repo-uuid` *TEXT*

比特桶存儲庫的 UUID。此選項特定於使用 Bitbucket OIDC 取得權限。

> **Note**
>
> 此值可以在 https://bitbucket.org/ *###*/*###*/管理員/插件/管理員/管理員/流水線/打開連接找到

`--bucket` *TEXT*

保存 AWS SAM 成品的 Amazon S3 存儲桶的 ARN。

`--cicd-provider` *TEXT*

管線的 CI/CD 平台。 AWS SAM

`--cloudformation-execution-role` *TEXT*

部署應用程式堆疊 AWS CloudFormation 時要採用的 IAM 角色的 ARN。僅當您想使用自己的角色時才提供。否則，該命令將創建一個新角色。

`--config-env` *TEXT*

環境名稱，指定組態檔案中要使用的預設參數值。預設值為 **`default`**。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--config-file` *PATH*

包含要使用的預設參數值的組態檔案的路徑和檔案名稱。預設值`samconfig.toml`位於專案目錄的根目錄中。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--confirm-changeset | --no-confirm-changeset`

提示確認資源的部署。

`--create-image-repository | --no-create-image-repository`

指定如果沒有提供任何影像儲存庫，是否要建立 Amazon ECR 影像儲存庫。Amazon ECR 儲存庫可保存 Lambda 函數的容器映像檔，或包含封裝類型為的`Image`層。預設值為 `--no-create-image-repository`。

`--debug`

開啟偵錯記錄並列印 AWS SAMCLI產生的偵錯訊息，並顯示時間戳記。

`--deployment-branch` *TEXT*

將發生部署的分支名稱。此選項特定於針對權限使用 GitHub 「動作 OIDC」。

`--github-org` *TEXT*

存放庫所屬的 GitHub 組織。如果沒有組織，請輸入存放庫擁有者的使用者名稱。此選項特定於針對權限使用 GitHub「動作 OIDC」。

`--github-repo` *TEXT*

將進行部署的 GitHub 儲存區域名稱。此選項特定於針對權限使用 GitHub 「動作 OIDC」。

`--gitlab-group` *TEXT*

存放庫所屬的 GitLab 群組。此選項特定於使用 GitLab OIDC 取得權限。

`--gitlab-project` *TEXT*

GitLab 專案名稱。此選項特定於使用 GitLab OIDC 取得權限。

`--help, -h`

顯示此訊息並結束。

`--image-repository` *TEXT*

Amazon ECR 影像儲存庫的 ARN，用於保存 Lambda 函數的容器映像檔，或包含封裝類型為的層。Image如果提供，則會忽略這些--create-image-repository選項。如果未提供且--create-image-repository已指定，則指令會建立一個。

`--interactive | --no-interactive`

停用啟動程序參數的互動式提示，如果缺少任何必要的參數，則會失敗。預設值為 --interactive。對於此命令，--stage是唯一需要的參數。

> **Note**
>
> 如果與--no-interactive一起指定--use-oidc-provider，則必須包含 OIDC 提供者的所有必要參數。

`--oidc-client-id` *TEXT*

設定用於 OIDC 提供者的用戶端識別碼。

`--oidc-provider` *[github-actions | gitlab | bitbucket-pipelines]*

將用於 OIDC 權限的 CI/CD 提供者名稱。 GitLab GitHub、和比特桶受支援。

`--oidc-provider-url` *TEXT*

OIDC 提供者的 URL。值必須以開頭**https://**。

`--permissions-provider` *[oidc | iam]*

選擇權限提供者以擔任管線執行角色。預設值為 **iam**。

`--pipeline-execution-role` *TEXT*

管線使用者假設在此階段操作的 IAM 角色的 ARN。僅當您想使用自己的角色時才提供。如果未提供，此命令將創建一個新角色。

`--pipeline-user` *TEXT*

具有存取金鑰 ID 和秘密存取金鑰與 CI/CD 系統共用的 IAM 使用者的 Amazon 資源名稱 (ARN)。它用於授予此 IAM 使用者存取對應 AWS 帳戶的權限。如果未提供，此命令將建立 IAM 使用者以及存取金鑰 ID 和秘密存取金鑰登入資料。

`--profile` *TEXT*

從您的認證檔案取得 AWS 認證的特定設定檔。

`--region` *TEXT*

要部署的 AWS 區域。例如 `us-east-1`。

`--save-params`

將您在命令列中提供的參數儲存到 AWS SAM 組態檔案中。

`--stage` *TEXT*

對應部署階段的名稱。它被用作創建的 AWS 基礎結構資源的後綴。

故障診斷

錯誤：缺少必要的參數

如果`--no-interactive`與`--use-oidc-provider`一起指定，但未提供任何必要的參數，則會顯示此錯誤訊息以及遺失參數的描述。

sam pipeline init

本頁提供「 AWS Serverless Application Model 指令行介面」(AWS SAMCLI) `sam local pipeline init` 子指令的參考資訊。

如需「」的簡介 AWS SAMCLI，請參閱什麼是 AWS SAMCLI？。

`sam pipeline init`子指令會產生一個管線組態檔，您的 CI/CD 系統可用來部署無伺服器應用程式。 AWS SAM

在使用之前`sam pipeline init`，您必須為管道中的每個階段引導必要的資源。您可以執行以引導完`sam pipeline init --bootstrap`成安裝和組態檔案產生程序，或參考您先前使用`sam pipeline bootstrap`指令建立的資源來執行此操作。

用量

```
$ sam pipeline init <options>
```

選項

`--bootstrap`

啟用互動模式，引導使用者建立必要的 AWS 基礎結構資源。

`--config-env` *TEXT*

指定組態檔案中要使用的預設參數值的環境名稱。預設值為 `default`。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--config-file` *TEXT*

包含要使用的預設參數值的組態檔案的路徑和檔案名稱。預設值`samconfig.toml`位於專案根目錄中。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--debug`

開啟偵錯記錄以列印 AWS SAMCLI產生的偵錯訊息，並顯示時間戳記。

`--help, -h`

顯示此訊息並結束。

`--save-params`

將您在命令列中提供的參數儲存到 AWS SAM 組態檔案中。

sam publish

本頁提供「 AWS Serverless Application Model 指令行介面」(AWS SAMCLI) `sam publish` 指令的參考資訊。

如需「」的簡介 AWS SAMCLI，請參閱什麼是 AWS SAMCLI？。

指`sam publish`令會將 AWS SAM 應用程式發佈至 AWS Serverless Application Repository. 此指令會取得封裝 AWS SAM 範本，並將應用程式發佈至指定的 AWS Region。

此命`sam publish`令預期 AWS SAM 範本包含一個`Metadata`區段，其中包含發佈所需的應用程式中繼資料。在本`Metadata`節中，`LicenseUrl`和`ReadmeUrl`屬性必須參考 Amazon Simple Storage Service (Amazon S3) 儲存貯體，而非本機檔案。若要取得有關 AWS SAM 範本`Metadata`區段的更多資訊，請參閱〈〉使用發佈您的應用程式 AWS SAMCLI。

依預設，sam publish會將應用程式建立為私用。在允許其他 AWS 帳戶檢視和部署您的應用程式之前，您必須共用它。如需共用應用程式的相關資訊，請參閱AWS Serverless Application Repository 開發人員指南中的資AWS Serverless Application Repository 源型政策範例。

> **Note**
>
> 目前sam publish不支援發佈在本機指定的巢狀應用程式。如果您的應用程式包含巢狀應用程式，您必須在發佈父應用程式之 AWS Serverless Application Repository 前，將它們分別發佈到。

用量

```
$ sam publish <options>
```

選項

`--config-env` *TEXT*

指定組態檔案中要使用的預設參數值的環境名稱。預設值為「預設值」。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--config-file` *PATH*

包含要使用的預設參數值的組態檔案的路徑和檔案名稱。在專案目錄的根目錄中，預設值為 samconfig.toml ""。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--debug`

開啟偵錯記錄以列印 AWS SAMCLI產生的偵錯訊息，並顯示時間戳記。

`--help`

顯示此訊息並結束。

`--profile` *TEXT*

從您的認證檔案取得 AWS 認證的特定設定檔。

`--region` *TEXT*

要部署的 AWS 區域。例如 us-east-1。

`--save-params`

將您在命令列中提供的參數儲存到 AWS SAM 組態檔案中。

`--semantic-version` *TEXT*

(選擇性) 使用此選項可提供應用程式的語意版本，以覆寫範本檔案`Metadata`區段中的`SemanticVersion`屬性。如需有關語意版本化的詳細資訊，請參閱語意版本化規格。

`--template, -t` *PATH*

AWS SAM 範本檔案的路徑`[default: template.[yaml|yml]]`。

範例

若要發佈應用程式：

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

sam remote invoke

本頁提供「AWS Serverless Application Model 指令行介面」(AWS SAMCLI) `sam remote invoke` 指令的參考資訊。

- 如需「」的簡介 AWS SAMCLI，請參閱什麼是 AWS SAMCLI？。
- 如需使用 AWS SAMCLI`sam remote invoke`指令的文件，請參閱雲端測試簡介 sam remote invoke。

指`sam remote invoke`令會叫用中支援的 AWS 雲端資源。

用量

```
$ sam remote invoke <arguments> <options>
```

引數

資源 ID

要叫用的所謂資源的 ID。

此引數接受下列值：

- Amazon 資源名稱（ARN）— 資源的 ARN。

  > **Tip**
  >
  > 使`sam list stack-outputs --stack-name <stack-name>`用可取得資源的 ARN。

- 邏輯 ID — 資源的邏輯 ID。您也必須使用`--stack-name`選項提供 AWS CloudFormation 堆疊名稱。
- 實體 ID — 資源的實體 ID。當您使用部署資源時，會建立此 ID AWS CloudFormation。

  > **Tip**
  >
  > 用`sam list resources --stack-name <stack-name>`於取得資源的實體 ID。

當您提供 ARN 或實體識別碼時：

如果您提供 ARN 或實體 ID，請勿提供堆疊名稱。使用`--stack-name`選項提供堆疊名稱時，或在組態檔案中定義堆疊名稱時， AWS SAM CLI會自動將您的資源 ID 處理為來自 AWS CloudFormation 堆疊的邏輯 ID 值。

當您未提供資源 ID 時：

如果您沒有提供資源 ID，但確實提供了堆疊名稱和`--stack-name`選項， AWS SAM CLI 會嘗試使用下列邏輯自動叫用 AWS CloudFormation 堆疊中的資源：

1. AWS SAM CLI將按照以下順序識別資源類型，並在堆棧中找到資源類型後移至下一個步驟：
   a. Lambda
   b. Step Functions
   c. Amazon SQS
   d. Kinesis Data Streams
2. 如果資源類型在堆棧中具有單個資源，則 AWS SAM CLI將調用它。如果堆棧中存在資源類型的多個資源，則 AWS SAM CLI將返回錯誤。

以下是 AWS SAM CLI將會執行的範例：

- 包含兩個 Lambda 函數和一個 Amazon SQS 佇列的堆疊 — 由於堆疊包含多個 Lambda 函數，因此 AWS SAM CLI會找到 Lambda 資源類型並傳回和錯誤。
- 包含 Lambda 函數和兩個 Amazon Kinesis Data Streams 應用程式的堆疊 — 由於堆疊包含單一 Lambda 資源，所以 AWS SAM CLI會找到並呼叫該函數。

- 包含單一 Amazon SQS 佇列和兩個 Kinesis Data Streams 應用程式的堆疊 — 由於堆疊包含單一 Amazon SQS 佇列，因此 AWS SAM CLI會找到 Amazon SQS 佇列並呼叫該佇列。

選項

`--beta-features | --no-beta-features`

允許或拒絕測試版功能。

`--config-env` *`TEXT`*

從 AWS SAMCLI組態檔案中指定要使用的環境。

預設：`default`

`--config-file` *`FILENAME`*

指定組態檔案的路徑和檔案名稱。

如需關於組態檔案的詳細資訊，請參閱 配置 AWS SAMCLI。

默認值：`samconfig.toml`位於項目目錄的根目錄。

`--debug`

啟動偵錯記錄。這會 AWS SAMCLI列印偵錯訊息和時間戳記由.

`--event, -e` *`TEXT`*

要傳送至目標資源的事件。

`--event-file` *`FILENAME`*

包含要傳送至目標資源之事件的檔案路徑。

`--help, -h`

顯示幫助消息並退出。

`--output` *`[ text | json ]`*

以特定的輸出格式輸出調用結果。

`json`— 請求中繼資料和資源回應會以 JSON 結構傳回。回應包含完整的 SDK 輸出。

`text`— 以文字結構傳回要求中繼資料。資源響應以調用資源的輸出格式返回。

`--parameter`

您可以傳遞給正在調用的資源的其他Boto3參數。

Amazon Kinesis Data Streams

下列其他參數可用於在 Kinesis 資料串流中放置記錄：

- `ExplicitHashKey='string'`
- `PartitionKey='string'`
- `SequenceNumberForOrdering='string'`
- `StreamARN='string'`

如需每個參數的描述，請參閱中心. 用戶端.

AWS Lambda

下列其他參數可用來叫用 Lambda 資源並接收緩衝回應：

- `ClientContext='base64-encoded string'`
- `InvocationType='[ DryRun | Event | RequestResponse ]'`
- `LogType='[ None | Tail ]'`
- `Qualifier='string'`

下列其他參數可用來叫用含回應串流的 Lambda 資源：

- `ClientContext='base64-encoded string'`
- `InvocationType='[ DryRun | RequestResponse ]'`
- `LogType='[ None | Tail ]'`
- `Qualifier='string'`

如需每個參數的說明，請參閱下列內容：

- 具有緩衝響應的拉姆達-Lambda 客戶端
- Lambda 與響應流-蘭巴達. 客戶端.

Amazon Simple Queue Service (Amazon SQS)

下列其他參數可用於將訊息傳送到 Amazon SQS 佇列：

- `DelaySeconds=integer`
- `MessageAttributes='json string'`

- MessageDeduplicationId='*string*'
- MessageGroupId='*string*'
- MessageSystemAttributes='*json string*'

如需每個參數的說明，請參閱 SQ.Client.

AWS Step Functions

以下附加參數可用於啟動狀態機執行：

- name='*string*'
- traceHeader='*string*'

如需每個參數的說明，請參閱 SFN. 用戶端.

`--profile` *TEXT*

從您的認證檔案取得 AWS 認證的特定設定檔。

`--region` *TEXT*

資源 AWS 區域 的。例如 `us-east-1`。

`--stack-name` *TEXT*

該資源所屬的 AWS CloudFormation 堆棧的名稱。

`--test-event-name` *NAME*

要傳遞至 Lambda 函數的可共用測試事件名稱。

> **Note**
>
> 此選項僅支援 Lambda 函數。

sam remote test-event

本頁提供「 AWS Serverless Application Model 指令行介面」(AWS SAMCLI) `sam remote test-event` 指令的參考資訊。

- 如需「」的簡介 AWS SAMCLI，請參閱什麼是 AWS SAMCLI？。
- 如需使用 AWS SAMCLI`sam remote test-event`指令的文件，請參閱雲端測試簡介 sam remote test-event。

此命令`sam remote test-event`會與 Amazon EventBridge 架構登錄中的可共用測試事件互動。

用量

```
$ sam remote test-event <options> <subcommand>
```

選項

`--help, -h`

顯示幫助消息並退出。

子命令

`delete`

從 EventBridge 結構描述登錄中刪除可共用的測試事件。如需詳細資訊，請參閱sam remote test-event delete。

`get`

從 EventBridge 結構描述登錄檔取得可共用的測試事件。如需詳細資訊，請參閱sam remote test-event get。

`list`

列出 AWS Lambda 函數的現有可共享測試事件。如需詳細資訊，請參閱sam remote test-event list。

`put`

將本機檔案中的事件儲存至結 EventBridge 構描述登錄。如需詳細資訊，請參閱sam remote test-event put。

sam remote test-event delete

本頁提供「 AWS Serverless Application Model 指令行介面」(AWS SAMCLI) `sam remote test-event delete` 子指令的參考資訊。

- 如需「」的簡介 AWS SAMCLI，請參閱什麼是 AWS SAMCLI？。
- 如需使用 AWS SAMCLI`sam remote test-event`指令的文件，請參閱雲端測試簡介 sam remote test-event。

`sam remote test-event delete`子命令會從 Amazon EventBridge 架構登錄中刪除可共用的測試事件。

用量

```
$ sam remote test-event delete <arguments> <options>
```

引數

資源 ID

與可共用測試事件相關聯的 AWS Lambda 函數識別碼。

如果您提供邏輯 ID，則還必須使用`--stack-name`選項為與 Lambda 函數關聯的 AWS CloudFormation 堆疊提供值。

有效值：資源的邏輯 ID 或資源ARN。

選項

`--config-env` *TEXT*

指定組態檔案中要使用的預設參數值的環境名稱。預設值為「預設值」。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--config-file` *PATH*

包含要使用的預設參數值的組態檔案的路徑和檔案名稱。在專案目錄的根目錄中，預設值為`samconfig.toml` ""。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--help, -h`

顯示幫助消息並退出。

`--name` *TEXT*

要刪除的可共用測試事件的名稱。

`--stack-name` *TEXT*

與 Lambda 函數相關聯的 AWS CloudFormation 堆疊名稱。

如果您要提供 Lambda 函數邏輯 ID 做為引數，則需要此選項。

sam remote test-event get

本頁提供「 AWS Serverless Application Model 指令行介面」(AWS SAMCLI) `sam remote test-event get` 子指令的參考資訊。

- 如需「」的簡介 AWS SAMCLI，請參閱什麼是 AWS SAMCLI？。
- 如需使用 AWS SAMCLI`sam remote test-event`指令的文件，請參閱雲端測試簡介 sam remote test-event。

`sam remote test-event get`子命令會從 Amazon EventBridge 架構登錄檔取得可共用的測試事件。

用量

```
$ sam remote test-event get <arguments> <options>
```

引數

資源 ID

與要取得之可共用測試事件相關聯的 AWS Lambda 函數識別碼。

如果您提供邏輯 ID，則還必須使用`--stack-name`選項為與 Lambda 函數關聯的 AWS CloudFormation 堆疊提供值。

有效值：資源的邏輯 ID 或資源ARN。

選項

`--config-env` *TEXT*

指定組態檔案中要使用的預設參數值的環境名稱。預設值為「預設值」。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--config-file` *PATH*

包含要使用的預設參數值的組態檔案的路徑和檔案名稱。在專案目錄的根目錄中，預設值為 `samconfig.toml` ""。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--help, -h`

顯示幫助消息並退出。

`--name` *TEXT*

要取得之可共用測試事件的名稱。

`--output-file` *FILENAME*

在本機電腦上儲存事件的檔案路徑和名稱。

如果您不提供此選項，則 AWS SAM CLI會將可共享測試事件的內容輸出到您的控制台。

`--stack-name` *TEXT*

與 Lambda 函數相關聯的 AWS CloudFormation 堆疊名稱。

如果您要提供 Lambda 函數邏輯 ID 做為引數，則需要此選項。

sam remote test-event list

本頁提供「 AWS Serverless Application Model 指令行介面」(AWS SAMCLI) `sam remote test-event list` 子指令的參考資訊。

- 如需「」的簡介 AWS SAMCLI，請參閱什麼是 AWS SAMCLI？。
- 如需使用 AWS SAMCLI`sam remote test-event`指令的文件，請參閱雲端測試簡介 sam remote test-event。

`sam remote test-event list`子命令列出 Amazon EventBridge 架構登錄中特定 AWS Lambda 函數的現有可共用測試事件。

用量

```
$ sam remote test-event list <arguments> <options>
```

引數

資源 ID

與可共用測試事件相關聯的 Lambda 函數識別碼。

如果您提供邏輯 ID，則還必須使用`--stack-name`選項為與 Lambda 函數關聯的 AWS CloudFormation 堆疊提供值。

有效值：資源的邏輯 ID 或資源ARN。

選項

`--config-env` *TEXT*

指定組態檔案中要使用的預設參數值的環境名稱。預設值為「預設值」。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--config-file` *PATH*

包含要使用的預設參數值的組態檔案的路徑和檔案名稱。在專案目錄的根目錄中，預設值為 `samconfig.toml` ""。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--help, -h`

顯示幫助消息並退出。

`--stack-name` *TEXT*

與 Lambda 函數相關聯的 AWS CloudFormation 堆疊名稱。

如果您要提供 Lambda 函數邏輯 ID 做為引數，則需要此選項。

sam remote test-event put

本頁提供「 AWS Serverless Application Model 指令行介面」(AWS SAMCLI) `sam remote test-event put` 子指令的參考資訊。

- 如需「」的簡介 AWS SAMCLI，請參閱什麼是 AWS SAMCLI？。
- 如需使用 AWS SAMCLI`sam remote test-event`指令的文件，請參閱雲端測試簡介 sam remote test-event。

`sam remote test-event put`子命令會將可共用的測試事件從本機電腦儲存到 Amazon EventBridge 架構登錄。

用量

```
$ sam remote test-event put <arguments> <options>
```

引數

資源 ID

與可共用測試事件相關聯的 AWS Lambda 函數識別碼。

如果您提供邏輯 ID，則還必須使用`--stack-name`選項為與 Lambda 函數關聯的 AWS CloudFormation 堆疊提供值。

有效值：資源的邏輯 ID 或資源ARN。

選項

`--config-env` *TEXT*

指定組態檔案中要使用的預設參數值的環境名稱。預設值為「預設值」。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--config-file` *PATH*

包含要使用的預設參數值的組態檔案的路徑和檔案名稱。在專案目錄的根目錄中，預設值為`samconfig.toml` ""。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--file` *FILENAME*

本機電腦上事件的檔案路徑和名稱。

提供`-`作為要讀取的檔案名稱值`stdin`。

此選項為必要。

`--force, -f`

以相同的名稱覆寫可共用的測試事件。

`--help, -h`

顯示幫助消息並退出。

`--name` *TEXT*

要將可共用測試事件另存為的名稱。

如果 EventBridge 結構描述登錄中存在具有相同名稱的可共用測試事件， AWS SAM CLI將不會覆寫它。若要覆寫，請新增選`--force`項。

`--output-file` *FILENAME*

在本機電腦上儲存事件的檔案路徑和名稱。

如果您不提供此選項，則 AWS SAM CLI會將可共享測試事件的內容輸出到您的控制台。

`--stack-name` *TEXT*

與 Lambda 函數相關聯的 AWS CloudFormation 堆疊名稱。

如果您要提供 Lambda 函數邏輯 ID 做為引數，則需要此選項。

sam sync

本頁提供「AWS Serverless Application Model 指令行介面」(AWS SAMCLI) `sam sync` 指令的參考資訊。

- 如需「」的簡介 AWS SAMCLI，請參閱什麼是 AWS SAMCLI？。
- 如需有關使用的文件 AWS SAMCLI，請參閱該 AWS SAMCLI。

此指`sam sync`令會將本機應用程式變更同步到 AWS 雲端.

用量

```
$ sam sync <options>
```

選項

`--base-dir, -s` *DIRECTORY*

解析與此目錄相關的函數或圖層原始程式碼的相對路徑。使用此選項可變更解析原始程式碼資料夾的相對路徑的方式。依預設，相對路徑會根據 AWS SAM 樣板的位置進行解析。

除了您正在建置的根應用程式或堆疊中的資源之外，此選項也適用於巢狀應用程式或堆疊。此外，此選項適用於下列資源類型和屬性：

- 資源類型:`AWS::Serverless::Function`屬性:`CodeUri`
- 資源類型:`AWS::Serverless::Function`資源屬性:`Metadata`項目:`DockerContext`
- 資源類型:`AWS::Serverless::LayerVersion`屬性:`ContentUri`
- 資源類型:`AWS::Lambda::Function`屬性:`Code`
- 資源類型:`AWS::Lambda::LayerVersion`屬性:`Content`

`--build-image` *TEXT*

建置應用程式時要使用之容器映像檔的 URI。默認情況下， AWS SAM 使用來自 Amazon Elastic Container Registry (Amazon ECR) 公共的容器映像存儲庫 URI。指定此選項可使用不同的影像。

您可以在單一指令中多次使用此選項。每個選項都接受一個字符串或鍵值對。

- String — 指定應用程式中所有資源將使用的容器映像檔的 URI。以下是範例：

```
$ sam sync --build-image amazon/aws-sam-cli-build-image-python3.8
```

- 鍵值配對 — 指定資源名稱作為金鑰，指定要與該資源搭配使用的容器映像 URI 作為值。使用此格式為應用程式中的每個資源指定不同的容器映像 URI。以下是範例：

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

此選項僅在指定選--use-container項時適用，否則將導致錯誤。

`--build-in-source | --no-build-in-source`

提供--build-in-source直接在源文件夾中構建項目。

--build-in-source此選項支援下列執行階段和建置方法：

- 執行階段 — sam init --runtime 選項支援的任何Node.js執行階段。
- 建置方法 — Makefile、esbuild.

此選--build-in-source項與下列選項不相容：

- --use-container

預設：--no-build-in-source

`--capabilities LIST`

您指定允許 AWS CloudFormation 建立特定堆疊的功能清單。某些堆疊範本可能包含可能影響 AWS 帳戶. 例如，透過建立新的 AWS Identity and Access Management (IAM) 使用者。指定此選項可取代預設值。有效值包括以下項目：

- 能力
- 能力命名 (_IAM)
- 能力資源政策
- CAPABILITY_AUTO_EXPAND

預設值：CAPABILITY_NAMED_IAM和 CAPABILITY_AUTO_EXPAND

`--code`

依預設，會 AWS SAM 同步應用程式中的所有資源。指定此選項以僅同步程式碼資源，其中包括下列項目：

- 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

若要同步程式碼資源，請直接 AWS SAM 使用 AWS 服務 API，而不是透過部署 AWS CloudFormation。若要更新您的 AWS CloudFormation 堆疊，請執行sam sync --watch或sam deploy。

`--config-env` *TEXT*

指定組態檔案中要使用的預設參數值的環境名稱。預設值為「預設值」。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--config-file` *PATH*

包含要使用的預設參數值的組態檔案的路徑和檔案名稱。在專案目錄的根目錄中，預設值為 `samconfig.toml` ""。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--dependency-layer | --no-dependency-layer`

指定是否將個別函數的相依性分隔到另一個圖層中，以加速同步程序。

預設：`--dependency-layer`

`--image-repository` *TEXT*

亞馬遜彈性容器註冊表 (Amazon ECR) 存儲庫的名稱，此命令會在其中上傳您的函數的映像。對於使用`Image`封裝類型宣告的函數是必要的。

`--image-repositories` *TEXT*

將函數對應至其 Amazon ECR 儲存庫 URI。通過它們的邏輯 ID 引用函數。以下是範例：

```
$ sam sync --image-repositories Function1=123456789012.dkr.ecr.us-
east-1.amazonaws.com/my-repo
```

您可以在單一指令中多次指定此選項。

`--kms-key-id` *TEXT*

用於加密 Amazon S3 儲存貯體中靜態成品的 AWS Key Management Service (AWS KMS) 金鑰識別碼。如果未指定此選項，則 AWS SAM 使用 Amazon S3 受管加密金鑰。

`--metadata`

要附加至範本中參考之所有成品的中繼資料對映。

`--notification-arns` *LIST*

與堆疊 AWS CloudFormation 關聯的亞馬遜簡單通知服務 (Amazon SNS) 主題 ARN 清單。

`--parameter-overrides`

包含編碼為索引鍵值配對之 AWS CloudFormation 參數覆寫的字串。使用與 AWS Command Line Interface (AWS CLI) 相同的格式。例如 `ParameterKey=ParameterValue InstanceType=t1.micro`。

`--resource` *TEXT*

指定要同步的資源類型。若要同步多個資源，您可以多次指定此選項。此選項支援此選`--code`項。該值必須是下列出的其中一個資源`--code`。例如 `--resource AWS::Serverless::Function --resource AWS::Serverless::LayerVersion`。

`--resource-id` *TEXT*

指定要同步的資源 ID。若要同步多個資源，您可以多次指定此選項。此選項支援此選`--code`項。例如 `--resource-id Function1 --resource-id Function2`。

`--role-arn` *TEXT*

在套用變更集時 AWS CloudFormation 假設的 IAM 角色的 Amazon 資源名稱 (ARN)。

`--s3-bucket` *TEXT*

Amazon Simple Storage Service (Amazon S3) 貯體的名稱，此命令會上傳您的 AWS CloudFormation 範本。如果您的範本大於 51,200 位元組，則需要使用`--s3-bucket`或`--resolve-s3`選項。如果同時指定`--s3-bucket`和`--resolve-s3`選項，則會發生錯誤。

`--s3-prefix TEXT`

新增至您上傳至 Amazon S3 儲存貯體之成品名稱的前置詞。前置詞名稱是 Amazon S3 儲存貯體的路徑名稱 (資料夾名稱)。這僅適用於使用Zip包類型聲明的函數。

`--save-params`

將您在指令行中提供的參數儲存到 AWS SAM 規劃檔中。

`--skip-deploy-sync | --no-skip-deploy-sync`

如果不需--skip-deploy-sync要，指定略過初始基礎結構同步。 AWS SAMCLI會將您的本機 AWS SAM 範本與已部署的 AWS CloudFormation 範本進行比較，並僅在偵測到變更時執行部署。

指--no-skip-deploy-sync定每次執行時執sam sync行 AWS CloudFormation 部署。

如需進一步了解，請參閱略過初始 AWS CloudFormation 部署。

預設：--skip-deploy-sync

`--stack-name TEXT`

您應用程式的 AWS CloudFormation 堆疊名稱。

此選項為必要。

`--tags LIST`

要與建立或更新之堆疊相關聯的標籤清單。 AWS CloudFormation 還將這些標籤傳播到支持它的堆棧中的資源。

`--template-file, --template, -t PATH`

AWS SAM 樣板所在的路徑和檔案名稱。

> **Note**
>
> 如果指定此選項，則僅 AWS SAM 部署範本及其指向的本機資源。

`--use-container, -u`

如果您的函數依賴於具有原生編譯依賴關係的軟件包，請使用此選項在 AWS Lambda類似Docker容器中構建函數。

> **Note**
>
> 目前，此選項與不相容`--dependency-layer`。如果您使用`--use-container`與`--dependency-layer`，會 AWS SAMCLI通知您並繼續。`--no-dependency-layer`

`--watch`

啟動監控本機應用程式是否有變更的程序，並自動將它們同步到 AWS 雲端. 依預設，當您指定此選項時，會在更新應用程式中 AWS SAM 同步所有資源。使用此選項， AWS SAM 執行初始 AWS CloudFormation 部署。然後， AWS SAM 使用 AWS 服務 API 更新程式碼資源。 AWS SAM 用 AWS CloudFormation 於在更新 AWS SAM 範本時更新基礎結構資源。

`--watch-exclude` *`TEXT`*

將檔案或資料夾排除在觀察到檔案變更之外。若要使用此選項，也`--watch`必須提供。

此選項接收鍵值對：

- 金鑰 — 應用程式中 Lambda 函數的邏輯識別碼。
- 值 — 要排除的關聯檔案名稱或資料夾。

當您更新使用`--watch-exclude`此選項指定的任何檔案或資料夾時， AWS SAM CLI將不會起始同步。不過，當其他檔案或資料夾的更新開始同步時，這些檔案或資料夾就會包含在同步中。

您可以在單一指令中多次提供此選項。

sam traces

本頁提供「 AWS Serverless Application Model 指令行介面」(AWS SAMCLI) `sam traces` 指令的參考資訊。

如需「」的簡介 AWS SAMCLI，請參閱什麼是 AWS SAMCLI？。

命`sam traces`令會擷取您 AWS 帳戶 中的 AWS X-Ray 追蹤。 AWS 區域

用量

```
$ sam traces <options>
```

選項

`--config-env` *TEXT*

指定組態檔案中要使用的預設參數值的環境名稱。預設值為「預設值」。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--config-file` *PATH*

包含要使用的預設參數值的組態檔案的路徑和檔案名稱。在專案目錄的根目錄中，預設值為 `samconfig.toml` ""。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--end-time` *TEXT*

擷取到此時為止的追蹤。時間可以是相對值，例如「5 分鐘前」，「明天」或「2018-01-01 10：10」等格式化的時間戳。

`--output` *TEXT*

指定記錄檔的輸出格式。若要列印格式化記錄檔，請指定`text`。若要將記錄檔列印為 JSON，請指定`json`。

`--save-params`

將您在命令列中提供的參數儲存到 AWS SAM 組態檔案中。

`--start-time` *TEXT*

從此時間開始擷取軌跡。時間可以是相對值，例如「5 分鐘前」，「昨天」，或像「2018-01-01 10：10」這樣的格式化時間戳。它默認為「10 分鐘前」。

`--tail`

尾巴跟踪輸出。這會忽略結束時間引數，並在追蹤可用時繼續顯示它們。

`--trace-id` *TEXT*

X-Ray 追蹤的唯一識別碼。

範例

執行下列命令以依 ID 擷取 X-Ray 追蹤。

```
$ sam traces --trace-id tracing-id-1 --trace-id tracing-id-2
```

執行以下指令，以在 X-Ray 軌跡可用時對其進行尾部處理。

```
$ sam traces --tail
```

sam validate

本頁提供「 AWS Serverless Application Model 指令行介面」(AWS SAMCLI) `sam validate` 指令的參考資訊。

如需「」的簡介 AWS SAMCLI，請參閱什麼是 AWS SAMCLI？。

該`sam validate`命令會驗證 AWS SAM 樣板檔是否有效。

用量

```
$ sam validate <options>
```

選項

`--config-env` *TEXT*

指定組態檔案中要使用的預設參數值的環境名稱。預設值為「預設值」。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--config-file` *PATH*

包含要使用的預設參數值的組態檔案的路徑和檔案名稱。默認值是項目目錄的根目錄中的「samconfig.toml」。如需關於組態檔案的詳細資訊，請參閱 AWS SAMCLI配置文件。

`--debug`

開啟偵錯記錄以列印由產生的偵錯訊息， AWS SAMCLI並顯示時間戳記。

`--lint`

通cfn-lint過在模板上運行林化驗證。建立`cfnlintrc`組態檔以指定其他參數。有關更多信息，請參閱存儲庫中的 cfn-lint。AWS CloudFormation GitHub

`--profile` *TEXT*

從您的認證檔案取得 AWS 認證的特定設定檔。

`--region` *TEXT*

要部署的 AWS 區域。例如 us-east-1。

`--save-params`

將您在命令列中提供的參數儲存到 AWS SAM 組態檔案中。

`--template-file, --template, -t` *PATH*

AWS SAM 範本檔案。預設值為 `template.[yaml|yml]`。

如果您的範本位於目前的工作目錄中且已命名`template.[yaml|yml|json]`，則不需要此選項。

如果您剛跑sam build，則不需要此選項。

AWS SAMCLI管理

本節包含如何管理和自訂您的版本的資訊 AWS SAMCLI。這包括如何使用專案層級組態檔來設定 AWS SAMCLI命令參數值的相關資訊。其中也包含有關管理不同版本的資訊 AWS SAMCLI、設定 AWS 認證 AWS SAM 以便代表您撥打 AWS 服務電話，以及您可以自訂的不同方式 AWS SAM。本節以一般 AWS SAM 疑難排解的章節結尾。

主題

- AWS SAMCLI配置文件
- 管理 AWS SAMCLI版本
- 設定 AWS 認證
- 遙測中的 AWS SAMCLI
- AWS SAMCLI疑難排

AWS SAMCLI配置文件

指 AWS Serverless Application Model 令行介面 (AWS SAMCLI) 支援可用於規劃 AWS SAMCLI指令參數值的專案層級規劃檔。

如需建立和使用組態檔的文件，請參閱配置 AWS SAMCLI。

主題

- 預設組態檔案設定
- 支援的組態檔案格式
- 指定組態檔
- 組態檔案基本概

- 參數值規則
- 配置優先級
- 建立和修改組態檔

預設組態檔案設定

AWS SAM 使用下列預設組態檔案設定：

- Name (名稱) – `samconfig`。
- 位置 — 在項目的根目錄。這與您的`template.yaml`檔案位置相同。
- 格式 —TOML. 若要進一步了解，請參閱TOML文件中的 TOML。

以下是包含預設組態檔案名稱和位置的範例專案結構：

```
sam-app
### README.md
### __init__.py
### events
### hello_world
### samconfig.toml
### template.yaml
### tests
```

以下是範例 `samconfig.toml` 檔案：

```
...
version = 0.1

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

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

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

```
resolve_s3 = true

[default.sync.parameters]
watch = true

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

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

支援的組態檔案格式

TOML和[YAML|YML]格式支持。請參閱以下基本語法：

TOML

```
version = 0.1
[environment]
[environment.command]
[environment.command.parameters]
option = parameter value
```

YAML

```
version: 0.1
environment:
  command:
    parameters:
      option: parameter value
```

指定組態檔

依預設，會依下列順序 AWS SAMCLI尋找組態檔案：

1. 自訂組態檔案 — 如果您使用--config-file選項來指定檔案名稱和位置，會先 AWS SAMCLI尋找此檔案。
2. 「默認**samconfig.toml**文件」— 這是默認配置文件名和格式，位於項目的根目錄中。如果您未指定自訂組態檔案，接下 AWS SAMCLI來會尋找此檔案。

3. **samconfig.[yaml|yml]**file — 如果專案的根目錄中samconfig.toml不存在，會 AWS SAMCLI 尋找此檔案。

以下是使用--config-file選項指定自訂組態檔案的範例：

```
$ sam deploy --config-file myconfig.yaml
```

組態檔案基本概

環境

環境是具名識別碼，其中包含一組唯一的組態設定。您可以在單個 AWS SAM 應用程序中擁有多個環境。

預設環境名稱為default。

使用 AWS SAMCLI--config-env此選項指定要使用的環境。

Command

指令是指定參數值的指 AWS SAMCLI令。

若要指定所有指令的參數值，請使用global識別碼。

參考 AWS SAMCLI指令時，請以底線 (_) 取代空格 (–) 和連字號 (_)。請參閱以下範例：

- build
- local_invoke
- local_start_api

參數

參數被指定為鍵值對。

- 關鍵是 AWS SAMCLI命令選項名稱。
- 該值是要指定的值。

指定金鑰時，請使用長格式命令選項名稱，並以底線 (–) 取代連字號 ()。_範例如下：

- region

- stack_name
- template_file

參數值規則

TOML

- 布林值可以是true或false。例如 confirm_changeset = true。
- 對於字串值，請使用引號 ("")。例如 region = "us-west-2"。
- 對於清單值，請使用引號 ("")，並使用空格 ( ) 分隔每個值。例如：capabilities = "CAPABILITY_IAM CAPABILITY_NAMED_IAM"。
- 對於包含索引鍵值配對清單的值，這些配對會以空格分隔 ( )，而且每個配對的值都會以編碼引號 (\" \") 括住。例如 tags = "project=\"my-application\" stage=\"production\""。
- 對於可以多次指定的參數值，該值是引數陣列。例如：image_repositories = ["my-function-1=image-repo-1", "my-function-2=image-repo-2"]。

YAML

- 布林值可以是true或false。例如 confirm_changeset: true。
- 對於包含單一字串值的項目，引號 ("") 是選擇性的。例如 region: us-west-2。這包括包含以單一字串形式提供的多個索引鍵值配對的項目。以下是範例：

```
$ sam deploy --tags "foo=bar hello=world"
```

```
default:
  deploy:
    parameters:
      tags: foo=bar hello=world
```

- 對於包含值清單的項目，或可在單一指令中多次使用的項目，請將它們指定為字串清單。

  以下是範例：

```
$ sam remote invoke --parameter "InvocationType=Event" --parameter "LogType=None"
```

```
default:
```

```
remote_invoke:
  parameter:
  - InvocationType=Event
  - LogType=None
```

配置優先級

設定值時，會發生下列優先順序：

- 您在指令行中提供的參數值優先於範本檔案和組態檔案Parameters區段中的對應值。
- 如果在指令行或組態檔案中使用該--parameter-overrides選項與parameter_overrides索引鍵一起使用，則其值優先於範本檔案Parameters區段中的值。
- 在組態檔案中，針對特定指令提供的項目優先於全域項目。在下面的例子中，sam deploy命令將使用堆棧名稱my-app-stack。

  TOML

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

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

  YAML

  ```
  default:
    global:
      parameters:
        stack_name: common-stack
    deploy:
      parameters:
        stack_name: my-app-stack
  ```

建立和修改組態檔

建立組態檔

當您使用建立應用程式時sam init，會建立預設samconfig.toml檔案。您也可以手動建立組態檔案。

修改組態檔

您可以手動修改組態檔案。此外，在任何 AWS SAMCLI互動式流程期間，已配置的值將顯示在括號 ([ ]) 中。如果您修改這些值， AWS SAMCLI將會更新您的組態檔案。

以下是使用`sam deploy --guided`指令的互動式流程範例：

```
$ sam deploy --guided

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

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

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

修改組態檔案時，會按如下方式 AWS SAMCLI處理全域值：

- 如果參數值存在於組態檔案的`global`區段中，則 AWS SAMCLI不會將該值寫入特定命令區段。
- 如果參數值同時存在於`global`和特定指令區段中，則 AWS SAMCLI會刪除特定項目，以支援整體值。

管理 AWS SAMCLI版本

透過升級、降級和解除安裝來管理您的 AWS Serverless Application Model 指令行介面 (AWS SAMCLI) 版本。或者，您可以下載並安裝 AWS SAMCLI夜間構建。

主題

升級 AWS SAMCLI

Linux

若要在 Linux AWS SAMCLI 上升級，請遵循中的安裝指示安裝 AWS SAMCLI，但將選`--update`項新增至 install 命令，如下所示：

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

macOS

AWS SAMCLI必須透過與安裝它相同的方法進行升級。我們建議您使用套件安裝程式來安裝和升級 AWS SAMCLI.

若要 AWS SAMCLI使用套件安裝程式升級，請安裝最新的套件版本。如需說明，請參閱安裝 AWS SAMCLI。

Windows

若要升級 AWS SAMCLI，請安裝 AWS SAMCLI再次重複中的 Windows 安裝步驟。

解除安裝 AWS SAMCLI

Linux

若要 AWS SAMCLI在 Linux 上解除安裝，您必須執行下列命令來刪除符號連結和安裝目錄：

1. 找到符號連結並安裝路徑。
   - 使用以下which命令查找符號鏈接：

```
which sam
```

輸出顯示 AWS SAM 二進製文件所在的路徑，例如：

```
 /usr/local/bin/sam
```

- 使用以下ls命令查找符號鏈接指向的目錄：

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

在下列範例中，安裝目錄為/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. 刪除符號鏈接。

```
sudo rm /usr/local/bin/sam
```

3. 刪除安裝目錄。

```
sudo rm -rf /usr/local/aws-sam-cli
```

macOS

AWS SAMCLI通過用於安裝它的相同方法卸載。我們建議您使用套件安裝程式來安裝 AWS SAMCLI.

如果您 AWS SAMCLI使用套件安裝程式安裝，請依照下列步驟解除安裝。

若要解除安裝 AWS SAMCLI

1. 修改並執行下列命令以移除 AWS SAMCLI程式：

```
$ sudo rm -rf /path-to/aws-sam-cli
```

   a. *sudo* — 如果您的使用者具有 AWS SAMCLI程式安裝位置的寫入權限，sudo則不需要。否則，sudo 是必要的。
   b. */##-#*裝程式的路徑。 AWS SAMCLI預設位置為 /usr/local。

2. AWS SAMCLI$PATH通過修改並運行以下命令來刪除：

```
$ sudo rm -rf /path-to-symlink-directory/sam
```

   a. *sudo* — 如果您的使用者具有寫入權限$PATH，sudo則不需要。否則，sudo 是必要的。

   b. *path-to-symlink-directory*— 您的$PATH環境變數。預設位置為 /usr/local/bin。

3. 執行下列命 AWS SAMCLI令，確認已解除安裝：

```
$ sam --version
command not found: sam
```

Windows

要卸載 AWS SAMCLI使用 Windows 設置，請按照下列步驟操作：

1. 從開始菜單中，搜索「添加或刪除程序」。
2. 選擇名為命AWS SAM 令列介面的結果，然後選擇 [解除安裝] 以啟動解除安裝程式。
3. 確認您要解除安裝 AWS SAMCLI.

從使用切換Homebrew到管理 AWS SAMCLI

如果您使Homebrew用安裝和升級 AWS SAMCLI，建議您使用 AWS 支援的方法。請依照下列指示切換至支援的方法。

切換使用 Homebrew

1. 依照中的指示解除Homebrew安裝已安裝的 AWS SAM CLI解除安裝受Homebrew管理的版本。
2. 請遵循中安裝 AWS SAMCLI的說明，使用支援的方法安裝 AWS SAM CLI。

管理 AWS SAMCLI夜間組建

您可以下載並安裝 AWS SAMCLI夜間版本。它包含 AWS SAMCLI代碼的預發布版本，可能比生產版本不太穩定。安裝後，您可以使用該`sam-nightly`命令的夜間構建。您可以同時安裝和使用的生產版本和夜間建置版本。 AWS SAMCLI

> **Note**
>
> 每晚構建不包含構建映像的預發布版本。因此，使用此--use-container選項建置無伺服器應用程式會使用組建映像的最新生產版本。

安裝 AWS SAMCLI夜間構建

要安裝 AWS SAMCLI夜間構建，請按照以下說明進行操作。

Linux

您可以使用套件安裝程式，在 Linux x86_64 平台 AWS SAMCLI上安裝夜間建置版本。

若要安裝 AWS SAMCLI夜間組建

1. 從aws-sam-cli GitHub存放庫sam-cli-nightly中下載套件安裝程式。
2. 請遵循安裝的步驟 AWS SAMCLI來安裝每晚組建套件。

macOS

您可以使用每晚構建包安裝程序安裝程序來安裝 AWS SAMCLI on macOS 的夜間構建版本。

若要安裝 AWS SAMCLI夜間組建

1. 從aws-sam-cli GitHub存放庫中下載適用於您平台sam-cli-nightly的套件安裝程式。
2. 請遵循安裝的步驟 AWS SAMCLI來安裝每晚組建套件。

Windows

的每晚構建版本可通過以下下載鏈接獲得：AWS SAMCLI每晚構建。 AWS SAMCLI若要在 Windows 上安裝夜間組建，請執行與中相同的步驟安裝 AWS SAMCLI，但請改用夜間組建下載連結。

若要確認您是否已安裝夜間建置版本，請執行命sam-nightly --version令。此命令的輸出格式為`1.X.Y.dev<YYYYMMDDHHmm>`，例如：

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

從切換Homebrew到套件安裝程式

如果您使Homebrew用安裝和升級 AWS SAMCLI夜間組建，而且想要切換到使用套件安裝程式，請依照下列步驟執行。

從套件安Homebrew裝程式切換

1. 卸載Homebrew已安裝的 AWS SAMCLI夜間構建。

```
$ brew uninstall aws-sam-cli-nightly
```

2. 執行下列命令，確認已解除安裝 AWS SAMCLI每晚組建：

```
$ sam-nightly --version
zsh: command not found: sam-nightly
```

3. 請依照上一節中的步驟安裝 AWS SAMCLI每晚組建。

使用將其安裝 AWS SAMCLI到虛擬環境中 pip

我們建議您使用原生套件安裝程式來安裝 AWS SAMCLI. 如果必須使用pip，建議您將安裝 AWS SAMCLI到虛擬環境中。如果發生錯誤，這可確保整潔的安裝環境和隔離的環境。

> **Note**
>
> 截至 2023 年 10 月 24 日， AWS SAM CLI正在停止支持。Python 3.7如需進一步了解，請參閱AWS SAMCLI中斷支援 Python 3.7。

若要 AWS SAMCLI將其安裝到虛擬環境中

1. 從您選擇的起始目錄中，創建一個虛擬環境並命名它。

   Linux / macOS

```
$ mkdir project
$ cd project
$ python3 -m venv venv
```

Windows

```
> mkdir project
> cd project
> py -3 -m venv venv
```

2. 啟動虛擬環境

Linux / macOS

```
$ . venv/bin/activate
```

提示會變更，以顯示您的虛擬環境處於作用中狀態。

```
(venv) $
```

Windows

```
> venv\Scripts\activate
```

提示會變更，以顯示您的虛擬環境處於作用中狀態。

```
(venv) >
```

3. 將安裝 AWS SAMCLI到您的虛擬環境中。

```
(venv) $ pip install --upgrade aws-sam-cli
```

4. 確認已 AWS SAMCLI正確安裝。

```
(venv) $ sam --version
SAM CLI, version 1.94.0
```

5. 您可以使用 `deactivate` 命令來離開虛擬環境。每當您啟動新的工作階段時，您都必須重新啟用環境。

管理 AWS SAMCLI與 Homebrew

> **Note**
>
> 從 2023 年 9 月開始， AWS 將不再維護 AWS SAMCLI (`aws/tap/aws-sam-cli`) 的 AWS 受管理Homebrew安裝程式。若要繼續使用Homebrew，您可以使用社群管理的安裝程式 (`aws-sam-cli`)。從 2023 年 9 月起，任何引用的Homebrew命令都`aws/tap/aws-sam-cli`將重定向到`aws-sam-cli`。
> 我們建議您使用我們支援的安裝和升級方法。

安裝使 AWS SAMCLI用 Homebrew

> **Note**
>
> 這些指示使用社群管理的 AWS SAMCLIHomebrew安裝程式。有關進一步的支持，請參閱自製核心存儲庫。

若要安裝 AWS SAMCLI

1. 執行下列命令：

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

2. 驗證安裝：

```
$ sam --version
```

成功安裝之後 AWS SAMCLI，您應該會看到如下所示的輸出：

```
SAM CLI, version 1.94.0
```

升級使 AWS SAMCLI用 Homebrew

若要升級使 AWS SAMCLI用Homebrew，請執行下列命令：

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

解除Homebrew安裝已安裝的 AWS SAM CLI

如果使 AWS SAMCLI用安裝Homebrew，請按照下列步驟將其解除安裝。

若要解除安裝 AWS SAMCLI

1. 執行下列命令：

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

2. 執行下列命 AWS SAMCLI令，確認已解除安裝：

```
$ sam --version
command not found: sam
```

切換至社群管理的Homebrew安裝程式

如果您使用的是 AWS 受管理的Homebrew安裝程式 (`aws/tap/aws-sam-cli`)，而且偏好繼續使用 Homebrew，我們建議您切換至社群管理的Homebrew安裝程式 (`aws-sam-cli`)。

若要在單一命令中切換，請執行下列命令：

```
$ brew uninstall aws-sam-cli && brew untap aws/tap && brew cleanup aws/tap && brew
 update && brew install aws-sam-cli
```

請依照下列指示個別執行每個命令。

切換至社群管理的Homebrew安裝程式

1. 解除安裝下列項目的 AWS 受管理Homebrew版本 AWS SAMCLI：

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

2. 確認 AWS SAMCLI已解除安裝：

```
$ which sam
sam not found
```

3. 移除受 AWS 管理的 AWS SAMCLI點選：

```
$ brew untap aws/tap
```

如果您收到類似下列的錯誤訊息，請新增選--force項，然後再試一次。

```
Error: Refusing to untap aws/tap because it contains the following installed
 formulae or casks:
aws-sam-cli-nightly
```

4. 移除 AWS 受管理安裝程式的快取檔案：

```
$ brew cleanup aws/tap
```

5. 更新Homebrew和所有公式：

```
$ brew update
```

6. 安裝以下版本的社群管理版本 AWS SAMCLI：

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

7. 確認已 AWS SAMCLI成功安裝：

```
$ sam --version
SAM CLI, version 1.94.0
```

故障診斷

如果您在安裝或使用時遇到錯誤 AWS SAMCLI，請參閱AWS SAMCLI疑難排。

設定 AWS 認證

命 AWS SAM 令列介面 (CLI) 要求您設定 AWS 認證，以便它可以代表您呼叫 AWS 服務。例如，AWS SAMCLI使調用 Amazon S3 和 AWS CloudFormation.

您可能已經設定 AWS 認證以使用工 AWS 具，例如其中一個 AWS SDK 或. AWS CLI如果您還沒有，本主題會顯示設定 AWS 認證的建議方法。

若要設定 AWS 登入資料，您必須擁有要設定之 IAM 使用者的存取金鑰 ID 和秘密存取金鑰。如需有關存取金鑰 ID 和秘密存取金鑰的資訊，請參閱《IAM 使用者指南》中的〈管理 IAM 使用者的存取金鑰〉。

接下來，確定您是否已 AWS CLI 安裝。然後按照以下其中一節中的說明進行操作：

使用 AWS CLI

如果您已 AWS CLI 安裝，請使用`aws configure`命令並按照提示進行操作：

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

若要取得有關指`aws configure`令的資訊，請參閱《AWS Command Line Interface 使用指南》AWS CLI中的〈快速規劃〉。

不使用 AWS CLI

如果您沒有 AWS CLI 安裝，則可以創建憑據文件或設置環境變量：

- 認證檔案 — 您可以在本機系統的 AWS 認證檔案中設定認證。此檔案必須位於下列其中一個位置：
  - `~/.aws/credentials`在 Linux 或 macOS 系統上
  - Windows 上的 `C:\Users\USERNAME\.aws\credentials`

  此檔案應該包含下列格式的行：

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

- 環境變數 — 您可以設定`AWS_ACCESS_KEY_ID`和`AWS_SECRET_ACCESS_KEY`環境變數。

  若要在 Linux 或 macOS 上設定這些變數，請使用匯出指令：

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

若要在 Windows 上設定這些變數，請使用 set 指令：

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

遙測中的 AWS SAMCLI

在 AWS，我們根據與客戶互動中學到的知識來開發和推出服務。我們使用客戶反饋來迭代我們的產品。遙測是其他資訊，可協助我們進一步瞭解客戶的需求、診斷問題，並提供改善客戶體驗的功能。

命 AWS SAM 令列介面 (CLI) 會收集遙測，例如一般使用量度、系統和環境資訊以及錯誤。如需所收集遙測類型的詳細資訊，請參閱 收集的資訊類型。

AWS SAMCLI不會收集個人資訊，例如使用者名稱或電子郵件地址。同時也不會擷取敏感的專案層級資訊。

客戶可控制是否開啟遙測功能，而且他們可以隨時變更其設定。如果遙測保持開啟狀態，就會在背景AWS SAMCLI傳送遙測資料，而不需要任何額外的客戶互動。

關閉工作階段的遙測

在 macOS 和 Linux 作業系統中，您可以關閉單一工作階段的遙測功能。若要關閉目前工作階段的遙測功能，請執行下列命令，將環境變數 SAM_CLI_TELEMETRY 設定為 `false`。針對每個新的終端或工作階段重複此命令。

```
export SAM_CLI_TELEMETRY=0
```

在所有工作階段中關閉您的設定檔的遙測功能

當您 AWS SAMCLI在作業系統上執行時，請執行下列命令，以關閉所有工作階段的遙測。

關閉 Linux 中的遙測功能

1. 執行：

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

2. 執行：

```
source ~/.profile
```

關閉 macOS 中的遙測功能

1. 執行：

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

2. 執行：

```
source ~/.profile
```

關閉 Windows 中的遙測功能

您可以使用以下命令在終端機視窗的生命週期內暫時設定環境變數：

如果使用命令提示符：

```
set SAM_CLI_TELEMETRY 0
```

如果使用 PowerShell：

```
$env:SAM_CLI_TELEMETRY=0
```

若要在命令提示字元中永久設定環境變數 PowerShell，或使用下列命令：

```
setx SAM_CLI_TELEMETRY 0
```

> **Note**
> 在終端機關閉並重新開啟之前，變更才會生效。

收集的資訊類型

- 使用資訊 — 客戶執行的一般命令和子命令。

- 錯誤和診斷資訊 — 客戶執行的命令狀態和持續時間，包括結束代碼、內部例外狀況名稱，以及連線到 Docker 時的失敗。
- 系統和環境資訊 — Python 版本、作業系統 (視窗、Linux 或 macOS)、 AWS SAMCLI執行環境 (例如 AWS IDE 工具組或終端機)，以及使用屬性的雜湊值。 AWS CodeBuild

進一步了解

所 AWS SAMCLI收集的遙測資料會遵守 AWS 資料隱私權政策。如需詳細資訊，請參閱下列內容：

- AWS 服務條款
- 資料隱私問答集

AWS SAMCLI疑難排

疑難排解使用、安裝和管理 AWS Serverless Application Model 指令行介面時的錯誤訊息 (AWS SAMCLI)。

主題

- 故障診斷
- 錯誤訊息
- 警告訊息

故障診斷

如需相關的疑難排解指引 AWS SAMCLI，請參閱排解安裝錯誤。

錯誤訊息

捲曲錯誤：「捲曲：（6）無法解決：...」

嘗試叫用 API Gateway 端點時，您會看到下列錯誤：

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

這表示您嘗試將要求傳送至無效的網域。如果您的無伺服器應用程式無法成功部署，或您的命令中有錯字，就會發生這種情況。curl使用 AWS CloudFormation 主控台或確認應用程式已成功部署 AWS CLI，並確認您的curl命令是否正確。

錯誤：找不到具有給定堆棧名稱的確切資源信息

在包含單一 Lambda 函數資源的應用程式上執行`sam remote invoke`命令時，您會看到下列錯誤：

```
Error: Can't find exact resource information with given <stack-name>. Please provide
 full resource ARN or --stack-name to resolve the ambiguity.
```

可能的原因：您沒有提供選**`--stack-name`**項。

如果函數 ARN 未作為引數提供，則該`sam remote invoke`命令需要提供該`--stack-name`選項。

解決方案：提供**`--stack-name`**選項。

以下是範例：

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

Invoking Lambda Function HelloWorldFunction



START RequestId: 40593abb-e1ad-4d99-87bd-ac032e364e82 Version: $LATEST
END RequestId: 40593abb-e1ad-4d99-87bd-ac032e364e82
REPORT RequestId: 40593abb-e1ad-4d99-87bd-ac032e364e82  Duration: 11.31 ms
  Billed Duration: 12 ms  Memory Size: 128 MB     Max Memory Used: 67 MB  Init
 Duration: 171.71 ms
{"statusCode":200,"body":"{\"message\":\"hello world\"}"}%
```

錯誤：無法從堆棧名稱中找到資源信息

執行`sam remote invoke`命令並傳遞 Lambda 函數 ARN 做為引數時，您會看到下列錯誤：

```
Error: Can't find resource information from stack name (<stack-name>) and resource id
 (<function-id>)
```

可能的原因：您的**`samconfig.toml`**檔案中定義了堆疊名稱值。

AWS SAMCLI第一個檢查您的`samconfig.toml`文件是否有堆棧名稱。如果有指定，則會將引數作為邏輯 ID 值傳遞。

解決方案：改為傳遞函數的邏輯 ID。

您可以將函數的邏輯 ID 作為引數傳遞，而不是函數的 ARN。

解決方案：從組態檔中移除堆疊名稱值。

您可以從組態檔案中移除堆疊名稱值。這可防 AWS SAMCLI止將函數 ARN 作為邏輯 ID 值傳遞。

修改配置文件`sam build`後運行。

錯誤:無法建立受管資源:找不到認證

執行sam deploy命令時，您會看到下列錯誤：

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

這表示您尚未設定 AWS 認證來啟用進 AWS SAMCLI行 AWS 維修呼叫。若要修正此問題，您必須設定 AWS 認證。如需詳細資訊，請參閱 設定 AWS 認證。

錯誤： FileNotFoundError 在視窗中

在 Windows AWS SAMCLI 上執行命令時，您可能會看到下列錯誤：

```
Error: FileNotFoundError
```

可能的原因：可 AWS SAMCLI能會與超過 Windows 最大路徑限制的檔案路徑互動。

解決方案：若要解決此問題，必須啟用新的長路徑行為。若要這麼做，請參閱 Microsoft 視窗應用程式開發文件中的啟用 Windows 10、1607 版和更新版本中的長路徑。

錯誤：pip 的依賴解析器...

範例錯誤文字：

```
ERROR: pip's dependency resolver does not currently take into account all the packages
 that are installed. This behaviour is the source of the following dependency
 conflicts.
aws-sam-cli 1.58.0 requires aws-sam-translator==1.51.0, but you have aws-sam-translator
 1.58.0 which is incompatible.
aws-sam-cli 1.58.0 requires typing-extensions==3.10.0.0, but you have typing-extensions
 4.4.0 which is incompatible.
```

可能的原因：如果您使用pip安裝套件，套件之間的相依性可能會發生衝突。

套件的每個版本都取決於aws-sam-cli套aws-sam-translator件的某個版本。例如，aws-sam-cli版本 1.58.0 可能取決於 1.51.0 版。aws-sam-translator

如果您安裝 AWS SAMCLI usingpip，然後安裝另一個依賴於較新版本的套件aws-sam-translator，將會發生下列情況：

- aws-sam-translator將會安裝較新版本的。
- 的目前版本aws-sam-cli和較新版本aws-sam-translator可能不相容。
- 當您使用時 AWS SAMCLI，會發生相依性解析程式錯誤。

解決方案：

1. 使用 AWS SAMCLI原生套件安裝程式。
   a. 卸載 AWS SAMCLI使用 pip。如需說明，請參閱解除安裝 AWS SAMCLI。
   b. AWS SAMCLI使用原生套件安裝程式安裝。如需說明，請參閱安裝 AWS SAMCLI。
   c. 必要時，請 AWS SAMCLI使用原生套件安裝程式升級。如需說明，請參閱升級 AWS SAMCLI。
2. 如果必須使用pip，建議您將 AWS SAM CLI 安裝到虛擬環境中。如果發生錯誤，這可確保整潔的安裝環境和隔離的環境。如需說明，請參閱使用將其安裝 AWS SAMCLI到虛擬環境中 pip。

錯誤：沒有這樣的命令 '遠程'

執行sam remote invoke命令時，您會看到下列錯誤：

```
$ sam remote invoke ...
2023-06-20 08:15:07 Command remote not available
Usage: sam [OPTIONS] COMMAND [ARGS]...
Try 'sam -h' for help.

Error: No such command 'remote'.
```

可能的原因：您的版本 AWS SAMCLI已過期。

該 AWS SAMCLIsam remote invoke命令與 AWS SAMCLI版本 1.88.0 一起發布。您可以通過運行sam --version命令來檢查您的版本。

解決方案：升級 AWS SAMCLI到最新版本。

如需說明，請參閱升級 AWS SAMCLI。

錯誤：在本機執行 AWS SAM 專案需要Docker。你安裝了嗎？

執行sam local start-api命令時，您會看到下列錯誤：

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

這表示您沒有Docker正確安裝。 Docker需要在本地測試您的應用程序。要解決此問題，請按照為開發主機安裝 Docker 的說明進行操作。如需詳細資訊，請參閱 安裝 Docker。

錯誤：未滿足安全性限制

執行時sam deploy --guided，系統會提示您提供問題`Function may not have authorization defined, Is this okay? [y/N]`。如果您使用 **N** (預設回應) 回應此提示，您會看到下列錯誤：

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

提示會通知您即將部署的應用程式可能已設定可公開存取的 Amazon API Gateway API，未經授權。通過響**N**應此提示，您說這是不確定的。

若要修正此問題，您有下列選項：

- 使用授權配置您的應用程序。如需有關配置授權的資訊，請參閱使用 AWS SAM 範本控制 API 存取。
- 如果您打算在未經授權的情況下擁有可公開存取的 API 端點，請重新啟動部署並回應此問題，**Y**以指出您可以進行部署。

消息：缺少身份驗證令牌

嘗試叫用 API Gateway 端點時，您會看到下列錯誤：

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

這表示您嘗試將要求傳送至正確的網域，但是 URI 無法辨識。若要修正此問題，請驗證完整的 URL，然後使用正確的 URL 更新curl命令。

警告訊息

警告：... AWS 將不再維護安Homebrew裝程序 AWS SAM ...

AWS SAMCLI使用安裝時Homebrew，您會看到下列警告訊息：

```
Warning: ... AWS will no longer maintain the Homebrew installer for AWS SAM (aws/tap/
aws-sam-cli).
    For AWS supported installations, use the first party installers ...
```

潛在原因： AWS 不再維持Homebrew支持。

從 2023 年 9 月開始， AWS 將不再維護. Homebrew AWS SAMCLI

解決方案：使用 AWS 支援的安裝方法。

- 您可以在中找到 AWS 支援的安裝方法安裝 AWS SAMCLI。

解決方案：若要繼續使用Homebrew，請使用社群管理的安裝程式。

- 您可以自行決定使用社群管理的Homebrew安裝程式。如需說明，請參閱 管理 AWS SAMCLI 與 Homebrew。

AWS SAM 連接器參考

本節包含 AWS Serverless Application Model (AWS SAM) 連接器資源類型的參考資訊。如需連接器的簡介，請參閱使用 AWS SAM 連接器管理資源權限。

連接器支援的來源和目標資源類型

資`AWS::Serverless::Connector`源類型支援來源和目標資源之間的選取連線數目。在 AWS SAM 範本中設定連接器時，請使用下表來參照支援的連線，以及需要針對每個來源和目標資源類型定義的內容。如需有關在範本中配置連接器的詳細資訊，請參閱AWS::Serverless::Connector。

對於來源和目標資源，在同一個範本中定義時，請使用`Id`屬性。或者，您`Qualifier`可以新增一個來縮小已定義資源的範圍。當資源不在同一個範本中時，請使用支援的屬性組合。

若要要求新的連線，請在serverless-application-model AWS GitHub儲存庫中提交新問題。

來源類型	目的地類型	許可	來源屬性	目的地屬性
AWS::ApiGateway::RestApi	AWS::Lambda::Function	Write	Id或Qualifier ResourceId 、和 Type	Id或Arn和 Type
AWS::ApiGateway::RestApi	AWS::Serverless::Function	Write	Id或Qualifier ResourceId 、和 Type	Id或Arn和 Type
AWS::ApiGatewayV2::Api	AWS::Lambda::Function	Write	Id或Qualifier ResourceId 、和 Type	Id或Arn和 Type
AWS::ApiGatewayV2::Api	AWS::Serverless::Function	Write	Id或Qualifier ResourceId 、和 Type	Id或Arn和 Type
AWS::AppSync::DataSource	AWS::DynamoDB::Table	Read	Id或RoleName和 Type	Id或Arn和 Type
AWS::AppSync::DataSource	AWS::DynamoDB::Table	Write	Id或RoleName和 Type	Id或Arn和 Type
AWS::AppSync::DataSource	AWS::Events::EventBus	Write	Id或RoleName和 Type	Id或Arn和 Type
AWS::AppSync::DataSource	AWS::Lambda::Function	Write	Id或RoleName和 Type	Id或Arn和 Type
AWS::AppSync::DataSource	AWS::Serverless::Function	Write	Id或RoleName和 Type	Id或Arn和 Type

來源類型	目的地類型	許可	來源屬性	目的地屬性
AWS::AppSync::DataSource	AWS::Serverless::SimpleTable	Read	Id或RoleName和Type	Id或Arn和 Type
AWS::AppSync::DataSource	AWS::Serverless::SimpleTable	Write	Id或RoleName和Type	Id或Arn和 Type
AWS::AppSync::GraphQLApi	AWS::Lambda::Function	Write	Id或ResourceId 和 Type	Id或Arn和 Type
AWS::AppSync::GraphQLApi	AWS::Serverless::Function	Write	Id或ResourceId 和 Type	Id或Arn和 Type
AWS::DynamoDB::Table	AWS::Lambda::Function	Read	Id或Arn和 Type	Id或RoleName和Type
AWS::DynamoDB::Table	AWS::Serverless::Function	Read	Id或Arn和 Type	Id或RoleName和Type
AWS::Events::Rule	AWS::Events::EventBus	Write	Id或RoleName和Type	Id或Arn和 Type
AWS::Events::Rule	AWS::Lambda::Function	Write	Id或Arn和 Type	Id或Arn和 Type
AWS::Events::Rule	AWS::Serverless::Function	Write	Id或Arn和 Type	Id或Arn和 Type

來源類型	目的地類型	許可	來源屬性	目的地屬性
AWS::Events::Rule	AWS::Serverless::StateMachine	Write	Id或RoleName和Type	Id或Arn和 Type
AWS::Events::Rule	AWS::SNS::Topic	Write	Id或Arn和 Type	Id或Arn和 Type
AWS::Events::Rule	AWS::SQS::Queue	Write	Id或Arn和 Type	Id或ArnQueueUrl、和 Type
AWS::Events::Rule	AWS::StepFunctions::StateMachine	Write	Id或RoleName和Type	Id或Arn和 Type
AWS::Lambda::Function	AWS::DynamoDB::Table	Read, Write	Id或RoleName和Type	Id或Arn和 Type
AWS::Lambda::Function	AWS::Events::EventBus	Write	Id或RoleName和Type	Id或Arn和 Type
AWS::Lambda::Function	AWS::Lambda::Function	Write	Id或RoleName和Type	Id或Arn和 Type
AWS::Lambda::Function	AWS::Location::PlaceIndex	Read	Id或RoleName和Type	Id或Arn和 Type
AWS::Lambda::Function	AWS::S3::Bucket	Read, Write	Id或RoleName和Type	Id或Arn和 Type

來源類型	目的地類型	許可	來源屬性	目的地屬性
AWS::Lambda::Function	AWS::Serverless::Function	Write	Id或RoleName和Type	Id或Arn和 Type
AWS::Lambda::Function	AWS::Serverless::SimpleTable	Read, Write	Id或RoleName和Type	Id或Arn和 Type
AWS::Lambda::Function	AWS::Serverless::StateMachine	Read, Write	Id或RoleName和Type	Id或ArnName、和 Type
AWS::Lambda::Function	AWS::SNS::Topic	Write	Id或RoleName和Type	Id或Arn和 Type
AWS::Lambda::Function	AWS::SQS::Queue	Read, Write	Id或RoleName和Type	Id或Arn和 Type
AWS::Lambda::Function	AWS::StepFunctions::StateMachine	Read, Write	Id或RoleName和Type	Id或ArnName、和 Type
AWS::S3::Bucket	AWS::Lambda::Function	Write	Id或Arn和 Type	Id或Arn和 Type
AWS::S3::Bucket	AWS::Serverless::Function	Write	Id或Arn和 Type	Id或Arn和 Type

來源類型	目的地類型	許可	來源屬性	目的地屬性
AWS::Serverless::Api	AWS::Lambda::Function	Write	Id或Qualifier ResourceId 、和 Type	Id或Arn和 Type
AWS::Serverless::Api	AWS::Serverless::Function	Write	Id或Qualifier ResourceId 、和 Type	Id或Arn和 Type
AWS::Serverless::Function	AWS::DynamoDB::Table	Read, Write	Id或RoleName和 Type	Id或Arn和 Type
AWS::Serverless::Function	AWS::Events::EventBus	Write	Id或RoleName和 Type	Id或Arn和 Type
AWS::Serverless::Function	AWS::Lambda::Function	Write	Id或RoleName和 Type	Id或Arn和 Type
AWS::Serverless::Function	AWS::S3::Bucket	Read, Write	Id或RoleName和 Type	Id或Arn和 Type
AWS::Serverless::Function	AWS::Serverless::Function	Write	Id或RoleName和 Type	Id或Arn和 Type
AWS::Serverless::Function	AWS::Serverless::SimpleTable	Read, Write	Id或RoleName和 Type	Id或Arn和 Type
AWS::Serverless::Function	AWS::Serverless::StateMachine	Read, Write	Id或RoleName和 Type	Id或ArnName、和 Type

來源類型	目的地類型	許可	來源屬性	目的地屬性
AWS::Serverless::Function	AWS::SNS::Topic	Write	Id或RoleName和Type	Id或Arn和 Type
AWS::Serverless::Function	AWS::SQS::Queue	Read, Write	Id或RoleName和Type	Id或Arn和 Type
AWS::Serverless::Function	AWS::StepFunctions::StateMachine	Read, Write	Id或RoleName和Type	Id或ArnName、和 Type
AWS::Serverless::HttpApi	AWS::Lambda::Function	Write	Id或Qualifier ResourceId 、和 Type	Id或Arn和 Type
AWS::Serverless::HttpApi	AWS::Serverless::Function	Write	Id或Qualifier ResourceId 、和 Type	Id或Arn和 Type
AWS::Serverless::SimpleTable	AWS::Lambda::Function	Read	Id或Arn和 Type	Id或RoleName和Type
AWS::Serverless::SimpleTable	AWS::Serverless::Function	Read	Id或Arn和 Type	Id或RoleName和Type
AWS::Serverless::StateMachine	AWS::DynamoDB::Table	Read, Write	Id或RoleName和Type	Id或Arn和 Type

來源類型	目的地類型	許可	來源屬性	目的地屬性
AWS::Serverless::StateMachine	AWS::Events::EventBus	Write	Id或RoleName和Type	Id或Arn和 Type
AWS::Serverless::StateMachine	AWS::Lambda::Function	Write	Id或RoleName和Type	Id或Arn和 Type
AWS::Serverless::StateMachine	AWS::S3::Bucket	Read, Write	Id或RoleName和Type	Id或Arn和 Type
AWS::Serverless::StateMachine	AWS::Serverless::Function	Write	Id或RoleName和Type	Id或Arn和 Type
AWS::Serverless::StateMachine	AWS::Serverless::SimpleTable	Read, Write	Id或RoleName和Type	Id或Arn和 Type
AWS::Serverless::StateMachine	AWS::Serverless::StateMachine	Read, Write	Id或RoleName和Type	Id或ArnName、和 Type
AWS::Serverless::StateMachine	AWS::SNS::Topic	Write	Id或RoleName和Type	Id或Arn和 Type

來源類型	目的地類型	許可	來源屬性	目的地屬性
AWS::Serverless::StateMachine	AWS::SQS::Queue	Write	Id或RoleName和Type	Id或Arn和 Type
AWS::Serverless::StateMachine	AWS::StepFunctions::StateMachine	Read, Write	Id或RoleName和Type	Id或ArnName、和 Type
AWS::SNS::Topic	AWS::Lambda::Function	Write	Id或Arn和 Type	Id或Arn和 Type
AWS::SNS::Topic	AWS::Serverless::Function	Write	Id或Arn和 Type	Id或Arn和 Type
AWS::SNS::Topic	AWS::SQS::Queue	Write	Id或Arn和 Type	Id或ArnQueueUrl、和 Type
AWS::SQS::Queue	AWS::Lambda::Function	Read, Write	Id或Arn和 Type	Id或RoleName和Type
AWS::SQS::Queue	AWS::Serverless::Function	Read, Write	Id或Arn和 Type	Id或RoleName和Type
AWS::StepFunctions::StateMachine	AWS::DynamoDB::Table	Read, Write	Id或RoleName和Type	Id或Arn和 Type

來源類型	目的地類型	許可	來源屬性	目的地屬性
AWS::StepFunctions::StateMachine	AWS::Events::EventBus	Write	Id或RoleName和Type	Id或Arn和 Type
AWS::StepFunctions::StateMachine	AWS::Lambda::Function	Write	Id或RoleName和Type	Id或Arn和 Type
AWS::StepFunctions::StateMachine	AWS::S3::Bucket	Read, Write	Id或RoleName和Type	Id或Arn和 Type
AWS::StepFunctions::StateMachine	AWS::Serverless::Function	Write	Id或RoleName和Type	Id或Arn和 Type
AWS::StepFunctions::StateMachine	AWS::Serverless::SimpleTable	Read, Write	Id或RoleName和Type	Id或Arn和 Type
AWS::StepFunctions::StateMachine	AWS::Serverless::StateMachine	Read, Write	Id或RoleName和Type	Id或ArnName、和 Type
AWS::StepFunctions::StateMachine	AWS::SNS::Topic	Write	Id或RoleName和Type	Id或Arn和 Type

來源類型	目的地類型	許可	來源屬性	目的地屬性
AWS::StepFunctions::StateMachine	AWS::SQS::Queue	Write	Id或RoleName和Type	Id或Arn和 Type
AWS::StepFunctions::StateMachine	AWS::StepFunctions::StateMachine	Read, Write	Id或RoleName和Type	Id或ArnName、和 Type

連接器建立的 IAM 政策

本節說明使用連接器 AWS SAM 時建立的 AWS Identity and Access Management (IAM) 政策。

AWS::DynamoDB::Table 設定為 AWS::Lambda::Function

策略類型

附加至AWS::Lambda::Function角色的客戶管理政策。

存取類別

Read

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:DescribeStream",
        "dynamodb:GetRecords",
        "dynamodb:GetShardIterator",
        "dynamodb:ListStreams"
      ],
      "Resource": [
        "%{Source.Arn}/stream/*"
      ]
    }
  ]
```

```
}
```

AWS::Events::Rule 設定為 AWS::SNS::Topic

策略類型

AWS::SNS::TopicPolicy附加到AWS::SNS::Topic.

存取類別

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "events.amazonaws.com"
      },
      "Resource": "%{Destination.Arn}",
      "Action": "sns:Publish",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "%{Source.Arn}"
        }
      }
    }
  ]
}
```

AWS::Events::Rule 設定為 AWS::Events::EventBus

策略類型

附加至AWS::Events::Rule角色的客戶管理政策。

存取類別

Write

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

```
      "Effect": "Allow",
      "Action": [
        "events:PutEvents"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

AWS::Events::Rule 設定為 AWS::StepFunctions::StateMachine

策略類型

附加至AWS::Events::Rule角色的客戶管理政策。

存取類別

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:StartExecution"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

AWS::Events::Rule 設定為 AWS::Lambda::Function

策略類型

AWS::Lambda::Permission附加到AWS::Lambda::Function.

存取類別

Write

```
{
  "Action": "lambda:InvokeFunction",
  "Principal": "events.amazonaws.com",
  "SourceArn": "%{Source.Arn}"
}
```

AWS::Events::Rule 設定為 AWS::SQS::Queue

策略類型

AWS::SQS::QueuePolicy附加到AWS::SQS::Queue.

存取類別

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "events.amazonaws.com"
      },
      "Resource": "%{Destination.Arn}",
      "Action": "sqs:SendMessage",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "%{Source.Arn}"
        }
      }
    }
  ]
}
```

AWS::Lambda::Function 設定為 AWS::Lambda::Function

策略類型

附加至AWS::Lambda::Function角色的客戶管理政策。

存取類別

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeAsync",
        "lambda:InvokeFunction"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

AWS::Lambda::Function 設定為 AWS::S3::Bucket

策略類型

附加至AWS::Lambda::Function角色的客戶管理政策。

存取類別

Read

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectAcl",
        "s3:GetObjectLegalHold",
        "s3:GetObjectRetention",
        "s3:GetObjectTorrent",
        "s3:GetObjectVersion",
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionForReplication",
        "s3:GetObjectVersionTorrent",
        "s3:ListBucket",
        "s3:ListBucketMultipartUploads",
        "s3:ListBucketVersions",
        "s3:ListMultipartUploadParts"
      ],
```

```
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}/*"
      ]
    }
  ]
}
```

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:AbortMultipartUpload",
        "s3:DeleteObject",
        "s3:DeleteObjectVersion",
        "s3:PutObject",
        "s3:PutObjectLegalHold",
        "s3:PutObjectRetention",
        "s3:RestoreObject"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}/*"
      ]
    }
  ]
}
```

AWS::Lambda::Function 設定為 AWS::DynamoDB::Table

策略類型

附加至AWS::Lambda::Function角色的客戶管理政策。

存取類別

Read

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

```
      "Effect": "Allow",
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:Query",
        "dynamodb:Scan",
        "dynamodb:BatchGetItem",
        "dynamodb:ConditionCheckItem",
        "dynamodb:PartiQLSelect"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}/index/*"
      ]
    }
  ]
}
```

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:PutItem",
        "dynamodb:UpdateItem",
        "dynamodb:DeleteItem",
        "dynamodb:BatchWriteItem",
        "dynamodb:PartiQLDelete",
        "dynamodb:PartiQLInsert",
        "dynamodb:PartiQLUpdate"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}/index/*"
      ]
    }
  ]
}
```

AWS::Lambda::Function 設定為 AWS::SQS::Queue

策略類型

附加至AWS::Lambda::Function角色的客戶管理政策。

存取類別

Read

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sqs:ReceiveMessage",
        "sqs:GetQueueAttributes"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sqs:DeleteMessage",
        "sqs:SendMessage",
        "sqs:ChangeMessageVisibility",
        "sqs:PurgeQueue"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

AWS::Lambda::Function 設定為 AWS::SNS::Topic

策略類型

附加至AWS::Lambda::Function角色的客戶管理政策。

存取類別

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

AWS::Lambda::Function 設定為 AWS::StepFunctions::StateMachine

策略類型

附加至AWS::Lambda::Function角色的客戶管理政策。

存取類別

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:StartExecution",
        "states:StartSyncExecution"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    },
    {
      "Effect": "Allow",
```

```
      "Action": [
        "states:StopExecution"
      ],
      "Resource": [
        "arn:${AWS::Partition}:states:${AWS::Region}:${AWS::AccountId}:execution:
%{Destination.Name}:*"
      ]
    }
  ]
}
```

Read

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:DescribeStateMachine",
        "states:ListExecutions"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "states:DescribeExecution",
        "states:DescribeStateMachineForExecution",
        "states:GetExecutionHistory"
      ],
      "Resource": [
        "arn:${AWS::Partition}:states:${AWS::Region}:${AWS::AccountId}:execution:
%{Destination.Name}:*"
      ]
    }
  ]
}
```

AWS::Lambda::Function 設定為 AWS::Events::EventBus

策略類型

附加至AWS::Lambda::Function角色的客戶管理政策。

存取類別

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:PutEvents"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

AWS::Lambda::Function 設定為 AWS::Location::PlaceIndex

策略類型

附加至AWS::Lambda::Function角色的客戶管理政策。

存取類別

Read

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "geo:DescribePlaceIndex",
        "geo:GetPlace",
        "geo:SearchPlaceIndexForPosition",
        "geo:SearchPlaceIndexForSuggestions",
        "geo:SearchPlaceIndexForText"
      ],
      "Resource": [
        "%{Destination.Arn}"
```

```
      ]
    }
  ]
}
```

AWS::ApiGatewayV2::Api 設定為 AWS::Lambda::Function

策略類型

AWS::Lambda::Permission附加到AWS::Lambda::Function.

存取類別

Write

```
{
  "Action": "lambda:InvokeFunction",
  "Principal": "apigateway.amazonaws.com",
  "SourceArn": "arn:${AWS::Partition}:execute-api:${AWS::Region}:${AWS::AccountId}:
%{Source.ResourceId}/%{Source.Qualifier}"
}
```

AWS::ApiGateway::RestApi 設定為 AWS::Lambda::Function

策略類型

AWS::Lambda::Permission附加到AWS::Lambda::Function.

存取類別

Write

```
{
  "Action": "lambda:InvokeFunction",
  "Principal": "apigateway.amazonaws.com",
  "SourceArn": "arn:${AWS::Partition}:execute-api:${AWS::Region}:${AWS::AccountId}:
%{Source.ResourceId}/%{Source.Qualifier}"
}
```

AWS::SNS::Topic 設定為 AWS::SQS::Queue

策略類型

AWS::SQS::QueuePolicy附加到AWS::SQS::Queue.

存取類別

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.amazonaws.com"
      },
      "Resource": "%{Destination.Arn}",
      "Action": "sqs:SendMessage",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "%{Source.Arn}"
        }
      }
    }
  ]
}
```

AWS::SNS::Topic 設定為 AWS::Lambda::Function

策略類型

AWS::Lambda::Permission附加到AWS::Lambda::Function.

存取類別

Write

```
{
  "Action": "lambda:InvokeFunction",
  "Principal": "sns.amazonaws.com",
  "SourceArn": "%{Source.Arn}"
}
```

AWS::SQS::Queue 設定為 AWS::Lambda::Function

策略類型

附加至AWS::Lambda::Function角色的客戶管理政策。

存取類別

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sqs:DeleteMessage"
      ],
      "Resource": [
        "%{Source.Arn}"
      ]
    }
  ]
}
```

Read

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sqs:ReceiveMessage",
        "sqs:GetQueueAttributes"
      ],
      "Resource": [
        "%{Source.Arn}"
      ]
    }
  ]
}
```

AWS::S3::Bucket 設定為 AWS::Lambda::Function

策略類型

AWS::Lambda::Permission附加到AWS::Lambda::Function.

存取類別

Write

```
{
  "Action": "lambda:InvokeFunction",
  "Principal": "s3.amazonaws.com",
  "SourceArn": "%{Source.Arn}",
  "SourceAccount": "${AWS::AccountId}"
}
```

AWS::StepFunctions::StateMachine 設定為 AWS::Lambda::Function

策略類型

附加至AWS::StepFunctions::StateMachine角色的客戶管理政策。

存取類別

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeAsync",
        "lambda:InvokeFunction"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

AWS::StepFunctions::StateMachine 設定為 AWS::SNS::Topic

策略類型

附加至AWS::StepFunctions::StateMachine角色的客戶管理政策。

存取類別

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

AWS::StepFunctions::StateMachine 設定為 AWS::SQS::Queue

策略類型

附加至AWS::StepFunctions::StateMachine角色的客戶管理政策。

存取類別

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sqs:SendMessage"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

AWS::StepFunctions::StateMachine 設定為 AWS::S3::Bucket

策略類型

附加至AWS::StepFunctions::StateMachine角色的客戶管理政策。

存取類別

Read

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectAcl",
        "s3:GetObjectLegalHold",
        "s3:GetObjectRetention",
        "s3:GetObjectTorrent",
        "s3:GetObjectVersion",
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionForReplication",
        "s3:GetObjectVersionTorrent",
        "s3:ListBucket",
        "s3:ListBucketMultipartUploads",
        "s3:ListBucketVersions",
        "s3:ListMultipartUploadParts"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}/*"
      ]
    }
  ]
}
```

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:AbortMultipartUpload",
        "s3:DeleteObject",
        "s3:DeleteObjectVersion",
```

```
        "s3:PutObject",
        "s3:PutObjectLegalHold",
        "s3:PutObjectRetention",
        "s3:RestoreObject"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}/*"
      ]
    }
  ]
}
```

AWS::StepFunctions::StateMachine 設定為 AWS::DynamoDB::Table

策略類型

附加至AWS::StepFunctions::StateMachine角色的客戶管理政策。

存取類別

Read

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:Query",
        "dynamodb:Scan",
        "dynamodb:BatchGetItem",
        "dynamodb:ConditionCheckItem",
        "dynamodb:PartiQLSelect"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}/index/*"
      ]
    }
  ]
}
```

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:PutItem",
        "dynamodb:UpdateItem",
        "dynamodb:DeleteItem",
        "dynamodb:BatchWriteItem",
        "dynamodb:PartiQLDelete",
        "dynamodb:PartiQLInsert",
        "dynamodb:PartiQLUpdate"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}/index/*"
      ]
    }
  ]
}
```

AWS::StepFunctions::StateMachine 設定為 AWS::StepFunctions::StateMachine

策略類型

附加至AWS::StepFunctions::StateMachine角色的客戶管理政策。

存取類別

Read

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:DescribeExecution"
      ],
      "Resource": [
        "arn:${AWS::Partition}:states:${AWS::Region}:${AWS::AccountId}:execution:
%{Destination.Name}:*"
      ]
    },
```

```
    {
      "Effect": "Allow",
      "Action": [
        "events:DescribeRule"
      ],
      "Resource": [
        "arn:${AWS::Partition}:events:${AWS::Region}:${AWS::AccountId}:rule/
StepFunctionsGetEventsForStepFunctionsExecutionRule"
      ]
    }
  ]
}
```

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:StartExecution"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "states:StopExecution"
      ],
      "Resource": [
        "arn:${AWS::Partition}:states:${AWS::Region}:${AWS::AccountId}:execution:
%{Destination.Name}:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "events:PutTargets",
        "events:PutRule"
      ],
      "Resource": [
```

```
        "arn:${AWS::Partition}:events:${AWS::Region}:${AWS::AccountId}:rule/
StepFunctionsGetEventsForStepFunctionsExecutionRule"
      ]
    }
  ]
}
```

AWS::StepFunctions::StateMachine 設定為 AWS::Events::EventBus

策略類型

附加至AWS::StepFunctions::StateMachine角色的客戶管理政策。

存取類別

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:PutEvents"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

AWS::AppSync::DataSource 設定為 AWS::DynamoDB::Table

策略類型

附加至AWS::AppSync::DataSource角色的客戶管理政策。

存取類別

Read

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

```
      "Effect": "Allow",
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:Query",
        "dynamodb:Scan",
        "dynamodb:BatchGetItem",
        "dynamodb:ConditionCheckItem",
        "dynamodb:PartiQLSelect"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}/index/*"
      ]
    }
  ]
}
```

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:PutItem",
        "dynamodb:UpdateItem",
        "dynamodb:DeleteItem",
        "dynamodb:BatchWriteItem",
        "dynamodb:PartiQLDelete",
        "dynamodb:PartiQLInsert",
        "dynamodb:PartiQLUpdate"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}/index/*"
      ]
    }
  ]
}
```

AWS::AppSync::DataSource 設定為 AWS::Lambda::Function

策略類型

附加至AWS::AppSync::DataSource角色的客戶管理政策。

存取類別

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeAsync",
        "lambda:InvokeFunction"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}:*"
      ]
    }
  ]
}
```

AWS::AppSync::DataSource 設定為 AWS::Events::EventBus

策略類型

附加至AWS::AppSync::DataSource角色的客戶管理政策。

存取類別

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:PutEvents"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
```

```
  ]
}
```

AWS::AppSync::GraphQLApi 設定為 AWS::Lambda::Function

策略類型

AWS::Lambda::Permission附加到AWS::Lambda::Function.

存取類別

Write

```
{
  "Action": "lambda:InvokeFunction",
  "Principal": "appsync.amazonaws.com",
  "SourceArn": "arn:${AWS::Partition}:appsync:${AWS::Region}:${AWS::AccountId}:apis/
%{Source.ResourceId}"
}
```

安裝泊塢視窗以搭配使用 AWS SAMCLI

Docker是在您的機器上運行容器的應用程序。使用Docker， AWS SAM 可以提供類似於容器的本機環境，以建置、測試和偵錯您的無伺服器應用程式。 AWS Lambda

> **Note**
>
> Docker只有在本機測試應用程式，以及使用--use-container選項建置部署套件時才需要。

主題

- 安裝 Docker
- 後續步驟

安裝 Docker

請按照以下說明Docker在您的操作系統上安裝。

Linux

Docker 可以在許多不同的操作系統上使用，包括大多數現代 Linux 發行版，例如 CentOSDebian，和 Ubuntu。如需在特定作業系統Docker上安裝的相關資訊，請參閱 Docker 文件網站上的取得 Docker。

要安裝Docker在 Amazon Linux 2 或 Amazon Linux 2023

1. 更新已安裝的套裝服務，並在執行個體上封裝快取。

```
$ sudo yum update -y
```

2. 安裝最新的Docker社群版套件。

   - 對於 Amazon Linux 2，運行以下命令：

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

   - 對於 Amazon 2023，運行以下命令：

```
$ sudo yum install -y docker
```

3. 啟動 Docker 服務。

```
$ sudo service docker start
```

4. 將新增ec2-user至docker群組，以便您無需使用即可執行Docker命令sudo。

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

5. 登出後再重新登入，以取得新的docker群組權限。若要這麼做，請關閉目前的 SSH 終端機視窗，然後在新的執行個體中重新連線至執行個體。您的新 SSH 工作階段應具有適當的docker群組權限。
6. 驗證ec2-user可以在不使sudo用的情況下運行 Docker 命令。

```
$ docker ps
```

   您應該會看到下列輸出，確認 Docker 已安裝並執行：

```
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
```

> **Note**
>
> 在 Linux 上，若要使用與主機不同的指令集架構來建置和執行 Lambda 函數，還需要額外的步驟進行設定Docker。例如，若要在x86_64電腦上執行arm64函式，您可以執行下列命令來設定Docker協助程式：`docker run --rm --privileged multiarch/qemu-user-static --reset -p yes`

如果您在安裝時遇到問題Docker，請參閱排解安裝錯誤。或者，請前往 Docker 文件網站，參閱 Linux 安裝後步驟的疑難排解章節。

macOS

> **Note**
>
> Docker桌面是官方支持的，但是從 1.47.0 AWS SAMCLI 版開始，只要它們使用運行時，您就可以使用替代方案。Docker

1. 安裝 Docker

   在 macOS 塞拉利昂 10.12 或更高版本上Docker運行的 AWS SAMCLI支持. 如需如何安裝 Docker，請參閱「Docker文件」網站上的「安裝 Mac Docker 版桌面」。

2. 設定共用雲端硬碟

   AWS SAMCLI需要在共用磁碟機中列出專案目錄或任何父目錄。這可能需要文件共享。如需詳細資訊，請參閱Docker文件中的磁碟區掛載需要檔案共用疑難排解主題。

3. 驗證安裝

   安裝Docker完成後，請確認它是否正常運作。同時確認您可以從命令列執行Docker命令 (例如，`docker ps`)。您不需要安裝、擷取或拉取任何容器，這會根據需要自動 AWS SAMCLI執行此動作。

如果您在安裝時遇到問題Docker，如需更多疑難排解提示，請參閱Docker文件網站的疑難排解與診斷一節。

Windows

> **Note**
>
> AWS SAM 正式支持Docker桌面。但是，從 AWS SAMCLI版本 1.47.0 開始，只要它們使用運行時，您就可以使用替代方案。Docker

1. 安裝 Docker。

   Docker桌上型電腦支援最新的 Windows 作業系統。對於 Windows 的舊版本，可以使用「Docker工具箱」。請選擇您的 Windows 版本，以取得正確的Docker安裝步驟：

   - 若要安裝 Docker Windows 10，請參閱「Docker文件」網站上的「安裝視窗Docker桌面」。
   - 若要安Docker裝舊版 Windows，請參閱Docker工具箱 GitHub 存放庫上的Docker工具箱。
2. 設定您的共用雲端硬碟。

   AWS SAMCLI需要在共用磁碟機中列出專案目錄或任何父目錄。在某些情況下，您必須共享驅動器Docker才能正常運行。
3. 驗證安裝。

   安裝Docker完成後，請確認它是否正常運作。同時確認您可以從命令列執行Docker命令 (例如，docker ps)。您不需要安裝、擷取或拉取任何容器，這會根據需要自動 AWS SAMCLI執行此動作。

如果您在安裝時遇到問題Docker，如需更多疑難排解提示，請參閱Docker文件網站的疑難排解與診斷一節。

後續步驟

如需如何安裝 AWS SAMCLI，請參閱安裝 AWS SAMCLI。

映像儲存庫

AWS SAM 藉助建置容器映像，簡化無伺服器應用程式的持續整合與持續交付 (CI/CD) 工作。 AWS SAM 提供的映像檔包括指 AWS SAM 令列介面 (CLI) 和建置工具，適用於許多受支援的執行 AWS Lambda 階段。這可讓您更輕鬆地使用 AWS SAMCLI. 您可以將這些映像與 CI/CD 系統搭配使用，以自動化應用程式的建置和部 AWS SAM 署作業。如需範例，請參閱 使用 CI/CD 系統和管線進行部署。

AWS SAM 構建容器映像 URI 標記為 AWS SAMCLI包含在該映像中的版本。如果您指定未標記的 URI，則會使用最新版本。例如，`public.ecr.aws/sam/build-nodejs20.x`使用最新的影像。不過，`public.ecr.aws/sam/build-nodejs20.x:1.24.1`會使用包含 AWS SAM CLI 版本 1.24.1 的映像檔。

從版本 1.33.0 開始 AWS SAMCLI，`x86_64`和`arm64`容器映像都可用於支援的執行階段。如需詳細資訊，請參閱AWS Lambda 開發人員指南中的 Lambda 執行階段。

> **Note**
>
> 在版本 1.22.0 之前 AWS SAMCLI， DockerHub 是從中 AWS SAMCLI提取容器映像的預設存放庫。從版本 1.22.0 開始，默認存儲庫更改為 Amazon 彈性容器註冊表公共（Amazon ECR 公共）。若要從目前預設值以外的儲存庫提取容器映像檔，您可以搭配--build-image選項使用sam build指令。本主題結尾的範例說明如何使用 DockerHub 儲存庫映像建置應用程式。

影像儲存庫 URI

下表列出 Amazon ECR 公開建置容器映像的 URI，可用來建置和封裝無伺服器應用程式。 AWS SAM

> **Note**
>
> Amazon ECR 公共更換為 1.22.0 AWS SAMCLI 版本DockerHub開始。如果您使用的是較早版本的 AWS SAMCLI，建議您升級。

執行期	Amazon ECR Public
自訂執行階段 (AL2023)	公共.aw/桑/構建提供的.al2023
自訂執行階段 (AL2)	公共 .ecr.aw/Sam /構建提交的.al2
自訂執行時間	公共.ecr.aw/SAM/構建提供
Go 1.x	公共 .ecr.aw/ SAM /構建-go1.x
Java 21	公共 .EW /三/構建java21
Java 17	公共 .EW /三/構建java17

執行期	Amazon ECR Public
Java 11	公共 .EW /SAM/構建java11
爪哇 8 (阿尔 2)	公共 .EW /SAM/構建java8.al2
Java 8	公共 .EW /SAM/構建java8
。淨值 8	公共 .ecr.aw/ SAM /構建網8
.NET 7	公共 .ecr.aw/ SAM /構建網7
.NET 6	公共 .ecr.aw/ SAM /構建網6
Node.js 20	公共 .ecr.aw/ SAM /構建-nodejs20.x
Node.js 18	公共 .ecr.aw/ SAM /構建-節點 18.x
Node.js 16	公共 .ecr.aw/ SAM /構建-節點16.x
Python 3.12	公共 .EW /SAM/構建-蟒蛇 3.12
Python 3.11	公共 .EW /SAM/構建-蟒蛇 3.11
Python 3.10	公共 .EW /SAM/構建-蟒蛇 3.10
Python 3.9	公共 .EW /SAM/構建-python3.9
Python 3.8	公共 .EW /SAM/構建-python3.8
紅寶石	公共 .ecr.aw/ SAM /構建紅寶石 3.3
Ruby 3.2	公共 .ecr.aw/ SAM /構建紅寶石 3.2

範例

下列兩個指令範例會使用 DockerHub 儲存庫中的容器映像來建置應用程式：

使Node.js 20用從以下位置提取的容器映像建立應用程式DockerHub：

```
$ sam build --use-container --build-image public.ecr.aws/sam/build-nodejs20.x
```

使用從以下位置提取的Python 3.12容器映像構建函數資源 DockerHub：

```
$ sam build --use-container --build-image Function1=public.ecr.aws/sam/build-python3.12
```

逐步部署無伺服器應用程

AWS Serverless Application Model (AWS SAM) 內建提CodeDeploy供逐步 AWS Lambda 部署。只需幾行配置，就可以為您 AWS SAM 執行以下操作：

- 部署 Lambda 函數的新版本，並自動建立指向新版本的別名。
- 逐漸將客戶流量轉移到新版本，直到您滿意其正常運作為止。如果更新無法正常運作，您可以復原變更。
- 定義流量前和流量後測試功能，以驗證新部署的程式碼是否正確設定，以及應用程式是否如預期般運作。
- 如果觸發 CloudWatch 警報，則自動復原部署。

> **Note**
>
> 如果您透過 AWS SAM 範本啟用漸進式部署，系統會自動為您建立 CodeDeploy 資源。您可以直接透過檢視 CodeDeploy 資源 AWS Management Console。

範例

下列範例示範如何使用逐步 CodeDeploy 將客戶轉移至您新部署的 Lambda 函數版本：

```
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
```

AWS SAM 範本的這些修訂會執行下列作業：

- `AutoPublishAlias`：通過添加此屬性並指定別名， AWS SAM：
  - 根據 Lambda 函數之 Amazon S3 URI 的變更，偵測何時部署新程式碼。
  - 使用最新的代碼創建並發布該函數的更新版本。
  - 使用您提供的名稱建立別名 (除非已存在別名)，並指向 Lambda 函數的更新版本。函式呼叫應該利用別名限定詞以充分善用此功能。如果您不熟悉 Lambda 函數版本控制和別名，請參閱AWS Lambda 函數版本控制和別名。
- `Deployment Preference Type`: 在上一個範例中，10% 的客戶流量會立即轉移到新版本。10 分鐘後，所有流量都轉移到新版本。但是，如果您的流量前或流量後測試失敗，或者觸發 CloudWatch 警報，請 CodeDeploy 復原部署。您可以透過下列方式指定流量在版本之間轉移的方式：
  - `Canary`：流量以兩個增量轉移。您可以從預定義的加那利選項中選擇。這些選項會指定在第一個增量中轉移至更新 Lambda 函數版本的流量百分比，以及在第二個增量中移動剩餘流量之前的間隔 (以分鐘為單位)。
  - `Linear`：流量以每個增量之間的相等分鐘數以同等增量轉移。您可以從預先定義的線性選項中進行選擇，以指定每個增量中移動的流量百分比以及每個增量之間的分鐘數。
  - `AllAtOnce`：所有流量都會立即從原始 Lambda 函數轉移到更新的 Lambda 函數版本。

  下表概述在範例中使用的選項以外可用的其他流量轉移選項。

部署偏好類型
Canary10Percent30Minutes
Canary10Percent5Minutes
Canary10Percent10Minutes
Canary10Percent15Minutes

部署偏好類型
線性 10 分鐘 PercentEvery
線性 10 PercentEvery 1 分鐘
線性 10 2 分鐘 PercentEvery
線性 10 3 分鐘 PercentEvery
AllAtOnce

- `Alarms`：這些是由部署引發的任何錯誤觸發的 CloudWatch 警示。遇到時，它們會自動回復您的部署。例如，如果您要部署的更新程式碼在應用程式中導致錯誤。另一個範例是，如果您指定的任何AWS Lambda或自訂 CloudWatch 量度超過警示臨界值。
- `Hooks`: 這些是交通前和交通後測試功能，可在流量轉移開始到新版本之前以及在流量轉換完成之前執行檢查。
  - `PreTraffic`：在流量轉移開始之前， CodeDeploy 叫用流量前掛鉤 Lambda 函數。此 Lambda 函數必須回呼 CodeDeploy 並指出成功或失敗。如果函數失敗，它會中止並將失敗報告給 AWS CloudFormation。如果功能成功，則 CodeDeploy 繼續流量轉移。
  - `PostTraffic`：在流量轉移完成之後， CodeDeploy 叫用流量後勾點 Lambda 函數。這類似於流量前掛鉤，其中函數必須回呼 CodeDeploy 以報告成功或失敗。使用後置流量掛勾執行整合測試或其他驗證動作。

  如需詳細資訊，請參閱安全部署的 SAM 參考。

第一次逐步部署 Lambda 函數

逐步部署 Lambda 函數時， CodeDeploy 需要先前部署的函數版本才能轉移流量。因此，您的第一個部署應該以兩個步驟完成：

- 步驟 1：部署您的 Lambda 函數，並使用`AutoPublishAlias`.
- 步驟 2：使用`DeploymentPreference`.

透過兩個步驟執行您的第一個逐步部署，可 CodeDeploy 提供先前的 Lambda 函數版本，以轉移流量。

步驟 1：部署您的 Lambda 函數

```
Resources:
MyLambdaFunction:
  Type: AWS::Serverless::Function
  Properties:
    Handler: index.handler
    Runtime: nodejs12.x
    CodeUri: s3://bucket/code.zip

    AutoPublishAlias: live
```

步驟 2：執行逐步部署

```
Resources:
MyLambdaFunction:
  Type: AWS::Serverless::Function
  Properties:
    Handler: index.handler
    Runtime: nodejs12.x
    CodeUri: s3://bucket/code.zip

    AutoPublishAlias: live

    DeploymentPreference:
      Type: Canary10Percent10Minutes
      Alarms:
        # A list of alarms that you want to monitor
        - !Ref AliasErrorMetricGreaterThanZeroAlarm
        - !Ref LatestVersionErrorMetricGreaterThanZeroAlarm
      Hooks:
        # Validation Lambda functions that are run before and after traffic shifting
        PreTraffic: !Ref PreTrafficLambdaFunction
        PostTraffic: !Ref PostTrafficLambdaFunction
```

進一步了解

如需設定逐步部署的實際操作範例，請參閱完整 AWS SAM 研討會中的單元 5-Canary 部署。

重要說明

本節包含 AWS Serverless Application Model (AWS SAM) 的重要注意事項和公告。

主題

2023 年的重要注意事項

2023 年 10 月

AWS SAMCLI中斷支援 Python 3.7

發佈於二零二零年十月二十日

Python 3.7在 2023 年六月收到 end-of-life 狀態。 AWS SAM CLI將於 2023 年 10 Python 3.7 月 24 日停止支援。如需詳細資訊，請參閱aws-sam-cli GitHub儲存庫中的公告。

此變更會影響下列使用者：

- 如果您使用Python 3.7並安裝 AWS SAM CLI通過`pip`.
- 如果您使用`aws-sam-cli`作為庫並使用Python 3.7.

如果您 AWS SAM CLI透過其他方法安裝和管理，則不會受到影響。

對於受影響的使用者，我們建議您將開發環境升級至Python 3.8或更新版本。

此變更不會影響Python 3.7 AWS Lambda 執行階段環境的支援。如需詳細資訊，請參閱AWS Lambda 開發人員指南中的執行階段淘汰政策。

二零二零年重要事項

2020 年 6 月

AWS SAMCLI在 32 位上安裝 Windows

對 32 位元視窗 AWS SAMCLI的 Support 很快就會被淘汰。如果您在 32 位元系統上操作，我們建議您升級到 64 位元系統，然後遵循中的指示安裝 AWS SAMCLI。

如果您無法升級到 64 位元系統，您可以在 32 位元系統 AWS SAMCLI上使用舊版 Docker 工具箱。但是，這將導致您遇到的某些限制 AWS SAMCLI。例如，您無法在 32 位元系統上執行 64 位元 Docker 容器。因此，如果您的 Lambda 函數依賴於 64 位元本機編譯的容器，您將無法在 32 位元系統上進行本機測試。

若要在 32 位元系統 AWS SAMCLI上安裝，請執行下列指令：

```
pip install aws-sam-cli
```

> **Important**
>
> 雖然此命`pip install aws-sam-cli`令也適用於 64 位元 Windows，但我們建議您使用 64 位元 MSI 安裝 AWS SAMCLI在 64 位元系統上。

無伺服器應用程式

下列範例說明如何下載、測試和部署許多額外的無伺服器應用程式，包括如何設定事件來源和資源。AWS

主題

- 處理動 DynamoDB 事件
- 處理 Amazon S3 事件

處理動 DynamoDB 事件

使用此範例應用程式，您可以根據您在概觀和快速入門指南中學到的內容建置，並安裝另一個範例應用程式。此應用程式由 DynamoDB 表格事件來源叫用的 Lambda 函數組成。Lambda 函數非常簡單 — 它會記錄透過事件來源訊息傳入的資料。

本練習說明如何模擬在叫用 Lambda 函數時傳遞給 Lambda 函數的事件來源訊息。

開始之前

請確定您已在中完成所需的設定安裝 AWS SAMCLI。

步驟 1：初始化應用程序

在本節中，您將下載包含 AWS SAM 範本和應用程式程式碼的應用程式套件。

初始化應用程式

1. 在命令提示字元中執行下列 AWS SAMCLI命令。

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

   請注意，gh:在上面的命令中被擴展到 GitHub url `https://github.com/`。

2. 檢閱命令所建立目錄的內容 (`dynamodb_event_reader/`)：

- `template.yaml`— 定義讀取 DynamoDB 應用程式需要的兩個 AWS 資源：Lambda 函數和一個 DynamoDB 表。範本也會定義兩個資源之間的對應。
- `read_dynamodb_event/`目錄 — 包含 DynamoDB 應用程式程式碼。

步驟 2：在本機測試應用程式

對於本機測試，請使 AWS SAMCLI用產生 DynamoDB 事件範例並叫用 Lambda 函數：

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

此命`generate-event`令會建立測試事件來源訊息，例如將所有元件部署至 AWS 雲端時所建立的訊息。此事件來源訊息會傳送至 Lambda 函數 ReadDynamo DBEvent。

根據中的原始程式碼，確認預期的訊息已列印到主控台`app.py`。

步驟 3：打 Package 應用程序

在本機測試應用程式之後，您可 AWS SAMCLI以使用建立部署套件，以便將應用程式部署到 AWS 雲端。

若要建立 Lambda 部署套件

1. 在要儲存封裝程式碼的位置建立一個 S3 儲存貯體。如果您想使用現有的 S3 儲存貯體，請跳過此步驟。

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

2. 在命令提示字元中執行下列 `package` CLI 命令，以建立部署套件。

   ```
   sam package \
       --template-file template.yaml \
       --output-template-file packaged.yaml \
       --s3-bucket bucketname
   ```

   當您在下一個步驟中部署應用程式時`packaged.yaml`，請指定新的範本檔案。

步驟 4：部署應用程式

現在您已經建立了部署套件，您可以使用它將應用程式部署到 AWS 雲端。然後測試應用程式。

將無伺服器應用程式部署到雲端 AWS

- 在中 AWS SAMCLI，使用 `deploy` CLI 命令部署您在範本中定義的所有資源。

```
sam deploy \
    --template-file packaged.yaml \
    --stack-name sam-app \
    --capabilities CAPABILITY_IAM \
    --region us-east-1
```

在命令中，`--capabilities`參數允 AWS CloudFormation許創建 IAM 角色。

AWS CloudFormation 會建立範本中定義的 AWS 資源。您可以在 AWS CloudFormation 主控台中存取這些資源的名稱。

在雲端測試無伺服器應用程式 AWS

1. 開啟 DynamoDB 主控台。
2. 將記錄插入您剛剛建立的表格中。
3. 移至表格的「測量結果」頁籤，然後選擇檢視所有 CloudWatch 測量結果。在主 CloudWatch 控台中，選擇 [記錄檔] 以檢視記錄輸出。

後續步驟

該 AWS SAM GitHub 存儲庫包含其他示例應用程序供您下載和實驗。若要存取此儲存庫，請參閱AWS SAM 範例應用程式。

處理 Amazon S3 事件

使用此示例應用程序，您可以基於您在前面示例中學到的內容，並安裝更複雜的應用程序。此應用程式包含由 Amazon S3 物件上傳事件來源叫用的 Lambda 函數組成。本練習說明如何透過 Lambda 函數存取 AWS 資源和進行 AWS 服務呼叫。

此範例無伺服器應用程式會處理 Amazon S3 中的物件建立事件。對於上傳到儲存貯體的每個映像，Amazon S3 會偵測物件建立的事件並叫用 Lambda 函數。Lambda 函數會叫用 Amazon Rekognition 來偵測影像中的文字。然後，它會將 Amazon Rekognition 傳回的結果儲存在 DynamoDB 表格中。

> **Note**
>
> 在此範例應用程式中，執行步驟的順序與先前範例略有不同。原因是此範例需要先建立資 AWS 源並設定 IAM 許可，才能在本機測試 Lambda 函數。我們將利用創 AWS CloudFormation 建資源並為您配置權限。否則，您需要手動執行此操作，然後才能在本機測試 Lambda 函數。
>
> 由於此範例較為複雜，因此在執行此範例之前，請確定您已熟悉先前的範例應用程式的安裝方式。

開始之前

請確定您已在中完成所需的設定安裝 AWS SAMCLI。

步驟 1：初始化應用程序

在本節中，您將下載範例應用程式，該應用程式包含 AWS SAM 範本和應用程式程式碼。

初始化應用程式

1. 在命令提示字元中執行下列 AWS SAMCLI命令。

```
sam init \
--location https://github.com/aws-samples/cookiecutter-aws-sam-s3-rekognition-
dynamodb-python \
--no-input
```

2. 檢閱命令所建立目錄的內容 (`aws_sam_ocr/`)：

   - `template.yaml`— 定義 Amazon S3 應用程式需要的三個 AWS 資源：Lambda 函數、一個 Amazon S3 儲存貯體和一個 DynamoDB 表。該模板還定義了這些資源之間的映射和權限。
   - `src/`目錄 — 包含 Amazon S3 應用程式程式碼。
   - `SampleEvent.json`— 範例事件來源，用於本機測試。

步驟 2：打 Package 應用程序

在本機測試此應用程式之前，您必須使用建立部署套件，以便將應用程式部署到 AWS 雲端。 AWS SAMCLI此部署會建立必要的 AWS 資源和權限，以便在本機測試應用程式。

若要建立 Lambda 部署套件

1. 在要儲存封裝程式碼的位置建立一個 S3 儲存貯體。如果您想使用現有的 S3 儲存貯體，請跳過此步驟。

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

2. 在命令提示字元中執行下列 package CLI 命令，以建立部署套件。

   ```
   sam package \
       --template-file template.yaml \
       --output-template-file packaged.yaml \
       --s3-bucket bucketname
   ```

   當您在下一個步驟中部署應用程式時packaged.yaml，請指定新的範本檔案。

步驟 3：部署應用程式

現在您已經建立了部署套件，您可以使用它將應用程式部署到 AWS 雲端。然後，您可以在 AWS 雲端中呼叫應用程式來測試應用程式。

將無伺服器應用程式部署到雲端 AWS

- 在中 AWS SAMCLI，使用命deploy令部署您在範本中定義的所有資源。

  ```
  sam deploy \
      --template-file packaged.yaml \
      --stack-name aws-sam-ocr \
      --capabilities CAPABILITY_IAM \
      --region us-east-1
  ```

  在命令中，該--capabilities參數 AWS CloudFormation 允許創建 IAM 角色。

AWS CloudFormation 會建立範本中定義的 AWS 資源。您可以在 AWS CloudFormation 主控台中存取這些資源的名稱。

在雲端測試無伺服器應用程式 AWS

1. 將映像上傳到您為此範例應用程式建立的 Amazon S3 儲存貯體。
2. 開啟 DynamoDB 主控台並尋找已建立的資料表。如需 Amazon Rekognition 傳回的結果，請參閱表格。
3. 確認 DynamoDB 表格包含新記錄，其中包含在上傳的映像中找到的 Amazon Rekognition 文字。

步驟 4：在本機測試應用程式

您必須先擷取由建立的 AWS 資源名稱，才能在本機測試應用程式 AWS CloudFormation。

- 從中擷取 Amazon S3 金鑰名稱和儲存貯體名稱 AWS CloudFormation。透過取代物件索引鍵、值區名稱和值區 ARN 的值來修改`SampleEvent.json`檔案。
- 擷取 DynamoDB 料表名稱。此名稱用於以下`sam local invoke`命令。

使 AWS SAMCLI用產生範例 Amazon S3 事件並叫用 Lambda 函數：

```
TABLE_NAME=Table name obtained from AWS CloudFormation console sam local invoke --event
 SampleEvent.json
```

此`TABLE_NAME=`部分會設定 DynamoDB 資料表名稱。此`--event`參數會指定包含要傳遞至 Lambda 函數之測試事件訊息的檔案。

您現在可以根據 Amazon Rekognition 傳回的結果，確認已建立預期的 DynamoDB 記錄。

後續步驟

該 AWS SAM GitHub 存儲庫包含其他示例應用程序供您下載和實驗。若要存取此儲存庫，請參閱AWS SAM 範例應用程式。

AWS SAMCLITerraform支持

本節介紹了使用 AWS Serverless Application Model 命令行界面（AWS SAMCLI）與您的Terraform項目和Terraform雲。

若要提供意見反應並提交功能要求，請建立GitHub問題。

主題

什麼是 AWS SAMCLI支持Terraform？

Terraform Cloud對您的專案使用 AWS Serverless Application Model 命令列介面 (AWS SAMCLI)，或執行下列Terraform項目的本機除錯和測試：

- AWS Lambda 功能和圖層。
- Amazon API Gateway API。

如需簡介Terraform，請參閱「什麼是Terraform？」 在網HashiCorpTerraform站上。

若要提供意見反應並提交功能要求，請建立GitHub問題。

> **Note**
>
> 作為集成的解析步驟 AWS SAMCLI的一部分， AWS SAMCLI流程用戶命令生成項目文件和數據。命令輸出應該保持不變，但在某些環境中，環境或運行程序可能會在輸出中插入其他日誌或信息。

主題

什麼是 AWS SAMCLI？

這 AWS SAMCLI是一個命令列工具，您可以搭配 AWS SAM 範本和支援的第三方整合使用，例如Terraform建置和執行無伺服器應用程式。如需「」的簡介 AWS SAMCLI，請參閱什麼是 AWS SAMCLI？。

支 AWS SAMCLI援下列指令Terraform：

- `sam local invoke`— 在本機啟動 AWS Lambda 函式資源的一次性叫用。若要瞭解有關此指令的更多資訊，請參閱測試簡介 sam local invoke。
- `sam local start-api`— 在本機執行 Lambda 資源，並透過本機 HTTP 伺服器主機進行測試。這種類型的測試對於 API Gateway 端點叫用的 Lambda 函數很有幫助。若要瞭解有關此指令的更多資訊，請參閱測試簡介 sam local start-api。
- `sam local start-lambda`— 啟動 Lambda 函數的本機端點，以便使用 AWS Command Line Interface (AWS CLI) 或 SDK 在本機叫用函數。若要瞭解有關此指令的更多資訊，請參閱測試簡介 sam local start-lambda。

我如何使用 AWS SAMCLI與Terraform？

核心Terraform工作流程包含三個階段：寫入、規劃和套用。透過 AWS SAMCLI支援Terraform，您可以利用指令 AWS SAMCLI`sam local`集，同時繼續使用Terraform工作流程來管理應用程式 AWS。一般來說，這意味著以下內容：

- 撰寫 — 將您的基礎結構編寫為程式碼使用Terraform.
- 測試和偵錯 — 使用在本 AWS SAMCLI機測試和偵錯您的應用程式。
- 計劃 — 在套用前預覽變更。
- 套用 — 佈建您的基礎架構。

如需 AWS SAMCLI搭配使用的範例Terraform，請參閱「一起更好：」 AWS SAMCLI 和「AWS 計算部HashiCorp Terraform落格」。

後續步驟

若要完成所有必要條件並進行設定Terraform，請參閱開始使用的Terraform支援 AWS SAMCLI。

開始使用的Terraform支援 AWS SAMCLI

本主題介紹如何開始使用指 AWS Serverless Application Model 令行介面 (AWS SAMCLI) 與 Terraform。

若要提供意見反應並提交功能要求，請建立GitHub問題。

主題

AWS SAMCLITerraform前提

完成所有先決條件，即可開始 AWS SAMCLI與您的Terraform專案搭配使用。

1. 安裝或升級 AWS SAMCLI

   要檢查是否已 AWS SAMCLI安裝，請運行以下命令：

   ```
   $ sam --version
   ```

   如果 AWS SAMCLI已安裝，輸出將顯示版本。若要升級至最新版本，請參閱升級 AWS SAMCLI。

   如需有關安裝 AWS SAMCLI及其所有先決條件的指示，請參閱安裝 AWS SAMCLI。
2. 安裝 Terraform

   要檢查是否已Terraform安裝，請運行以下命令：

   ```
   $ terraform -version
   ```

   若要安裝Terraform，請參閱Terraform在Terraform登錄中安裝。
3. 安裝進Docker行本地測試

   本地測試Docker的 AWS SAMCLI要求。若要安裝Docker，請參閱安裝泊塢視窗以搭配使用 AWS SAMCLI。

使用 AWS SAMCLI指令搭配 Terraform

當您執行支援的 AWS SAMCLI命令時，請使用`--hook-name`選項並提供`terraform`值。以下是範例：

```
$ sam local invoke --hook-name terraform
```

您可以在 AWS SAMCLI組態檔案中使用下列項目配置此選項：

```
hook_name = "terraform"
```

為Terraform專案設定

完成本主題中的步驟以使用 AWS SAMCLI與Terraform專案搭配使用。

如果您在項目之外構建 AWS Lambda 成品，則不需要額外的Terraform設置。請參使用 AWS SAMCLI與進Terraform行本機除錯和測試閱以開始使用 AWS SAMCLI。

如果您在Terraform專案中建立 Lambda 成品，則必須執行下列動作：

1. 安裝 Python 3.8 或更新版本
2. 安裝工Make具。
3. 在Terraform專案中定義 Lambda 成品建置邏輯。
4. 定義`sam metadata`資源以通知您 AWS SAMCLI的構建邏輯。
5. 使用此命 AWS SAMCLI`sam build`令建立您的 Lambda 成品。

安裝 Python 3.8 或更新版本

Python需要 3.8 或更新版本才能搭配使用 AWS SAMCLI。當您執行時`sam build`， AWS SAMCLI會`makefiles`建立包含用來建置 Lambda 成品之Python命令的建立項目。

如需安裝說明，請參閱 Python 初學者指南中的下載 Python。

請執行下列指令，確認 Python 3.8 或更新版本已新增至您的電腦路徑：

```
$ python --version
```

輸出應該顯示一個版本的 Python 是 3.8 或更高版本。

安裝工Make具

GNUMake是一種控制項目的可執行文件和其他非源文件的生成的工具。依賴此工具來建 AWS SAMCLI立makefiles您的 Lambda 成品的建立項目。

如果您尚未Make在本地計算機上安裝，請在繼續之前進行安裝。

對於視窗, 您可以使用安裝巧克力. 如需指示，請參閱「如何在 Windows 中安裝和使用「Make」中的「使用巧克力」。

定義 Lambda 成品建置邏輯

使用資null_resourceTerraform源類型定義您的 Lambda 建置邏輯。以下是使用自訂建置指令碼建置 Lambda 函數的範例。

```
resource "null_resource" "build_lambda_function" {
    triggers = {
        build_number = "${timestamp()}"
    }

    provisioner "local-exec" {
        command =  substr(pathexpand("~"), 0, 1) == "/"? "./
py_build.sh \"${local.lambda_src_path}\" \"${local.building_path}\"
 \"${local.lambda_code_filename}\" Function" : "powershell.exe -File .\\PyBuild.ps1
 ${local.lambda_src_path} ${local.building_path} ${local.lambda_code_filename}
 Function"
    }
}
```

定義資sam metadata源

資sam metadata源是一種null_resourceTerraform資源類型，提 AWS SAMCLI供尋找 Lambda 成品所需的資訊。專案中的每個 Lambda 函數或層都需要唯一的sam metadata資源。若要深入了解此資源類型，請參閱登錄中的 null_resource。Terraform

若要定義資sam metadata源

1. 為您的資源命名，sam_metadata_以將資源識別為sam metadata資源。
2. 在資源triggers區塊內定義 Lambda 成品屬性。
3. 使用null_resourcedepends_on引數指定包含 Lambda 建置邏輯的內容。

   以下是範例範本：

```
resource "null_resource" "sam_metadata_..." {
  triggers = {
    resource_name = resource_name
    resource_type = resource_type
    original_source_code = original_source_code
    built_output_path = built_output_path
  }
  depends_on = [
    null_resource.build_lambda_function # ref to your build logic
  ]
}
```

以下是一個示例sam metadata資源：

```
resource "null_resource" "sam_metadata_aws_lambda_function_publish_book_review" {
    triggers = {
        resource_name = "aws_lambda_function.publish_book_review"
        resource_type = "ZIP_LAMBDA_FUNCTION"
        original_source_code = "${local.lambda_src_path}"
        built_output_path = "${local.building_path}/${local.lambda_code_filename}"
    }
    depends_on = [
        null_resource.build_lambda_function
    ]
}
```

sam metadata資源的內容會根據 Lambda 資源類型 (函數或層) 以及封裝類型 (ZIP 或影像) 而有所不同。若要取得更多資訊和範例，請參閱〈〉sam 中繼資料資源。

當您設定sam metadata資源並使用支援的 AWS SAMCLI命令時， AWS SAMCLI會在執行命 AWS SAMCLI令之前產生中繼資料檔案。產生此檔案後，您可以將此--skip-prepare-infra選項與 future 的 AWS SAMCLI指令搭配使用，以略過中繼資料產生程序並節省時間。只有在您尚未進行任何基礎設施變更 (例如建立新的 Lambda 函數或新的 API 端點) 時，才應使用此選項。

使用建置 AWS SAMCLI您的 Lambda 成品

使用此命 AWS SAMCLIsam build令建立您的 Lambda 成品。當您執行時sam build， AWS SAMCLI會執行下列動作：

1. 尋找Terraform專案中的sam metadata資源，以瞭解並找出您的 Lambda 資源。

2. 啟動 Lambda 建置邏輯以建置您的 Lambda 成品。
3. 建立組織Terraform專案以與 AWS SAMCLIsam local指令搭配使用的.aws-sam目錄。

要建立與山姆構建

1. 從包含Terraform根模塊的目錄中，運行以下命令：

   ```
   $ sam build --hook-name terraform
   ```

2. 若要建立特定的 Lambda 函數或層，請執行下列命令

   ```
   $ sam build --hook-name terraform lambda-resource-id
   ```

   Lambda 資源識別碼可以是 Lambda 函數名稱或完整Terraform資源地址，例如aws_lambda_function.list_books或module.list_book_function.aws_lambda_functi

如果您的函數源代碼或其他Terraform配置文件位於包含Terraform根模塊的目錄之外，則需要指定位置。使用--terraform-project-root-path此選項可指定包含這些檔案之頂層目錄的絕對路徑或相對路徑。以下是範例：

```
$ sam build --hook-name terraform --terraform-project-root-path ~/projects/terraform/
demo
```

使用容器建置

執行命 AWS SAMCLIsam build令時，您可以設定 AWS SAMCLI為使用本機Docker容器建置應用程式。

> **Note**
>
> 您必須已Docker安裝並設定。如需說明，請參閱安裝泊塢視窗以搭配使用 AWS SAMCLI。

若要使用容器建置

1. 建立Dockerfile包含TerraformPython、和Make工具的。您也應該包含您的 Lambda 函數執行階段。

   下面是一個例子Dockerfile：

```
FROM public.ecr.aws/amazonlinux/amazonlinux:2

RUN yum -y update \
    && yum install -y unzip tar gzip bzip2-devel ed gcc gcc-c++ gcc-gfortran \
    less libcurl-devel openssl openssl-devel readline-devel xz-devel \
    zlib-devel glibc-static libcxx libcxx-devel llvm-toolset-7 zlib-static \
    && rm -rf /var/cache/yum

RUN yum -y install make \
    && yum -y install zip

RUN yum install -y yum-utils \
    && yum-config-manager --add-repo https://rpm.releases.hashicorp.com/
AmazonLinux/hashicorp.repo \
    && yum -y install terraform \
    && terraform --version

# AWS Lambda Builders
RUN amazon-linux-extras enable python3.8
RUN yum clean metadata && yum -y install python3.8
RUN curl -L get-pip.io | python3.8
RUN pip3 install aws-lambda-builders
RUN ln -s /usr/bin/python3.8 /usr/bin/python3
RUN python3 --version

VOLUME /project
WORKDIR /project

ENTRYPOINT ["sh"]
```

2. 用docker build來建立您的Docker映像檔。

   以下是範例：

   ```
   $ docker build --tag terraform-build:v1 <path-to-directory-containing-Dockerfile>
   ```

3. 使用--use-container和--build-image選項執行 AWS SAMCLIsam build命令。

   以下是範例：

   ```
   $ sam build --use-container --build-image terraform-build:v1
   ```

後續步驟

若要開始 AWS SAMCLI與您的Terraform專案搭配使用，請參閱使用 AWS SAMCLI與進Terraform行本機除錯和測試。

設定 Terraform Cloud

我們建議您使用Terraform v1.6.0或更新版本。如果您使用的是舊版本，則必須在本端產生Terraform計畫檔。本地計劃文件提供 AWS SAM CLI了執行本地測試和調試所需的信息。

產生本端平面檔的步驟

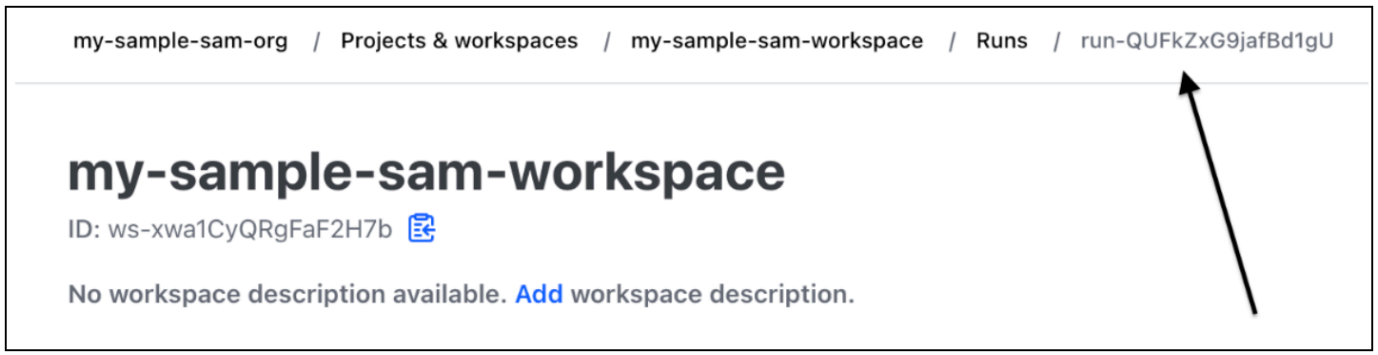
> **Note**
>
> 這些步驟不是Terraform v1.6.0或更新的必要步驟。若要開始使用 AWS SAM CLI與Terraform Cloud，請參閱 AWS SAMCLI搭配使用 Terraform。

1. 配置 API 令牌 — 令牌的類型將取決於您的訪問級別。要了解更多信息，請參閱文檔中的 Terraform CloudAPI 令牌。
2. 設置您的 API 令牌環境變量-以下是命令行中的示例：

   ```
   $ export TOKEN="<api-token-value>"
   ```

3. 取得您的執行 ID — 從Terraform Cloud主控台，找出您要與之搭配使用的Terraform執行 ID AWS SAMCLI。

   執行 ID 位於執行階段的階層連結路徑中。

4. 獲取計劃文件 — 使用您的 API 令牌，獲取本地計劃文件。以下是命令列中的範例：

   ```
   curl \
   ```

```
    --header "Authorization: Bearer $TOKEN" \
    --header "Content-Type: application/vnd.api+json" \
    --location \
    https://app.terraform.io/api/v2/runs/<run ID>/plan/json-output \
    > custom_plan.json
```

您現在可以使用 AWS SAMCLI與Terraform Cloud。使用支援的指 AWS SAMCLI令時，請使用`--terraform-plan-file`選項來指定本端平面檔的名稱和路徑。以下是範例：

```
$ sam local invoke --hook-name terraform --terraform-plan-file custom-plan.json
```

以下是使用`sam local start-api`指令的範例：

```
$ sam local start-api --hook-name terraform --terraform-plan-file custom-plan.json
```

如需可搭配這些範例使用的範例應用程式，請參閱 AWS 範例儲存庫中的 api_gateway_v2_tf_cloud。GitHub

後續步驟

若要開始使用 AWS SAMCLI與Terraform Cloud，請參閱使用 AWS SAMCLI與進Terraform行本機除錯和測試。

使用 AWS SAMCLI與進Terraform行本機除錯和測試

本主題介紹如何在Terraform專案和中使用支援的指 AWS Serverless Application Model 令行介面 (AWS SAMCLI) 指令Terraform Cloud。

若要提供意見反應並提交功能要求，請建立GitHub問題。

主題

- 本地測試 sam local invoke
- 本地測試 sam local start-api
- 本地測試 sam local start-lambda
- Terraform 限制

本地測試 sam local invoke

> **Note**
>
> 若要使用在本機 AWS SAMCLI進行測試，您必須安裝並設定 Docker。如需說明，請參閱安裝泊塢視窗以搭配使用 AWS SAMCLI。

以下是透過傳入事件在本機測試 Lambda 函數的範例：

```
$ sam local invoke --hook-name terraform hello_world_function -e events/event.json -
```

若要瞭解有關使用此指令的更多資訊，請參閱測試簡介 sam local invoke。

本地測試 sam local start-api

若要`sam local start-api`搭配使用Terraform，請執行下列命令：

```
$ sam local start-api --hook-name terraform
```

以下是範例：

```
$ sam local start-api --hook-name terraform




Running Prepare Hook to prepare the current application

Executing prepare hook of hook "terraform"

Initializing Terraform application

...
```

```
Creating terraform plan and getting JSON output


....
Generating metadata file




Unresolvable attributes discovered in project, run terraform apply to resolve them.




Finished generating metadata file. Storing in...
Prepare hook completed and metadata file generated at: ...
Mounting HelloWorldFunction at http://127.0.0.1:3000/hello [GET]


Mounting None at http://127.0.0.1:3000/hello [POST]


You can now browse to the above endpoints to invoke your functions. You do not need
 to restart/reload SAM CLI while working on your functions, changes will be reflected
 instantly/automatically. If you
used sam build before running local commands, you will need to re-run sam build for the
 changes to be picked up. You only need to restart SAM CLI if you update your AWS SAM
 template
2023-06-26 13:21:20  * Running on http://127.0.0.1:3000/ (Press CTRL+C to quit)
```

若要瞭解有關此指令的更多資訊，請參閱測試簡介 sam local start-api。

使用 Lambda 授權器的 Lambda 函數

對於設定為使用 Lambda 授權器的 Lambda 函數， AWS SAMCLI會在叫用 Lambda 函數端點之前自動叫用您的 Lambda 授權器。

- 若要進一步瞭解中的此功能 AWS SAMCLI，請參閱使用 Lambda 授權器的 Lambda 函數。
- 如需有關使用 Lambda 授權者的詳細資訊Terraform，請參閱Terraform登錄Resource: aws_api_gateway_authorizer中的。

本地測試 sam local start-lambda

以下是使用 AWS Command Line Interface (AWS CLI) 在本機測試 Lambda 函數的範例：

1. 使用建 AWS SAMCLI立本機測試環境：

```
$ sam local start-lambda --hook-name terraform hello_world_function
```

2. 使用在本 AWS CLI 地調用您的函數：

```
$ aws lambda invoke --function-name hello_world_function --endpoint-
url http://127.0.0.1:3001/ response.json --cli-binary-format raw-in-base64-out --
payload file://events/event.json
```

若要瞭解有關此指令的更多資訊，請參閱測試簡介 sam local start-lambda。

Terraform 限制

以下是搭配使用時 AWS SAMCLI的限制Terraform：

- 連結至多個圖層的 Lambda 函數。
- Terraform定義資源之間連結的區域變數。
- 引用尚未創建的 Lambda 函數。這包括在 REST API 資源的主體屬性中定義的函數。

若要避免這些限制，您可以在新增資源`terraform apply`時執行。

使用 AWS SAMCLI與無伺服器 .tf 進行本機偵錯和測試

命 AWS Serverless Application Model 令列介面 (AWS SAMCLI) 可與無伺服器 .tf 模組搭配使用，以進行本機除錯和測試 AWS Lambda 函數和層。支援下列 AWS SAMCLI指令：

- `sam build`
- `sam local invoke`
- `sam local start-api`
- `sam local start-lambda`

> **Note**
>
> 無伺服器 .tf 4.6.0 及更新版本支援整合。 AWS SAMCLI

若要開始使用無伺服器 .tf 模組，請更新至最新版本的無伺服器 .tf AWS SAMCLI 和. AWS SAMCLI

從無伺服器 .tf 6.0.0 版開始，您必須將參數設定為。`create_sam_metadata` `true`這會產生命 AWS SAMCLI`sam build`令所需的中繼資料資源。

若要進一步了解Serverless.tf，請參閱terraform-aws-lambda-module。

AWS SAMCLI與Terraform參考

本節是將指 AWS Serverless Application Model 令行介面 (AWS SAMCLI) 搭配Terraform使用進行本機除錯和測試的參考。

若要提供意見反應並提交功能要求，請建立GitHub問題。

AWS SAM 支援的特徵參考

Terraform您可以在這裡找到支援與搭配使用之 AWS SAMCLI功能的參考文件：

- sam build
- sam local invoke
- sam local start-api
- sam local start-lambda

Terraform特定參考

您可以在這裡找到特定使 AWS SAMCLITerraform用 for 的參考文件：

- sam 中繼資料資源

sam 中繼資料資源

此頁面包含與Terraform專案搭配使用之sam metadata resource資源類型的參考資訊。

- 若要取得將 AWS Serverless Application Model 指令行介面 (AWS SAMCLI) 與配合使用的簡介 Terraform，請參閱〈〉什麼是 AWS SAMCLI支持Terraform？。
- 若要使用 AWS SAMCLI與Terraform，請參閱使用 AWS SAMCLI與進Terraform行本機除錯和測試。

主題

- 引數
- 範例

引數

引數	描述
built_output_path	AWS Lambda 函數構建工件的路徑。
docker_build_args	碼頭構建參數 JSON 對象的解碼字符串。此為選用引數。
docker_context	包含 Docker 映像構建上下文的目錄路徑。
docker_file	碼頭檔案的路徑。此路徑是相對於docker_context 路徑的路徑。 此為選用引數。預設值為 Dockerfile 。
docker_tag	創建的碼頭圖像標記的值。此值是選用的。
depends_on	Lambda 函數或層的建置資源路徑。若要深入了解，請depends_on參閱登錄中的Terraform引數。
original_source_code	定義 Lambda 函數的路徑。該值可以是字符串，字符串數組或解碼的 JSON 對象作為字符串。 • 對於字串陣列，只會使用第一個值，因為不支援多個程式碼路徑。 • 對於 JSON 物件，也source_code_property 必須定義。
resource_name	Lambda 函數名稱。
resource_type	Lambda 函數套件類型的格式。接受的值為： • IMAGE_LAMBDA_FUNCTION

引數	描述
	• LAMBDA_LAYER • ZIP_LAMBDA_FUNCTION
source_code_property	JSON 物件中 Lambda 資源程式碼的路徑。如果original_source_code 是 JSON 物件，請定義此屬性。

範例

sam 中繼資料資源參考使用 ZIP 套件類型的 Lambda 函數

```
# Lambda function resource
resource "aws_lambda_function" "tf_lambda_func" {
  filename = "${path.module}/python/hello-world.zip"
  handler = "index.lambda_handler"
  runtime = "python3.8"
  function_name = "function_example"
  role = aws_iam_role.iam_for_lambda.arn
  depends_on = [
    null_resource.build_lambda_function # function build logic
  ]
}

# sam metadata resource
resource "null_resource" "sam_metadata_function_example" {
  triggers = {
    resource_name = "aws_lambda_function.function_example"
    resource_type = "ZIP_LAMBDA_FUNCTION"
    original_source_code = "${path.module}/python"
    built_output_path = "${path.module}/building/function_example"
  }
  depends_on = [
    null_resource.build_lambda_function # function build logic
  ]
}
```

sam 中繼資料資源使用映像套件類型參考 Lambda 函數

```
resource "null_resource" "sam_metadata_function {
  triggers = {
```

```
    resource_name = "aws_lambda_function.image_function"
    resource_type = "IMAGE_LAMBDA_FUNCTION"
    docker_context = local.lambda_src_path
    docker_file = "Dockerfile"
    docker_build_args = jsonencode(var.build_args)
    docker_tag = "latest"
  }
}
```

sam 參考 Lambda 層的中繼資料資源

```
resource "null_resource" "sam_metadata_layer1" {
  triggers = {
    resource_name = "aws_lambda_layer_version.layer"
    resource_type = "LAMBDA_LAYER"
    original_source_code = local.layer_src
    built_output_path = "${path.module}/${layer_build_path}"
  }
  depends_on = [null_resource.layer_build]
}
```

在本機測試和建置 AWS CDK 應用程式 AWS SAMCLI

您可以使用在 AWS SAMCLI本機測試和建置使用定義的無伺服器應用程式 AWS Cloud Development Kit (AWS CDK)。因為在 AWS CDK 專案結構內 AWS SAMCLI工作，您仍然可以使用 AWS CDK Toolkit 來建立、修改和部署 AWS CDK 應用程式。

如需有關安裝和設定的資訊 AWS CDK，請參閱開AWS Cloud Development Kit (AWS CDK) 發人員指南 AWS CDK中的《入門使用》。

> **Note**
>
> 從版本 1.135.0 和第 AWS CDK 2 版開始，從 2.0.0 版開始 AWS SAMCLI支援 AWS CDK 第 1 版。

主題

開始使用 AWS SAM 和 AWS CDK

本主題說明 AWS SAMCLI搭配 AWS CDK 應用程式使用所需的內容，並提供建置和在本機測試簡單 AWS CDK 應用程式的指示。

必要條件

若要 AWS SAMCLI與一起使用 AWS CDK，您必須安裝和 AWS SAMCLI. AWS CDK

- 如需有關安裝的資訊 AWS CDK，請參閱開AWS Cloud Development Kit (AWS CDK) 發人員指南 AWS CDK中的〈開始使用〉。
- 如需有關安裝的資訊 AWS SAMCLI，請參閱安裝 AWS SAMCLI。

建立並在本機測試 AWS CDK 應用程式

若要使用本機測試 AWS CDK 應用程式 AWS SAMCLI，您必須擁有包含 Lambda 函數的 AWS CDK 應用程式。使用下列步驟建立具有 Lambda 函數的基本 AWS CDK 應用程式。如需詳細資訊，請參閱AWS Cloud Development Kit (AWS CDK) 開發人員指南 AWS CDK中的使用建立無伺服器應用程式。

> **Note**
>
> 從版本 1.135.0 和第 AWS CDK 2 版開始，從 2.0.0 版開始 AWS SAMCLI支援 AWS CDK 第 1 版。

步驟 1：建立 AWS CDK 應用程式

在本教學課程中， AWS CDK 請初始化使用 TypeScript.

要執行的命令：

AWS CDK v2

```
mkdir cdk-sam-example
cd cdk-sam-example
cdk init app --language typescript
```

AWS CDK v1

```
mkdir cdk-sam-example
cd cdk-sam-example
cdk init app --language typescript
npm install @aws-cdk/aws-lambda
```

步驟 2：將 Lambda 函數新增至您的應用程式

以下列項目取代中`lib/cdk-sam-example-stack.ts`的程式碼：

AWS CDK v2

```
import { Stack, StackProps } from 'aws-cdk-lib';
```

```
import { Construct } from 'constructs';
import * as lambda from 'aws-cdk-lib/aws-lambda';

export class CdkSamExampleStack extends Stack {
  constructor(scope: Construct, id: string, props?: StackProps) {
    super(scope, id, props);

    new lambda.Function(this, 'MyFunction', {
      runtime: lambda.Runtime.PYTHON_3_9,
      handler: 'app.lambda_handler',
      code: lambda.Code.fromAsset('./my_function'),
    });
  }
}
```

AWS CDK v1

```
import * as cdk from '@aws-cdk/core';
import * as lambda from '@aws-cdk/aws-lambda';

export class CdkSamExampleStack extends cdk.Stack {
  constructor(scope: Construct, id: string, props?: StackProps) {
    super(scope, id, props);

    new lambda.Function(this, 'MyFunction', {
      runtime: lambda.Runtime.PYTHON_3_9,
      handler: 'app.lambda_handler',
      code: lambda.Code.fromAsset('./my_function'),
    });
  }
}
```

步驟 3：新增您的 Lambda 函數程式碼

建立名為 my_function 的目錄。在該目錄中，建立名為 app.py 的檔案。

要執行的命令：

```
mkdir my_function
cd my_function
touch app.py
```

為 app.py 添加以下程式碼：

```
def lambda_handler(event, context):
    return "Hello from SAM and the CDK!"
```

步驟 4：測試您的 Lambda 函數

您可以使用在本 AWS SAMCLI機叫用您在應用 AWS CDK 程式中定義的 Lambda 函數。為此，您需要函數構造標識符和合成 AWS CloudFormation 模板的路徑。

要執行的命令：

```
cdk synth --no-staging
```

```
sam local invoke MyFunction --no-event -t ./cdk.out/CdkSamExampleStack.template.json
```

示例輸出：

```
Invoking app.lambda_handler (python3.9)

START RequestId: 5434c093-7182-4012-9b06-635011cac4f2 Version: $LATEST
"Hello from SAM and the CDK!"
END RequestId: 5434c093-7182-4012-9b06-635011cac4f2
REPORT RequestId: 5434c093-7182-4012-9b06-635011cac4f2 Init Duration: 0.32 ms Duration:
 177.47 ms Billed Duration: 178 ms Memory Size: 128 MB Max Memory Used: 128 MB
```

如需有關使用 AWS SAM CLI 測試 AWS CDK 應用程式的可用選項的詳細資訊，請參閱本地測試 AWS CDK 應用。

本地測試 AWS CDK 應用

您可以從 AWS CDK 應 AWS SAMCLI用程式的專案根目錄執行下列命令，使用在本機測試應用 AWS CDK 程式：

- sam local invoke
- sam local start-api
- sam local start-lambda

在您使 AWS CDK 用應用程式執行任何sam local命令之前，您必須執行`cdk synth`。

運行時，sam local invoke您需要要調用的函數構造標識符以及合成 AWS CloudFormation 模板的路徑。如果您的應用程式使用巢狀堆疊，為了解決命名衝突，您也需要定義函數的堆疊名稱。

用法：

```
# 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 AWS CDK application
sam local start-api -t ./cdk.out/CdkSamExampleStack.template.json [OPTIONS]

# Start a local endpoint that emulates AWS Lambda
sam local start-lambda -t ./cdk.out/CdkSamExampleStack.template.json [OPTIONS]
```

範例

請考慮使用下列範例宣告的堆疊和函數：

```
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', {
        ...
```

```
    });
}
```

下列命令會在本機叫用上述範例中定義的 Lambda 函數：

```
# 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
```

建立 AWS CDK 應用

可支 AWS SAMCLI援建置 AWS CDK 應用程式中定義的 Lambda 函數和層sam build。

對於使用 zip 加工品的 Lambda 函數，cdk synth請在執行sam local命令之前執行。 sam build不是必需的。

如果您的 AWS CDK 應用程式使用具有映像類型的函數，請sam build先執行cdk synth並執行，然後再執行sam local命令。執行時sam build， AWS SAM 不會建置使用執行階段特定建構的 Lambda 函數或層，例如。NodejsFunction sam build不支持捆綁的資產。

範例

從項目根 AWS CDK 目錄運行以下命令構建應用程序。

```
sam build -t ./cdk.out/CdkSamExampleStack.template.json
```

部署 AWS CDK 應用

AWS SAMCLI不支援部署 AWS CDK 應用程式。用cdk deploy於部署您的應用程式。如需詳細資訊，請參閱AWS Cloud Development Kit (AWS CDK) 開發人員指南中的AWS CDK 工具組 (cdk 指令)

使用發佈您的應用程式 AWS SAMCLI

若要讓其他人可以尋找和部署您的 AWS SAM 應用程式，您可以使用 AWS SAMCLI將應用程式發佈到 AWS Serverless Application Repository. 若要使用發行應用程式 AWS SAMCLI，您必須使用 AWS SAM 範本來定義應用程式。您還必須在本地或 AWS 雲中測試過它。

依照本主題中的指示建立新應用程式、建立現有應用程式的新版本，或更新現有應用程式的中繼資料。(您的作業取決於應用程式是否已存在於中 AWS Serverless Application Repository，以及是否有任何應用程式中繼資料正在變更。) 如需應用程式中繼資料的詳細資訊，請參閱 AWS SAM 樣板中繼資料區段性

必要條件

使用將應 AWS Serverless Application Repository 用程式發佈到之前 AWS SAMCLI，您必須具備下列項目：

- 已 AWS SAMCLI安裝。如需詳細資訊，請參閱 安裝 AWS SAMCLI。若要判斷 AWS SAMCLI是否已安裝，請執行下列命令：

```
sam --version
```

- 一個有效的 AWS SAM 模板。
- AWS SAM 範本參考的應用程式程式碼和相依性。
- 語義版本，只需要公開共享您的應用程序。此值可以像 1.0 一樣簡單。
- 指向應用程式原始程式碼的 URL。
- README.md 檔案。此檔案應說明客戶如何使用您的應用程式，以及如何在將應用程式部署到自己的 AWS 帳戶中之前進行設定。
- 一個LICENSE.txt文件，只需要公開共享您的應用程序。
- 如果您的應用程式包含任何巢狀應用程式，您必須已將它們發佈到 AWS Serverless Application Repository.
- 有效的 Amazon Simple Storage Service (Amazon S3) 儲存貯體政策，可授與封裝應用程式時上傳到 Amazon S3 的成品的服務讀取許可。若要設定此原則，請執行下列動作：

  1. 前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (Amazon S3) 主控台。
  2. 選擇您用來封裝應用程式的 Amazon S3 儲存貯體名稱。

3. 選擇許可。
4. 在 Permissions (許可) 索引標籤上，Bucket policy (儲存貯體政策) 下，選擇 Edit (編輯)。
5. 在 [編輯值區政策] 頁面上，將下列政策陳述式貼到 [原則編輯器] 中。在政策聲明中，請確保在元素中使用您的值區名稱，並在`Resource`元素中使用您的 AWS `Condition`帳戶 ID。`Condition`元素中的運算式可確保僅 AWS Serverless Application Repository 具有從指定 AWS 帳戶存取應用程式的權限。如需政策陳述式的詳細資訊，請參閱 IAM 使用者指南中的 IAM JSON 政策元素參考資料。

```
{
    "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. 選擇儲存變更。

發佈新的應用程式

步驟 1：將**`Metadata`**區段新增至 AWS SAM 範本

首先，在 AWS SAM 範本中新增`Metadata`區段。提供要發佈至的應用程式資訊 AWS Serverless Application Repository。

以下是範例`Metadata`區段：

```
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
        ...
```

如需 AWS SAM 範本Metadata區段的詳細資訊，請參閱AWS SAM 樣板中繼資料區段性。

步驟 2：打 Package 應用程序

執行下列 AWS SAMCLI命令，該命令會將應用程式的成品上傳到 Amazon S3，並輸出名為的新範本檔案packaged.yaml：

```
sam package --output-template-file packaged.yaml --s3-bucket <your-bucket-name>
```

您可以在下一個步驟中使用packaged.yaml範本檔案將應用程式發佈到 AWS Serverless Application Repository. 此檔案與原始範本檔案 (template.yaml) 類似，但有一個主要差異 — CodeUri LicenseUrl、和ReadmeUrl屬性指向 Amazon S3 儲存貯體和包含各自成品的物件。

packaged.yaml 範例範本檔案的下列程式碼片段會顯示 CodeUri 屬性：

```
MySampleFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: s3://bucketname/fbd77a3647a4f47a352fc0bjectGUID

...
```

步驟 3：發佈應用程式

若要將 AWS SAM 應用程式的私人版本發佈到 AWS Serverless Application Repository，請執行下列 AWS SAMCLI命令：

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

`sam publish`指令的輸出包括上應用程式的連結 AWS Serverless Application Repository。您也可以直接前往AWS Serverless Application Repository 登陸頁面並搜尋您的應用程式。

步驟 4：共享應用程序（可選）

根據預設，您的應用程式設定為私人，因此其他 AWS 帳戶無法看見該應用程式。若要與他人共用您的應用程式，您必須將應用程式設為公開，或授予特定 AWS 帳戶清單的權限。

如需有關使用共用應用程式的資訊 AWS CLI，請參閱AWS Serverless Application Repository 開發人員指南中的以資AWS Serverless Application Repository 源為基礎的政策範例。如需使用共用應用程式的相關資訊 AWS Management Console，請參閱AWS Serverless Application Repository 開發人員指南中的「共用應用程式」。

發佈現有應用程式的新版本

將應用程式發佈到之後 AWS Serverless Application Repository，您可能會想要發行該應用程式的新版本。例如，您可能已變更 Lambda 函數程式碼，或將新元件新增至應用程式架構。

若要更新先前已發佈的應用程式，請使用先前詳述的相同程序再次發佈應用程式。在 AWS SAM 範本檔案的`Metadata`區段中，提供您最初發佈該檔案時使用的相同應用程式名稱，但包含新`SemanticVersion`值。

例如，假設使用名稱`SampleApp`和 a `SemanticVersion` 的發行應用程式`1.0.0`。若要更新該應用程式， AWS SAM 範本必須具有應用程式名稱`SampleApp`和 a `SemanticVersion` `1.0.1` (或其他項目`1.0.0`)。

其他主題

- AWS SAM 樣板中繼資料區段性

AWS SAM 樣板中繼資料區段性

`AWS::ServerlessRepo::Application`是中繼資料索引鍵，可用來指定您要發佈至的應用程式資訊 AWS Serverless Application Repository。

> **Note**
>
> AWS CloudFormation `AWS::ServerlessRepo::Application`中繼資料金鑰不支援內建函式。

屬性

此表格提供樣 AWS SAM 板`Metadata`段落特性的相關資訊。若要將應用程式發行到 AWS Serverless Application Repository 使用，需要本節 AWS SAMCLI。

屬性	Type	必要	描述
`Name`	字串	TRUE	應用程式名稱。 最小長度 = 1。最大長度 = 140。 模式：`"[a-zA-Z0-9\\-]+";`
`Description`	字串	TRUE	應用程式的描述。 最小長度 = 1。最大長度 = 256。
`Author`	字串	TRUE	發佈應用程式的作者名稱。 最小長度 = 1。最大長度 = 127。 模式：`"^[a-z0-9](([a-z0-9]|-(?!-)` `)*[a-z0-9])?$";`
`SpdxLicenseId`	字串	FALSE	有效的授權識別碼。若要檢視有效的授權識別碼清單，請參閱軟體 Package 件 Data Exchange (SPDX) 網站上的 SPDX 授權清單。

屬性	Type	必要	描述
`LicenseUrl`	字串	FALSE	本機授權檔案或授權檔案的 Amazon S3 連結的參考，符合您應用程式的 SPDxLicenseID 值。 尚未使用`sam package`指令封裝的 AWS SAM 範本檔案可以具有此屬性的本機檔案參考。不過，若要使用`sam publish`命令發佈應用程式，此屬性必須是 Amazon S3 儲存貯體的參考。 最大尺寸：5 MB。 您必須為此屬性提供值，才能將您的應用程式公開。請注意，您無法在應用程式發佈之後更新此屬性。因此，若要將授權新增至應用程式，您必須先將其刪除，或使用不同名稱發佈新的應用程式。
`ReadmeUrl`	字串	FALSE	本機讀我檔案的參考或 Amazon S3 連結至 Readme 檔案，其中包含應用程式及其運作方式的更詳細說明。 尚未使用`sam package`指令封裝的 AWS SAM 範本檔案可以具有此屬性的本機檔案參考。不過，若要使用`sam publish`命令發佈，此屬性必須是 Amazon S3 儲存貯體的參考。 最大尺寸：5 MB。
`Labels`	字串	FALSE	改善搜尋結果中應用程式探索的標籤。 最小長度 = 1。最大長度 = 127。最大標籤數量：10。 模式：`"^[a-zA-Z0-9+\\-_:\\/@]+$";`
`HomePageUrl`	字串	FALSE	一個 URL，其中包含有關應用程式的詳細資訊，例如應用程式 GitHub 存放庫的位置。

屬性	Type	必要	描述
SemanticVersion	字串	FALSE	應用程式的語義版本。如需語意版本設定規格，請參閱語意版本控制網站。 您必須為此屬性提供值，才能將您的應用程式公開。
SourceCodeUrl	字串	FALSE	應用程式原始程式碼的公有儲存庫連結。

使用案例

本節列出發佈應用程式的使用案例，以及針對該使用案例處理的Metadata屬性。系統會忽略未針對指定使用案例列出的屬性。

- 建立新應用程式 — 如果中沒有 AWS Serverless Application Repository 與帳戶名稱相符的應用程式，則會建立新應用程式。
  - Name
  - SpdxLicenseId
  - LicenseUrl
  - Description
  - Author
  - ReadmeUrl
  - Labels
  - HomePageUrl
  - SourceCodeUrl
  - SemanticVersion
  - AWS SAM 範本的內容 (例如，任何事件來源、資源和 Lambda 函數程式碼)

- 建立應用程式版本 — 如果中已有 AWS Serverless Application Repository 與帳戶相符名稱的應用程式，且正在變更，則會建立應用程 SemanticVersion式版本。
  - Description
  - Author
  - ReadmeUrl

  - Labels
  - HomePageUrl
  - SourceCodeUrl
  - SemanticVersion
  - AWS SAM 範本的內容 (例如，任何事件來源、資源和 Lambda 函數程式碼)

- 更新應用程式 — 如果中已有 AWS Serverless Application Repository 與帳戶相符名稱的應用程式，且未變更應用程式， SemanticVersion則會更新應用程式。
  - Description
  - Author
  - ReadmeUrl
  - Labels
  - HomePageUrl

範例

以下是一個示例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
```

的文件歷史記錄 AWS SAM

下表說明《AWS Serverless Application Model 開發人員指南》每個版本中的重要變更。如需有關此文件更新的通知，您可以訂閱 RSS 摘要。

- 最新文件更新：2024 年 6 月 20 日

變更	描述	日期
整個開發人員指南中的重組和更新內容	重新整理並重新架構指南，以提高可發現性和可用性。更新和改進的標題。在介紹主題和概念時提供其他詳細資訊。	2024年6月20日
增加了 AWS SAMCLI對紅寶石 3.3 的支持	Ruby 3.3 現在可作為運行時和圖像存儲庫使用。有關詳細信息，請參見映像存儲庫和 sam init。	2024年4月4日
添加了 AWS SAMCLI命令選項	新的選項可用於命令山姆本地啟動 API:`--ssl-cert-file PATH`,. `--ssl-key-file PATH` 此外，新的命令行選項可用`--add-host LIST`於 sam 本地調用，sam 本地啟動 API 和山姆本地啟動-lambda	2024年3月20日
增加了 AWS SAMCLI對 .NET 8 的支持	.NET 8 現在可作為運行時和圖像存儲庫使用。不再支援 .NET 核心 3.1、Node.js 14、Node.js 12、Python 3.7、紅寶石 2.7 的執行時間和映像檔儲存庫。請參閱映像檔儲存庫和 sam 初始化。	2024年2月22 日

為添加了 AWS SAMCLI arm64 軟件包安裝程序 Linux	如需指示，請參閱安裝 AWS SAMCLI.	2023 年 12 月 6 日
為 sam 同步命令添加了-手錶排除選項 AWS SAMCLI	將檔案和資料夾排除在起始同步作業之外。若要深入了解，請參閱指定無法啟動同步的檔案和資料夾。	2023 年 12 月 6 日
為 AWS SAMCLI sam 同步命令添加了--build-in-source 選項	在源文件夾中構建項目以加快構建過程。要了解更多信息，請參閱通過在源文件夾中構建項目來加快構建時間。	2023 年 12 月 6 日
為 AWS SAMCLI山姆構建命令添加--build-in-source 選項	在源文件夾中構建項目以加快構建過程。要了解更多信息，請參閱通過在源文件夾中構建項目來加快構建時間。	2023 年 12 月 6 日
增加了對 AWS SAMCLI遠程調用命令的新資源支持	`sam remote invoke`搭配 Kinesis Data Streams 應用程式、Amazon SQS 佇列和 Step Functions 數狀態機器搭配使用。要了解更多信息，請參閱使用 sam 遠程調用。	2023 年 11 月 15 日
為可共享的測試事件添加了新的 AWS SAMCLI遠程測試事件命令	使用從 EventBridge 結構描述登錄存取和管理可共用的測試事件，以在中測試您的 Lambda 函數。 AWS SAM CLI AWS 雲端要了解更多信息，請參閱使用 sam 遠程測試事件。	2023 年 10 月 3 日
AWS SAMCLI支援現Terraform已正式推出	若要進一步了解 AWS SAMCLI 支援Terraform，請參閱AWS SAMCLITerraform支援。	2023 年 9 月 5 日

增加了 AWS SAMCLI支持 Terraform Cloud	AWS SAMCLI現在支援 Terraform Cloud. 若要深入瞭解，請參閱設定Terraform Cloud。	2023 年 9 月 5 日
增加了對 AWS SAMCLI配置YAML文件的文件格式支持	AWS SAMCLI現在支援 [.yaml|.yml] 檔案格式。配置 AWS SAMCLI和AWS SAMCLI配置文件頁面已更新。	2023 年 7 月 18 日
增加了對的 AWS SAMCLIsam local start-api命令支持 Terraform	什麼是 AWS SAMCLI支持 Terraform？ 區段已更新，以包含的 AWS SAMCLI`sam local start-api` 指令支援 Terraform。	2023 年 7 月 6 日
添加了新的 AWS SAMCLI遠程調用命令	若要開始使用`sam remote invoke`，請參閱使用 sam 遠端呼叫。	2023 年 6 月 22 日
新增 AWS AppSyncGraphQL API無伺服器資源類型	建立說`AWS::Serverless::GraphQLApi`明如何使用定義GraphQL API資源的新區段 AWS SAM。	2023 年 6 月 22 日
增加了對 Ruby 3.2 的 AWS SAMCLI支持	更新 sam init 頁面以包含新的基本圖像和運行時值。使用 Ruby 3.2 Amazon ECR URI 更新映像儲存庫頁面。	2023 年 6 月 6 日
添加了 AWS SAMCLI軟件包安裝程序的完整性驗證的可選步驟	更新安裝 AWS SAMCLI頁面以反映可選步驟。創建驗證 AWS SAMCLI安裝程序頁面的完整性以記錄步驟。	2023 年 5 月 31 日

添加了 sam 同步選項以跳過基礎架構同步	自訂每次執行時`sam sync`是否需要 AWS CloudFormation 部署。若要深入了解，請參閱略過初始 AWS CloudFormation 部署。	2023 年 3 月 23 日
增加了對文檔數據庫事件源類型的支持	範 AWS SAM 本規格現在支援`AWS::Serverless::Function` 資源的`DocumentDB` 事件來源類型。若要深入了解，請參閱 DocumentDB。	2023 年 3 月 10 日
使用建置 Lambda 鏽函數 Cargo Lambda	您 AWS SAMCLI可以使用來建置您的 Rust Lambda 函數Cargo Lambda。若要深入了解，請參閱使用Cargo Lambda.	2023 年 2 月 23 日
在以外構建函數資源 AWS SAM	已新增使用指令時略過功能的sam build指引。若要深入瞭解，請參閱以外的建置功能 AWS SAM。	2023 年 2 月 14 日
新的內嵌連接器語法	使用新的內嵌連接器語法來定義您的`AWS::Serverless::Connector` 資源。若要深入了解，請參閱使用 AWS SAM 連接器管理資源權限。	2023 年 2 月 8 日
添加了新的山姆列表命令 AWS SAMCLI	用sam list於檢視無伺服器應用程式中資源的重要資訊。要了解更多信息，請參閱 sam 列表。	2023 年 2 月 2 日

為 es OutExtension build 添加了格式和構建屬性	使用電子建置建置 Node.js Lambda 函式現在可支援Format並OutExtension 建置屬性。若要深入了解，請參閱使用電子建置來建置 Node.js Lambda 函數。	2023 年 2 月 2 日
在 AWS SAM 模板規範中添加了運行時管理選項	設定 Lambda 函數的執行階段管理選項。如需進一步了解，請參閱RuntimeManagementConfig 。	2023 年 1 月 24 日
EventSource 針對 AWS::Serverless::StateMachine資源新增目標屬性。	AWS::Serverless::StateMachine 資源類型支援的Target屬性EventBridgeRule 和Schedule事件來源。	2023 年 1 月 13 日
針對 Lambda 函數設定 SQS 輪詢器的調整	使用的ScalingConfig 內容設定 SQS 輪詢器的調整。AWS::Serverless::Function 如需進一步了解，請參閱 ScalingConfig 。	2023 年 1 月 12 日
使用 cfn- AWS SAM lint 驗證應用程式	您可以使用 cfn-lint 來驗證您的 AWS SAM 模板，通過. AWS SAMCLI 要了解更多信息，請參閱使用 cfn-lint 進行驗證。	2023 年 1 月 11 日
使用應用程式洞察來監控無伺服器 CloudWatch 應用	設定 Amazon 應用 CloudWatch 程式洞察以監控您的 AWS SAM 應用程式。若要深入了解，請參閱使 CloudWatch 用應用程式深入解析來監控無伺服器應用	2022 年 12 月 19 日

為 macOS 添加了 AWS SAMCLI軟件包安裝	AWS SAMCLI使用新的 macOS 軟件包安裝程序進行安裝。若要深入瞭解，請參閱安裝 AWS SAMCLI.	2022 年 12 月 6 日
增加了對 Lambda 支持 SnapStart	設 SnapStart 定 Lambda 函數以建立快照，快照是初始化函數的快取狀態。如需進一步了解，請參閱`AWS::Serverless::Function`。	2022 年 11 月 28 日
增加了對節點 18.x 的 AWS SAMCLI支持	AWS SAMCLI現在支援 nodejs18.x 執行階段。要了解更多信息，請參閱 sam 初始化。	2022 年 11 月 17 日
已新增設定存取權和權限的指引	AWS SAM 提供兩個選項，可簡化無伺服器應用程式的存取和權限管理。若要深入瞭解，請參閱管理資源存取和權限。	2022 年 11 月 17 日
增加了對使用本機 AOT 編譯構建 .NET 7 Lambda 函數的支持	使用原生提前時間 (AOT) 編譯來建置和封裝 .NET 7 Lambda 函數 AWS SAM，以改善 Lambda 冷啟動時間。若要深入了解，請參閱使用原生 AOT 編譯建置 .NET 7 Lambda 函數。	2022 年 11 月 15 日
增加了 AWS SAMCLITerraform對本地調試和測試的支持	使用Terraform專案 AWS SAMCLI中的，對 Lambda 函數和層執行本機偵錯和測試。若要深入了解，請參閱 AWS SAM CLI Terraform 支援。	2022 年 11 月 14 日

增加了對 EventBridge 調度的 AWS SAM 支持	AWS Serverless Application Model (AWS SAM) 範本規格提供簡單、簡短的語法，您可以使用這些語法來 EventBridge 排程與的 Scheduler 事件。 AWS Lambda AWS Step Functions如需詳細資訊，請參閱使用排程 EventBridge 器排程事件。	2022 年 11 月 10 日
簡化了 AWS SAMCLI安裝說明	AWS SAMCLI必要條件和選擇性步驟已移至不同的頁面。您可以在「安裝」中找到支援作業系統的安裝步驟 AWS SAMCLI。	2022 年 11 月 4 日
增加了修復程序，以允許 Windows 10 用戶的長路徑	AWS SAMCLI應用程序模板存儲庫包含一些長文件路徑，`sam init`由於 Windows 10 的MAX_PATH限制，運行時可能會導致錯誤。如需詳細資訊，請參閱安裝 AWS SAMCLI	2022 年 11 月 4 日
針對首次部署更新逐步部署程序	逐步部署 Lambda 函數， AWS CodeDeploy 需要兩個步驟。若要深入了解，請參閱第一次逐步部署 Lambda 函數。	2022 年 10 月 13 日
更多事件類型的額外 Lambda 事件篩選支援	`FilterCritera` 屬性已加入至`MSKMQ`、和`SelfManagedKafka` 事件來源類型。	2022 年 10 月 13 日

增加了對管道的 OpenID Connect（OIDC）支持 AWS SAM	AWS SAM 支持 OpenID Connect（OIDC）用戶身份驗證，用於比特桶， GitHub操作以及持 GitLab 續集成和持續交付（CI/CD）平台。若要深入瞭解，請參閱搭 AWS SAM 配管線使用 OIDC 使用者帳戶。	2022 年 10 月 13 日
注意 JwtConfiguration 屬性	已在「」下新增有關定義issuer和audience屬性JwtConfiguration 的附註OAuth2Authorizer 。	2022 年 10 月 7 日
功能和的新屬性 StateMachine EventSource	Enabled和State屬性已新增至的CloudWatchEvent 事件來源AWS::Serverless::Function 。 State屬性已加入至和的Schedule事件AWS::Serverless::Function 來源AWS::Serverless::StateMachine 。	2022 年 10 月 6 日
AWS SAM 連接器現已正式推出	連接器是一種 AWS SAM 抽象資源類型，識別為AWS::Serverless::Connector ，可在無伺服器應用程式資源之間提供一種簡單且安全的佈建權限方法。若要深入了解，請參閱使用 AWS Serverless Application Model 連接器管理資源權限。	2022 年 10 月 6 日
增加了新的山姆同步選項 AWS SAMCLI	--dependency-layer 和--use-container 選項已新增至 sam sync.	2022 年 9 月 20 日

增加了新的 sam 部署選項 AWS SAMCLI	`--on-failure` 選項已加入至sam deploy。	2022 年 9 月 9 日
ebuild 支持現已正式推出	若要建置和封裝 Node.js Lambda 函數，您可以使用 AWS SAMCLI與電子建置 JavaScript 捆綁程式搭配使用。	2022 年 9 月 1 日
更新的 AWS SAMCLI遙測	已更新收集的系統與環境資訊說明，以包含使用狀況屬性的雜湊值。	2022 年 9 月 1 日
增加了局部環境變量支持 AWS SAMCLI	在本機叫用 Lambda 函數 AWS SAMCLI時以及在本機執行 API Gateway 時使用環境變數。	2022 年 9 月 1 日
Support Lambda 指令集架構	使用建 AWS SAMCLI置 Lambda 函數和 Lambda 層`x86_64`或`arm64`指令集架構。如需詳細資訊，請參閱`AWS::Serverless::Function` 資源類型的架構`CompatibleArchitectures`屬性和`AWS::Serverless::LayerVersion` 資源類型的屬性。	2021 年 10 月 1 日
產生範例配管組態	使用可 AWS SAMCLI使用 new `sam pipeline bootstrap` 和`sam pipeline init`指令為多個 CI/CD 系統產生範例管線。如需詳細資訊，請參閱 < 產生範例 CI/CD 管線 > 。	2021 年 7 月 21 日

AWS SAMCLIAWS CDK 整合 (預覽，階段 2)	透過公開預覽版的第 2 階段，您現在可以使用封裝和部署 AWS CDK 應用程式。 AWS SAMCLI您也可以使用直接下載範例 AWS CDK 應用程式 AWS SAMCLI。若要取得更多資訊，請參閱 AWS Cloud Development Kit (AWS CDK) (預覽)。	2021 年 7 月 13 日
Support RabbitMQ 作為函數的事件來源	已新增對 RabbitMQ 做為無伺服器功能事件來源的支援。如需詳細資訊，請參閱資源類型之MQ事件來AWS::Serverless::Function 源的SourceAccessConfigurations 屬性。	2021 年 7 月 7 日
使用 Amazon ECR 部署無伺服器應用程式建置容器映像	使用 Amazon ECR 建置容器映像 AWS CodePipeline，透過 Jenkins、CI/CD 和動作等常見 CI/CD 系統部署無伺服器應用程式。 GitLab GitHub 如需詳細資訊，請參閱部署無伺服器應用程式。	2021 年 6 月 24 日
使用 AWS 工具組偵錯 AWS SAM 應用程式	AWS Toolkit 現在透過更多整合式開發環境 (IDE) 和執行階段組合來支援逐步偵錯。如需詳細資訊，請參閱使用 AWS 工具組。	2021 年 5 月 20 日

AWS SAMCLIAWS CDK 整合 (預覽)	您現在可以使用在本 AWS SAMCLI機測試和建置 AWS CDK 應用程式。這是公開預覽版本。若要取得更多資訊，請參閱 AWS Cloud Development Kit (AWS CDK) (預覽)。	2021 年 4 月 29 日
預設容器映像儲存庫已變更為 Amazon ECR 公開	預設容器映像檔儲存庫從變更 DockerHub 為 Amazon ECR 公用版。如需詳細資訊，請參閱映像儲存庫。	2021 年 4 月 6 日
每晚構 AWS SAMCLI建	您現在可以安裝每晚建置的 AWS SAMCLI預發行版本。如需詳細資訊，請參閱安裝. AWS SAMCLI	2021 年 3 月 25 日
建置容器環境變數支援	您現在可以傳遞環境變數來建置容器。如需詳細資訊，請參閱中的`--container-env-var` 和`--container-env-var-file` 選項`sam build`。	2021 年 3 月 4 日
新的安裝程序	您現在可以 AWS SAMCLI使用原生 Linux 安裝程式來安裝。如需詳細資訊，請參閱AWS SAMCLI在 Linux 上安裝。	2021 年 2 月 10 日

Support 無效字母佇列 EventBridge	新增對無伺服器函數和狀態機器的無效字母佇列 EventBridge 和Schedule事件來源的支援。如需詳細資訊，請參閱EventBridgeRule 和Schedule事件來源的DeadLetterConfig 內容，以瞭解AWS::Serverless::Function 和AWS::Serverless::StateMachine 資源類型。	2021 年 1 月 29 日
Support 自訂檢查點	針對無伺服器函數新增 DynamoDB 和 Kinesis 事件來源的自訂檢查點的支援。如需詳細資訊，請參閱資源類型的FunctionResponseTypes 屬性Kinesis和AWS::Serverless::Function 資DynamoDB料類型。	2021 年 1 月 29 日
Support 翻滾窗戶	已針對無伺服器功能新增 DynamoDB 和 Kinesis 事件來源重置視窗的支援。如需詳細資訊，請參閱資源類型的TumblingWindowInSeconds 屬性Kinesis和AWS::Serverless::Function 資DynamoDB料類型。	2020 年 12 月 17 日

Support 溫暖的容器	在使用 AWS SAMCLI命令`sam local start-api` 和進行本地測試時添加了對溫容器的支持`sam local start-lambda` 。如需詳細資訊，請參閱這些指令的`--warm-containers` 選項。	2020 年 12 月 16 日
Support Lambda 容器映像	增加了對 Lambda 容器映像的支持。如需詳細資訊，請參閱建置應用程式。	2020 年 12 月 1 日
Support 程式碼簽章	已新增程式碼簽章和受信任的無伺服器應用程式程式碼部署支援。如需詳細資訊，請參閱設定 AWS SAM 應用程式的程式碼簽章。	2020 年 11 月 23 日
Support parallel 和緩存構建	透過在`sam build`命令中新增兩個選項`--parallel` ，以改善無伺服器應用程式組建的效能：這些選項會以 parallel 而非循序方式建置函數和層`--cached`，並且在未進行任何需要重建的變更時，會使用先前組建的建置加工品。	2020 年 11 月 10 日

Support Amazon MQ 和相互 TLS 身份驗證	已新增對 Amazon MQ 作為無伺服器功能事件來源的支援。如需詳細資訊，請參閱資源類型的EventSource 和AWS::Serverless::Function 資MQ料類型。此外，也新增了 API Gateway API 和 HTTP API 的相互傳輸層安全性 (TLS) 驗證的支援。如需詳細資訊，請參閱AWS::Serverless::Api 資源類型的資料類型或HttpApiDomainConfiguration 資AWS::Serverless::HttpApi 源類型的資料類型。DomainConfiguration	2020 年 11 月 5 日
Support Lambda API 的授權者	已新增對AWS::Serverless::HttpApi 資源類型的 Lambda 授權者的支援。如需詳細資訊，請參閱 Lambda 授權者範例 (AWS::Serverless::HttpApi)。	2020 年 10 月 27 日
Support 多個配置文件和環境	增加了對多個配置文件和環境的支持，以存儲 AWS SAMCLI 命令的默認參數值。如需詳細資訊，請參閱AWS SAMCLI組態檔案。	2020 年 9 月 24 日

Support 使用 Step Functions 的 X-Ray，以及控制 API 存取時的參考	新增對 X-Ray 做為無伺服器狀態機事件來源的支援。如需詳細資訊，請參閱AWS::Serverless::StateMachine 資源類型的Tracing內容。在控制 API 存取時，也新增了對參考的支援。如需詳細資訊，請參閱資ResourcePolicyStatement 料類型。	2020 年 9 月 17 日
Support Amazon MSK	已新增 Amazon MSK 作為無伺服器功能事件來源的支援。這可讓 Amazon MSK 主題中的記錄觸發您的 Lambda 函數。如需詳細資訊，請參閱資源類型的EventSource 和AWS::Serverless::Function 資MSK料類型。	2020 年 8 月 13 日
Support Amazon EFS	已新增將 Amazon EFS 檔案系統掛載到本機目錄的支援。這可讓您的 Lambda 函數程式碼存取和修改共用資源。如需詳細資訊，請參閱AWS::Serverless::Function 資源類型的FileSystemConfigs 內容。	2020 年 6 月 16 日
協調無伺服器應用程式	已新增透過使用建立 Step Functions 狀態機器來協調應用 AWS SAM程式的支援。如需詳細資訊，請參閱使用 AWS Step Functions和 AWS 資源類型協調AWS::Serverless::StateMachine 資源。	2020 年 5 月 27 日

建立自訂執行階段	新增建立自訂執行階段的功能。如需詳細資訊，請參閱建立自訂執行階段。	2020 年 5 月 21 日
建築層	新增建立個別LayerVersion 資源的功能。若要取得更多資訊，請參閱〈建置層〉	2020 年 5 月 19 日
產生的 AWS CloudFormation 資源	提供有關 AWS SAM 產生的 AWS CloudFormation 資源以及如何參考這些資源的詳細資訊。如需詳細資訊，請參閱產生的 AWS CloudFormation 資源。	2020 年 4 月 8 日
設定 AWS 認證	已新增設定 AWS 認證的指示，以防您尚未將憑證設定為與其他 AWS 工具搭配使用，例如其中一個 AWS SDK 或. AWS CLI如需詳細資訊，請參閱設定 AWS 認證。	2020 年 1 月 17 日
AWS SAM 規格和 AWS SAMCLI更新	從中移轉 AWS SAM 規格 GitHub。如需詳細資訊，請參閱AWS SAM 規格。還使用對sam deploy指令的變更更新了部署工作流程。	2019 年 11 月 25 日

控制 API Gateway API 和政策範本更新存取的新選項	新增控制 API Gateway API 存取的新選項：IAM 許可、API 金鑰和資源政策。如需詳細資訊，請參閱控制 API Gateway API 的存取。還更新了兩個策略模板：RekognitionFacesPolicy 和 ElasticsearchHttpPostPolicy。如需詳細資訊，請參閱AWS SAM 策略範本。	2019 年 8 月 29 日
開始使用更新	更新了入門章節與改進的安裝說明 AWS SAMCLI和 Hello World 教程。如需詳細資訊，請參閱開始使用 AWS SAM。	2019 年 7 月 25 日
控制對 API Gateway API 的存取	新增了對控制 API Gateway API 存取的支援。如需詳細資訊，請參閱控制 API Gateway API 的存取。	2019 年 3 月 21 日
已新`sam publish`增至 AWS SAMCLI	中的新`sam publish`指令 AWS SAMCLI簡化了在中發佈無伺服器應用程式的程序 AWS Serverless Application Repository。如需詳細資訊，請參閱使用發佈無伺服器應用程式。 AWS SAMCLI	2018 年 12 月 21 日
嵌套應用程序和圖層支持	增加了對嵌套應用程序和圖層的支持。如需詳細資訊，請參閱使用巢狀應用程式和處理圖層。	2018 年 11 月 29 日

已新sam build增至 AWS SAMCLI	中的新sam build指令 AWS SAMCLI簡化了編譯具有相依性的無伺服器應用程式的程序，以便您可以在本機測試和部署這些應用程式。如需詳細資訊，請參閱建置應用程式。	2018 年 11 月 19 日
增加了新的安裝選項 AWS SAMCLI	添加了 Linux 軟件（Linux），微星（視窗）和自製軟件（macOS）的安裝選項。AWS SAMCLI如需詳細資訊，請參閱安裝 AWS SAMCLI.	2018 年 11 月 7 日
新的指南	這是《AWS Serverless Application Model 開發人員指南》的第一版。	2018 年 10 月 17 日

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。