



開發人員指南

AWS Serverless Application Model



AWS Serverless Application Model: 開發人員指南

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商標和商業外觀不得用於任何非 Amazon 的產品或服務，也不能以任何可能造成客戶混淆、任何貶低或使 Amazon 名譽受損的方式使用 Amazon 的商標和商業外觀。所有其他非 Amazon 擁有的商標均為其各自擁有者的財產，這些擁有者可能附屬於 Amazon，或與 Amazon 有合作關係，亦或受到 Amazon 贊助。

Table of Contents

什麼是 AWS SAM ?	1
使用時機 AWS SAM	1
主要功能	2
相關資訊	2
運作方式	2
範本 AWS SAM 規格是什麼 ?	3
什麼是 AWS SAM 專案和 AWS SAM 範本 ?	3
什麼是 AWS SAM CLI ?	10
進一步了解	18
後續步驟	18
無伺服器概念	18
無伺服器概念	19
基礎設施即程式碼	19
無 AWS 伺服器應用程式的 IaC 工具	20
比較無伺服器應用程式的 IaC 工具	20
進一步了解	21
開始使用	22
先決條件	22
註冊 AWS 帳戶	23
步驟 1：安裝 AWS CLI	23
步驟 2：使用 AWS CLI 設定 AWS 登入資料	23
步驟 3：(選用) 安裝 AWS Toolkit for VS 程式碼	24
後續步驟	25
安裝 AWS SAM CLI	25
安裝 AWS SAM CLI	26
對安裝錯誤進行故障診斷	35
後續步驟	36
選用：驗證 AWS SAM CLI 安裝程式	36
Hello World 教學課程	48
先決條件	49
步驟 1：初始化範例 Hello World 應用程式	50
步驟 2：建置您的應用程式	54
步驟 3：將您的應用程式部署到 AWS 雲端	55
步驟 4：執行您的應用程式	60

步驟 5：與 中的 函數互動 AWS 雲端	61
步驟 6：修改您的應用程式並將其同步至 AWS 雲端	62
步驟 7：（選用）在本機測試您的應用程式	65
步驟 8：從 刪除您的應用程式 AWS 雲端	67
疑難排解	68
進一步了解	68
從 Lambda 轉換	68
進一步了解	69
如何使用 AWS SAM	70
AWS SAM CLI	71
如何 AWS SAM CLI 記錄命令	71
設定 AWS SAM CLI	72
如何建立您的組態檔案（samconfig 檔案）	73
設定專案設定	74
設定登入資料和基本設定	78
核心命令	78
使用 進行本機測試 AWS SAM CLI	79
在本機測試 AWS SAM 應用程式	80
在本機測試 CloudFormation 範本	80
在本機測試 AWS CDK 應用程式	80
在本機測試 Terraform 應用程式	80
AWS SAM 範本	81
範本剖析	82
YAML	82
範本區段	83
後續步驟	85
全域變數	85
資源和屬性	91
AWS::Serverless::Api	92
AWS::Serverless::Application	147
AWS::Serverless::CapacityProvider	152
AWS::Serverless::Connector	162
AWS::Serverless::Function	176
AWS::Serverless::GraphQLApi	329
AWS::Serverless::HttpApi	363
AWS::Serverless::WebSocketApi	392

AWS::Serverless::LayerVersion	410
AWS::Serverless::MicrovmImage	417
AWS::Serverless::NetworkConnector	424
AWS::Serverless::SimpleTable	430
AWS::Serverless::StateMachine	435
AWS::Serverless::WebFunction	473
產生的資源	490
參考產生的 CloudFormation 資源	490
產生的 CloudFormation 資源案例	491
AWS::Serverless::Api	495
AWS::Serverless::Application	496
AWS::Serverless::CapacityProvider	497
AWS::Serverless::Connector	497
AWS::Serverless::Function	498
AWS::Serverless::GraphQLApi	504
AWS::Serverless::HttpApi	505
AWS::Serverless::MicrovmImage	507
AWS::Serverless::NetworkConnector	507
AWS::Serverless::WebFunction	508
AWS::Serverless::WebSocketApi	509
AWS::Serverless::LayerVersion	512
AWS::Serverless::SimpleTable	512
AWS::Serverless::StateMachine	512
支援的資源屬性	514
例外狀況	515
API Gateway 擴充功能	516
內建函數	517
語言擴充功能支援	518
啟用語言延伸模組	518
Fn::ForEach	520
支援的內部函數	527
驗證錯誤	527
限制	528
遙測	529
範例應用程式	530
處理 DynamoDB 事件	530

開始之前	530
步驟 1：初始化應用程式	530
步驟 2：在本機測試應用程式	531
步驟 3：封裝應用程式	531
步驟 4：部署應用程式	532
後續步驟	532
處理 Amazon S3 事件	533
開始之前	533
步驟 1：初始化應用程式	533
步驟 2：封裝應用程式	534
步驟 3：部署應用程式	534
步驟 4：在本機測試應用程式	535
後續步驟	535
開發您的應用程式	536
建立您的應用程式	536
初始化新的無伺服器應用程式	537
sam init 的選項	542
疑難排解	543
範例	543
進一步了解	543
後續步驟	544
定義您的基礎設施	544
定義應用程式資源	544
設定存取權	546
控制 API 存取	621
使用 layer 提高效率	634
重複使用程式碼	636
管理以時間為基礎的事件	639
協調應用程式	642
設定程式碼簽署	644
驗證 AWS SAM 範本檔案	646
建置您的應用程式	647
簡介 sam build	647
預設建置	661
自訂您的建置	669
測試您的應用程式	700

簡介 sam local	700
使用 sam local 命令	701
簡介 sam local generate-event	701
簡介 sam local invoke	707
簡介 sam local start-api	713
簡介 sam local start-lambda	718
本機叫用 函數	721
環境變數檔案	722
層	724
進一步了解	724
在本機執行 API Gateway	724
環境變數檔案	725
層	727
使用 測試 sam remote test-event	727
設定 AWS SAM CLI 以使用 sam remote test-event	728
使用 sam remote test-event 命令	728
使用可共用的測試事件	731
管理可共用的測試事件	731
使用 測試 sam remote invoke	732
使用 sam 遠端叫用命令	733
使用 sam 遠端叫用命令選項	737
設定您的專案組態檔案	742
範例	742
相關連結	757
自動化整合測試	758
產生範例承載	759
測試和偵錯耐用函數	760
本機測試工作流程	760
常見測試案例	760
疑難排解	761
偵錯秘訣	762
進一步了解	762
偵錯您的應用程式	763
本機偵錯函數	763
使用 AWS 工具組	763
在偵錯模式下於 AWS SAM 本機執行	765

傳遞多個執行時間引數	766
使用 cfn-lint 驗證	766
範例	767
進一步了解	767
部署您的應用程式和資源	768
簡介 sam deploy	768
先決條件	769
使用 sam 部署來部署應用程式	769
最佳實務	779
sam 部署的選項	779
疑難排解	779
範例	779
進一步了解	788
部署選項	788
如何使用 AWS SAM CLI 手動部署	788
使用 CI/CD 系統和管道部署	789
逐步部署	789
使用 對部署進行故障診斷 AWS SAM CLI	789
進一步了解	724
使用 CI/CD 系統和管道部署	790
什麼是管道？	790
如何 AWS SAM 上傳本機檔案	791
產生入門管道	799
自訂入門管道	804
自動化您的部署	806
使用 OIDC 身分驗證	810
簡介 sam sync	812
自動偵測本機變更並將其同步至 AWS 雲端	813
自訂要同步到 的本機變更 AWS 雲端	814
在雲端中準備您的應用程式以進行測試和驗證	814
sam 同步命令的選項	814
疑難排解	817
範例	817
進一步了解	824
監控您的應用程式	825
Application Insights	825

使用 設定 CloudWatch Application Insights AWS SAM	825
後續步驟	829
使用日誌	829
依 CloudFormation 堆疊擷取日誌	829
依 Lambda 事件函數名稱擷取日誌	829
自訂日誌	829
檢視特定時間範圍的日誌	830
篩選日誌	830
反白顯示錯誤	830
JSON 美型列印	830
AWS SAM 參考	831
AWS SAM 規格和 AWS SAM 範本	831
AWS SAM CLI 命令參考	831
AWS SAM 政策範本	832
主題	832
AWS SAM CLI 命令	832
sam build	833
sam delete	838
sam deploy	840
sam init	846
sam list	850
sam local callback	857
sam local execution	863
sam local generate-event	868
sam local invoke	870
sam local start-api	876
sam local start-lambda	881
sam logs	886
sam package	889
sam pipeline bootstrap	892
sam pipeline init	896
sam publish	897
sam remote callback	899
sam remote execution	905
sam remote invoke	911
sam remote test-event	916

sam sync	922
sam traces	928
sam validate	930
AWS SAM CLI 管理	931
AWS SAM CLI 組態檔案	932
管理 AWS SAM CLI 版本	938
設定 AWS 登入資料	947
AWS SAM CLI 遙測	948
疑難排解	951
連接器參考	956
支援的連接器資源類型	956
連接器建立的 IAM 政策	966
安裝 Docker	989
安裝 Docker	990
後續步驟	992
安裝 Finch	993
安裝 Finch	993
Finch 使用 CLI AWS SAM 設定	995
驗證安裝	996
停用 Finch	997
疑難排解	998
後續步驟	998
映像儲存庫	999
映像儲存庫 URIs	999
範例	1001
逐步部署	1002
第一次逐漸部署 Lambda 事件函數	1004
進一步了解	1005
重要說明	1005
2023	1006
Terraform 支援	1007
開始使用	1007
先決條件	1007
搭配 使用 AWS SAM CLI 命令 Terraform	1008
設定 Terraform 專案	1008
設定 Terraform Cloud	1013

搭配 使用 AWS SAM CLI Terraform	1015
使用 進行本機測試 sam local invoke	1015
使用 進行本機測試 sam local start-api	1015
使用 進行本機測試 sam local start-lambda	1017
Terraform 限制	1018
搭配 Serverless.tf 使用 AWS SAM CLI	1018
Terraform 參考	1018
AWS SAM 支援的功能參考	1019
Terraform 特定參考	1019
sam 中繼資料	1019
AWS SAM CLI Terraform 支援	1022
什麼是 AWS SAM CLI ?	1022
如何搭配 AWS SAM CLI使用 Terraform ?	1023
後續步驟	1023
發佈供其他人使用	1024
先決條件	1024
發佈新的應用程式	1025
步驟 1：將Metadata區段新增至 AWS SAM 範本	1025
步驟 2：封裝應用程式	1026
步驟 3：發佈應用程式	1027
步驟 4：共用應用程式（選用）	1027
發佈現有應用程式的新版本	1027
其他主題	1027
中繼資料區段屬性	1028
Properties	1028
使用案例	1030
範例	1031
文件歷史紀錄	1032
.....	mlx

什麼是 AWS Serverless Application Model (AWS SAM) ?

AWS Serverless Application Model (AWS SAM) 是一種開放原始碼架構，可使用基礎設施即程式碼 (IaC) 建置無伺服器應用程式。使用 AWS SAM 的速記語法，開發人員會宣告 [CloudFormation](#) 資源和專用無伺服器資源，這些資源會在部署期間轉換為基礎設施。使用時 AWS SAM，您將與 互動：

1. AWS SAM CLI - 命令列工具，可協助您開發、本機測試和部署無伺服器應用程式。
2. AWS SAM 範本 - 的延伸 CloudFormation，提供定義無伺服器資源的簡化語法。

當您使用 `sam init` 命令時，它會建立專案目錄，我們將此目錄稱為 AWS SAM 專案，通常包含您的 AWS SAM 範本、應用程式碼和其他組態檔案。

使用時機 AWS SAM

AWS SAM 是理想的 IaC 解決方案，適用於您希望使用的完整功能進行簡化無伺服器開發的案例 CloudFormation。例如，您可以將 SAM 用於：

- 無伺服器應用程式：您可以使用 SAM 快速定義 AWS Lambda 事件函數（先前稱為 Lambda 函數）、Lambda 持久函數、Amazon API Gateway APIs、Amazon DynamoDB 資料表，以及其他程式碼最少的無伺服器資源。
- CloudFormation 增強功能：您可以將 SAM 與現有的 CloudFormation 範本結合，將無伺服器元件新增至傳統基礎設施。SAM 資源可與相同範本中的標準 CloudFormation 資源搭配使用。
- 本機開發和測試：您可以使用 SAM 在本機 CLI 測試 Lambda 函數、模擬 API Gateway 端點，以及在部署至之前在開發機器上偵錯無伺服器應用程式 AWS。
- 無伺服器 CI/CD：您可以使用 SAM 範本建置部署管道，自動產生預備和生產環境所需的 CloudFormation 基礎設施。
- 從主控台建立的資源遷移：您可以使用 SAM 範本，將 AWS Lambda 函數和在管理主控台中建立的 API Gateway 資源轉換為基礎設施做為程式碼。

AWS SAM 與其他 IaC 工具比較

- 使用 SAM 而非 CloudFormation 來簡化無伺服器資源定義，同時維持範本相容性。
- AWS CDK 如果您偏好使用宣告式方法來描述基礎設施，而不是使用程式設計方法，請使用 SAM。
- AWS CDK 使用 SAM CLI 的本機測試功能來增強 CDK 應用程式，以將 SAM 與 結合。

主要功能

AWS SAM 提供各種優點，可讓您：

使用較少的程式碼，快速定義您的應用程式基礎設施程式碼

編寫 AWS SAM 範本以定義您的無伺服器應用程式基礎設施程式碼。將您的範本直接部署到 CloudFormation 以佈建您的 資源。

在整個開發生命週期中管理您的無伺服器應用程式

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

使用 AWS SAM 連接器在資源之間快速佈建許可

在 AWS SAM 範本中使用 AWS SAM 連接器來定義 AWS 資源之間的許可。會將您的程式碼 AWS SAM 轉換為促進意圖所需的 IAM 許可。如需詳細資訊，請參閱[使用 AWS SAM 連接器管理資源許可](#)。

在您開發時持續同步本機變更至雲端

使用 AWS SAM CLIsam sync命令自動同步本機變更至雲端，加速開發和雲端測試工作流程。如需詳細資訊，請參閱[使用 sam sync 同步至 的簡介 AWS 雲端](#)。

管理您的無Terraform伺服器應用程式

使用 AWS SAM CLI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 SAM CLI Terraform 支援](#)。

相關資訊

- 如需 AWS SAM 運作方式的詳細資訊，請參閱 [AWS SAM 運作方式](#)。
- 若要開始使用 AWS SAM，請參閱 [入門 AWS SAM](#)。
- 如需如何使用 AWS SAM 建立無伺服器應用程式的概觀，請參閱 [如何使用 AWS SAM](#)。

AWS SAM 運作方式

使用 AWS SAM 建立無伺服器應用程式時，您將與下列元件互動：

1. [AWS SAM 範本](#) – 定義 AWS 資源的重要檔案。此範本包含 AWS SAM 範本規格 – 開放原始碼架構，隨附您用來定義無伺服器應用程式的函數、事件、APIs、組態和許可的簡化速記語法。此檔案位於 AWS SAM 專案中，這是在您執行 `sam init` 命令時建立的應用程式資料夾。
2. [AWS SAM CLI](#) – 命令列工具，可與 AWS SAM 專案和支援的第三方整合搭配使用，以建置和執行無伺服器應用程式。AWS SAM CLI 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。

若要表達定義無伺服器應用程式的資源、事件來源映射和其他屬性，您可以在 AWS SAM 範本和 AWS SAM 專案中的其他檔案中定義資源和開發應用程式。您可以使用在 AWS SAM CLI AWS SAM 專案上執行命令，這是初始化、建置、測試和部署無伺服器應用程式的方式。

第一次使用無伺服器？

我們建議您檢閱 [的無伺服器概念 AWS Serverless Application Model](#)。

範本 AWS SAM 規格是什麼？

AWS SAM 範本規格是一種開放原始碼架構，可用來定義和管理無伺服器應用程式基礎設施程式碼。AWS SAM 範本規格為：

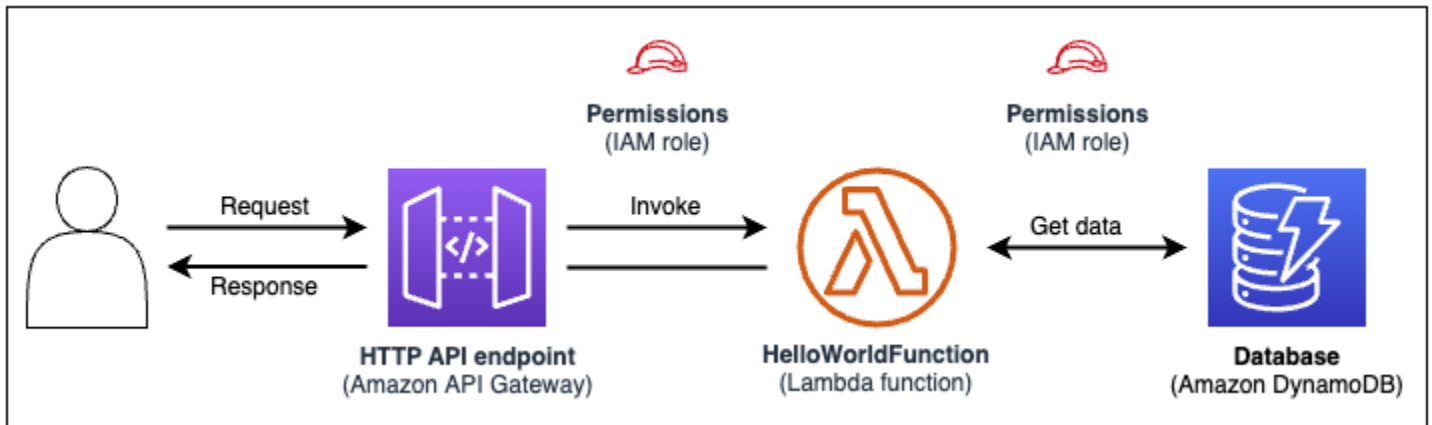
- 建置於 AWS CloudFormation - 您可以直接在 AWS SAM 範本中使用 CloudFormation 語法，充分利用其對資源和屬性組態的廣泛支援。如果您已經熟悉 CloudFormation，就不必學習新的服務來管理您的應用程式基礎設施程式碼。
- 的延伸 CloudFormation AWS SAM 提供自己的唯一語法，特別著重於加速無伺服器開發。您可以在相同的範本中使用 AWS SAM CloudFormation 和 語法。
- 抽象的速記語法 – 使用 AWS SAM 語法，您可以快速定義基礎設施、減少程式碼行，並降低發生錯誤的機率。其語法經過特別策劃，可消除定義無伺服器應用程式基礎設施的複雜性。
- 轉換 – AWS SAM 執行將範本轉換為透過 佈建基礎設施所需的程式碼的複雜工作 CloudFormation。

什麼是 AWS SAM 專案和 AWS SAM 範本？

AWS SAM 專案包含範本 AWS SAM，其中包含 AWS SAM 範本規格。此規格是您用來定義無伺服器應用程式基礎設施的開放原始碼架構 AWS，具有一些額外的元件，可讓您更輕鬆地使用。在此意義上，AWS SAM 範本是 CloudFormation 範本的延伸。

以下是基本無伺服器應用程式的範例。此應用程式會處理透過 HTTP 請求從資料庫取得所有項目的請求。它包含下列部分：

1. 包含處理請求之邏輯的函數。
2. 做為用戶端（請求者）與應用程式之間通訊的 HTTP API。
3. 存放項目的資料庫。
4. 應用程式安全執行的許可。



您可以在下列 AWS SAM 範本中定義此應用程式的基礎設施程式碼：

```

AWSTemplateFormatVersion: 2010-09-09
Transform: AWS::Serverless-2016-10-31
Resources:
  getAllItemsFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: src/get-all-items.getAllItemsHandler
      Runtime: nodejs20.x
      Events:
        Api:
          Type: HttpApi
          Properties:
            Path: /
            Method: GET
      Connectors:
        MyConn:
          Properties:
            Destination:
              Id: SampleTable
  
```

```
Permissions:
  - Read
SampleTable:
  Type: AWS::Serverless::SimpleTable
```

在 23 行程式碼中，定義了下列基礎設施：

- 使用 AWS Lambda 服務的 函數。
- 使用 Amazon API Gateway 服務的 HTTP API。
- 使用 Amazon DynamoDB 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。

若要佈建此基礎設施，會部署 範本 CloudFormation。在部署期間，會將 23 行程式碼 AWS SAM 轉換為產生這些資源所需的 CloudFormation 語法 AWS。轉換後的 CloudFormation 範本包含超過 200 行程式碼！

轉換的 CloudFormation 範本

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Resources": {
    "getAllItemsFunction": {
      "Type": "AWS::Lambda::Function",
      "Metadata": {
        "SamResourceId": "getAllItemsFunction"
      },
      "Properties": {
        "Code": {
          "S3Bucket": "amzn-s3-demo-source-bucket-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 行基礎設施程式碼。會將您的程式碼 AWS SAM 轉換為佈建應用程式所需的 200 多行 CloudFormation 程式碼。

什麼是 AWS SAM CLI？

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

- 快速初始化新的應用程式專案。
- 建置您的應用程式以進行部署。
- 執行本機偵錯和測試。
- 部署您的應用程式。
- 設定 CI/CD 部署管道。
- 在雲端中監控您的應用程式並進行故障診斷。
- 在您開發時同步本機變更至雲端。
- 還有更多！

與 AWS SAM CLI AWS SAM 和 CloudFormation 範本搭配使用時，最適合使用。它也適用於第三方產品，例如 Terraform。

初始化新專案

從入門範本中選取，或選擇自訂範本位置以開始新的專案。

在這裡，我們使用 `sam init` 命令來初始化新的應用程式專案。我們會選取要開始的 Hello World 範例專案。會 AWS SAM CLI 下載入門範本並建立專案資料夾目錄結構。

```
→ what-is sam init

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

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

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

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

如需詳細資訊，請參閱[在中建立您的應用程式 AWS SAM](#)。

建置您的應用程式以進行部署

封裝函數相依性並組織專案程式碼和資料夾結構，以準備部署。

在這裡，我們使用 `sam build` 命令來準備我們的應用程式以進行部署。AWS SAM CLI 會建立 `.aws-sam` 目錄，並在其中組織我們的應用程式相依性和檔案以進行部署。

```
→ sam-app sam build
Building codeuri: /Users/evzz/Demo/what-is/sam-app/hello_world runtime: python3.9 metadata: {} architecture: 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[建置您的應用程式](#)。

執行本機偵錯和測試

在本機機器上，模擬事件、測試 APIs、叫用 函數等，以偵錯和測試您的應用程式。

在這裡，我們使用 `sam local invoke` 命令在 `HelloWorldFunction` 本機叫用。為了達成此目的，AWS SAM CLI 會建立本機容器、建置函數、叫用它，並輸出結果。您可以使用 Docker 之類的應用程式在機器上執行容器。

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

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

如需詳細資訊，請參閱[測試您的應用程式](#)和[偵錯您的應用程式](#)。

部署您的應用程式

設定應用程式的部署設定，並部署到 AWS 雲端以佈建您的 資源。

在這裡，我們使用 `sam deploy --guided` 命令透過互動式流程部署我們的應用程式。會 AWS SAM CLI 引導我們設定應用程式的部署設定、將範本轉換為 CloudFormation，以及將 範本部署至 CloudFormation 以建立資源。

```
→ 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 部署管道

使用支援的 CI/CD 系統建立安全的持續整合和交付 (CI/CD) 管道。

在這裡，我們使用 `sam pipeline init --bootstrap` 命令來設定應用程式的 CI/CD 部署管道。會引導我們完成選項，並產生要用於 CI/CD AWS SAM CLI 系統 AWS 的資源和組態檔案。

[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]: 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 系統和管道部署](#)。

在雲端中監控您的應用程式並進行故障診斷

檢視已部署資源的重要資訊、收集日誌，以及利用內建監控工具，例如 AWS X-Ray。

在這裡，我們使用 `sam list` 命令來檢視我們部署的資源。我們取得 API 端點並叫用它，這會觸發我們的函數。然後，我們使用 `sam logs` 來檢視函數的日誌。

```
→ 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[監控您的應用程式](#)。

在您開發時同步本機變更至雲端

當您在本機電腦上開發時，會自動將變更同步至雲端。快速查看您的變更，並在雲端中執行測試和驗證。

在這裡，我們使用 `sam sync --watch` 命令讓 AWS SAM CLI 監看本機變更。我們會修改 `HelloWorldFunction` 程式碼，而 AWS SAM CLI 會自動偵測變更，並將我們的更新部署到雲端。

```
-----
Key           HelloWorldFunctionIamRole
Description   Implicit IAM Role created for Hello World function
Value         arn:aws:iam::513423067560:role/sam-app-HelloWorldFunctionRole-15GL0UR9LMT1W

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: {} architecture: x86_64 functions: HelloWorldFunction
Running PythonPipBuilder:CopySource
Finished syncing Lambda Function HelloWorldFunction.
█
```

在雲端測試支援的資源

叫用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。我們調用 Lambda 函數並接收其日誌和回應。設定 Lambda 函數以串流回應時 AWS SAM CLI，會即時將其回應串流回去。


```
→ lambda-streaming-app sam remote invoke StreamingFunction --stack-name lambda-streaming-app
```

進一步了解

若要繼續了解 AWS SAM，請參閱下列資源：

- [完整 AWS SAM 研討會](#) – 旨在教導您許多 AWS SAM 提供的主要功能的研討會。
- [使用 SAM 的工作階段](#) – 由我們的 AWS Serverless Developer Advocate 團隊在使用時建立的影片系列 AWS SAM。
- [無伺服器土地](#) – 將最新資訊、部落格、影片、程式碼和學習資源整合在一起的網站 AWS。

後續步驟

如果這是您第一次使用 AWS SAM，請參閱 [入門 AWS SAM](#)。

的無伺服器概念 AWS Serverless Application Model

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

無伺服器概念

事件驅動型架構

無伺服器應用程式包含個別 AWS 服務，例如 AWS Lambda 用於運算的 和用於資料庫管理的 Amazon DynamoDB，每個服務都會執行專門的角色。然後，這些服務會透過事件驅動架構彼此鬆散整合。若要進一步了解事件驅動型架構，請參閱[什麼是事件驅動型架構？](#)。

基礎設施即程式碼 (IaC)

基礎設施即程式碼 (IaC) 是一種處理基礎設施的方式，就像開發人員處理程式碼一樣，將相同的嚴格應用程式程式碼開發套用至基礎設施佈建。您可以在範本檔案中定義基礎設施、將其部署到 AWS，並為您 AWS 建立資源。使用 IaC，您可以在程式碼中定義 AWS 您要佈建的內容。如需可用於的 IaC 工具比較 AWS，請參閱[基礎設施即程式碼 \(IaC\)](#)。

無伺服器技術

透過無 AWS 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。所有伺服器管理都由 AWS 完成，提供許多好處，例如自動擴展和內建高可用性，讓您快速將想法帶入生產環境。使用無伺服器技術，您可以專注於產品的核心，而不必擔心管理和操作伺服器。若要進一步了解無伺服器，請參閱下列內容：

- [上的無伺服器 AWS](#)
- [無伺服器開發人員指南](#) – 提供 AWS 雲端中無伺服器開發的概念概觀。

如需核心無 AWS 伺服器服務的基本簡介，請參閱[Serverless 101：了解 Serverless Land 的無伺服器服務](#)。

無伺服器應用程式

使用時 AWS SAM，您會管理應用程式中的相關資源，其中包含您的 AWS SAM 專案和範本。應用程式中的所有資源都會在您的 AWS SAM 範本中定義或參考。當 AWS SAM 處理您的範本時，它會建立 CloudFormation 資源。在 CloudFormation 中，資源是在稱為堆疊的單一單位中管理，而堆疊中的所有資源都由堆疊的 CloudFormation 範本定義。

基礎設施即程式碼 (IaC)

透過基礎設施即程式碼 (IaC)，您可以自動化 AWS 資源的部署和管理，包括無伺服器應用程式。IaC 可讓您使用程式碼定義基礎設施，讓您更輕鬆地版本化、共用和複寫部署。此方法可協助您：

- 加速您的開發週期

- 簡化組態管理
- 改善部署的可靠性和一致性

無 AWS 伺服器應用程式的 IaC 工具

AWS 提供多種 IaC 工具，協助您建置、部署和管理雲端資源。本節說明如何 AWS SAM 符合此生態系統，以及如何與其他 AWS IaC 工具搭配使用。

AWS CloudFormation

使用 [CloudFormation](#)，您可以使用 YAML 或 JSON 範本來建立和佈建整個 AWS 基礎設施。會自動 CloudFormation 處理資源建立、更新和刪除。當您部署 AWS SAM 應用程式時，會 CloudFormation 處理轉換後的範本，以建立和管理資源。

AWS Serverless Application Model (AWS SAM)

AWS SAM 可協助您使用簡化的語法建置無伺服器應用程式，以定義無伺服器資源。您可以使用 AWS SAM 範本，使用簡潔 YAML 語法佈建 Lambda 事件函數、APIs、資料庫和事件來源。會在部署期間將這些範本 AWS SAM 轉換為 CloudFormation 範本。

Note

雖然 AWS SAM 專精於無伺服器應用程式，但您可以在 AWS SAM 範本中使用任何 CloudFormation 資源類型。這可讓您在需要時靈活地包含非伺服器資源。

AWS 雲端開發套件 (AWS CDK)

透過 [AWS CDK](#)，您可以使用熟悉的程式設計語言定義無伺服器基礎設施，例如 TypeScript、Python、Java、C#/。Net 或 Go。您可以使用迴圈和條件等程式設計建構來定義基礎設施，並 AWS CDK 產生用於部署的 CloudFormation 範本。您可以使用在 AWS SAM CLI 本機測試和偵錯使用建立的應用程式 AWS CDK。若要進一步了解，請參閱[在本機測試 CDK 應用程式](#)。

比較無伺服器應用程式的 IaC 工具

為無伺服器應用程式選擇 IaC 工具時，請考慮團隊的偏好設定、專案需求和現有的工作流程。下表比較無伺服器開發的 AWS IaC 工具的主要特性：

工具	主要用途	最適合	適用於 AWS SAM	何時選擇
CloudFormation	管理複雜的 AWS 基礎設施	需要詳細控制 AWS 資源的應用程式	AWS SAM 範本在部署期間轉換為 CloudFormation 範本	精細控制非伺服器資源
AWS SAM	無伺服器應用程式開發	使用 Lambda 建置無伺服器應用程式的團隊	原生功能	主要專注於具有 Lambda 函數、API Gateway APIs 和其他無伺服器資源的無伺服器架構時
AWS CDK	使用程式設計語言的基礎設施定義	團隊偏好類型語言和程式碼優先方法	產生 AWS SAM 範本並使用 AWS SAM CLI 進行測試	當您需要程式設計基礎設施定義或複雜的資源組態邏輯時

Note

雖然本指南著重於原生 IaC AWS 工具，但 Terraform 是另一個常用的 IaC 解決方案，可用於定義無伺服器應用程式。支援本機測試 Terraform AWS SAM CLI 中定義的 Lambda 函數。如需詳細資訊，請參閱[AWS SAM CLI Terraform 支援](#)。

進一步了解

- 若要進一步了解 上的 DevOps 實務 AWS，請參閱 [上的 DevOps 簡介 AWS](#)
- 如需搭配不同 IaC 工具使用 Lambda 的資訊，請參閱[搭配基礎設施做為程式碼使用 Lambda \(IaC\)](#)

入門 AWS SAM

透過檢閱並完成本節中的主題 AWS SAM 來開始使用。[AWS SAM 先決條件](#)提供設定 AWS 帳戶、建立 IAM 使用者、建立金鑰存取，以及安裝和設定 的詳細說明 AWS SAM CLI。完成先決條件後，您可以選擇開發方法：

- [安裝 AWS SAM CLI](#)，您可以在 Linux、Windows 和 macOS 作業系統上執行此操作。
- [使用 VS Code Toolkit 開始使用 AWS SAM](#)，透過整合偵錯和測試功能獲得視覺化開發體驗。

安裝完成後，您可以選擇逐步解說 AWS SAM Hello World 教學課程。遵循本教學課程將引導您完成使用 建立基本無伺服器應用程式的程序 AWS SAM。完成教學課程後，您將準備好檢閱 中詳述的概念[如何使用 AWS Serverless Application Model \(AWS SAM\)](#)。

主題

- [AWS SAM 先決條件](#)
- [安裝 AWS SAM CLI](#)
- [教學課程：使用 部署 Hello World 應用程式 AWS SAM](#)
- [將 Lambda 事件函數轉換為 AWS SAM 應用程式](#)

AWS SAM 先決條件

在安裝和使用 AWS Serverless Application Model 命令列界面 () 之前，請先完成下列先決條件AWS SAM CLI。

若要使用 AWS SAM CLI，您需要下列項目：

- AWS 帳戶、AWS Identity and Access Management (IAM) 登入資料和 IAM 存取金鑰對。
- 用來設定 AWS 登入資料的 AWS Command Line Interface (AWS CLI)。

主題

- [註冊 AWS 帳戶](#)
- [步驟 1：安裝 AWS CLI](#)
- [步驟 2：使用 AWS CLI 設定 AWS 登入資料](#)

- [步驟 3：\(選用\) 安裝 AWS Toolkit for VS 程式碼](#)
- [後續步驟](#)

註冊 AWS 帳戶

若要開始使用 AWS，您需要 AWS 帳戶。如需建立的相關資訊 AWS 帳戶，請參閱《AWS 帳戶管理參考指南》中的 [入門 AWS 帳戶](#)。

步驟 1：安裝 AWS CLI

是一種 AWS CLI 開放原始碼工具，可讓您 AWS 服務 在命令列 Shell 中使用命令與互動。對於 AWS SAM CLI設定登入資料等活動 AWS CLI，需要。若要進一步了解 AWS CLI，請參閱[什麼是 AWS Command Line Interface ?](#) AWS Command Line Interface 《使用者指南》中的。

若要安裝 AWS CLI，請參閱AWS Command Line Interface 《使用者指南》中的[安裝或更新最新版本的 AWS CLI](#)。

步驟 2：使用 AWS CLI 設定 AWS 登入資料

使用 IAM Identity Center 設定登入資料

- 若要使用 IAM Identity Center 設定登入資料，請參閱[使用設定 sso 精靈 AWS 設定您的設定檔](#)。

使用 設定登入資料 AWS CLI

1. 從aws configure命令列執行 命令。
2. 設定下列項目。選取每個連結以進一步了解：
 - a. [存取金鑰 ID](#)
 - b. [私密存取金鑰](#)
 - 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]: wJa1rXUtnFEMI/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` 檔案中名為 `default` 的設定檔（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之設定檔的 AWS CLI 命令時，會使用此設定檔中的資訊。如需 `credentials` 檔案的詳細資訊，請參閱 AWS Command Line Interface 《使用者指南》中的[組態和登入資料檔案設定](#)。

如需設定登入資料的詳細資訊，例如使用現有的組態和登入資料檔案，請參閱 AWS Command Line Interface 《使用者指南》中的[快速設定](#)。

步驟 3：（選用）安裝 AWS Toolkit for VS 程式碼

對於偏好整合式開發環境的開發人員，AWS Toolkit for VS Code 為無伺服器開發提供了強大的功能，包括視覺化偵錯、CodeLens 整合和簡化的部署工作流程。

VS 程式碼開發的先決條件

- 安裝在系統上的 Visual Studio Code (1.73.0 版或更新版本)
- VS 程式碼的 YAML 語言支援延伸

安裝 AWS Toolkit for VS 程式碼

1. 開啟 Visual Studio 程式碼
2. 開啟延伸模組檢視 (Ctrl+Shift+X 或 Cmd+Shift+X)
3. 搜尋「AWS Toolkit」
4. Amazon Web Services 安裝「AWS Toolkit」延伸模組
5. Red Hat 安裝「YAML」延伸模組 (SAM 範本 CodeLens 功能需要)

搭配 SAM 使用 VS AWS 程式碼的優點

- 視覺化偵錯：在本機設定中斷點並逐步完成 Lambda 事件函數
- CodeLens 整合：直接從 SAM 範本建置、部署和叫用函數
- 整合式終端機：在不離開編輯器的情況下存取 AWS SAM AWS CLI 命令
- 範本驗證：即時驗證和 IntelliSense for SAM 範本

如需有關在 VS 程式碼中設定 AWS 登入資料的資訊，請參閱 AWS Toolkit for VS Code 使用者指南中的[設定登入資料](#)。

後續步驟

- 您現在可以安裝 AWS SAM CLI 並開始使用 AWS SAM。若要安裝 AWS SAM CLI，請參閱 [安裝 AWS SAM CLI](#)。
- 若要設定 Visual Studio Code 進行無伺服器開發，請參閱[設定 AWS Toolkit for VS Code](#)。

安裝 AWS SAM CLI

遵循 中的指示，在支援的作業系統上安裝最新版本的 AWS Serverless Application Model 命令列界面 (AWS SAM CLI) [步驟 1：安裝 AWS CLI](#)。

如需管理目前安裝的 版本的資訊 AWS SAM CLI，包括如何升級、解除安裝或管理夜間組建，請參閱[管理 AWS SAM CLI 版本](#)。

 這是您第一次安裝 AWS SAM CLI 嗎？

在繼續之前，請先完成上一節中的所有[先決條件](#)。其中包含：

1. 註冊 AWS 帳戶。
2. 設定 的安全存取 AWS。
3. 建立存取金鑰 ID 和私密存取金鑰。
4. 安裝 AWS CLI。
5. 設定 AWS 登入資料。

主題

- [安裝 AWS SAM CLI](#)
- [對安裝錯誤進行故障診斷](#)
- [後續步驟](#)
- [選用：驗證安裝程式的 AWS SAM CLI 完整性](#)

安裝 AWS SAM CLI

Note

從 2023 年 9 月開始，AWS 將不再維護 AWS SAM CLI() 的 AWS 受管 Homebrew 安裝程式 `aws/tap/aws-sam-cli`。如果您使用 Homebrew 安裝和管理 AWS SAM CLI，請參閱下列選項：

- 若要繼續使用 Homebrew，您可以使用 社群受管安裝程式。如需詳細資訊，請參閱 [使用 AWS SAM CLI 管理 Homebrew](#)。
- 我們建議您使用此頁面中記錄的其中一個第一方安裝方法。使用這些方法之一之前，請參閱 [從 切換 Homebrew](#)。
- 如需其他詳細資訊，請參閱 [發行版本：1.121.0](#)。

若要安裝 AWS SAM CLI，請遵循作業系統的指示。

Linux

x86_64 - command line installer

1. 將 [AWS SAM CLI .zip 檔案](#) 下載到您選擇的目錄。
2. （選用）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 [選用：驗證安裝程式的 AWS SAM CLI 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. 執行可執行檔來安裝 AWS SAM CLI `install`。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，您應該會看到輸出，以最新的可用版本取代下列括號文字：

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

arm64 - command line installer

1. 將 [AWS SAM CLI .zip 檔案](#) 下載到您選擇的目錄。
2. （選用）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[選用：驗證安裝程式的 AWS SAM CLI 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. 執行可執行檔來安裝 AWS SAM CLI `install`。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，您應該會看到如下所示的輸出，但會將括號文字取代為最新的 SAM CLI 版本：

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

macOS

安裝步驟

使用 套件安裝程式來安裝 AWS SAM CLI。此外，套件安裝程式有兩種安裝方法供您選擇：GUI 和命令列。您可以為所有使用者或只為目前的使用者安裝。若要為所有使用者安裝，需要超級使用者授權。

Note

AWS SAM CLI 不支援比 macOS 13.x 更舊的 macOS 版本。升級至支援的 macOS AWS SAM 版本，或使用安裝 CLIHomebrew。

GUI - All users

若要下載套件安裝程式並安裝 AWS SAM CLI

Note

如果您之前透過 AWS SAM CLIHomebrew 或安裝 pip，則需要先將其解除安裝。如需說明，請參閱[解除安裝 AWS SAM CLI](#)。

1. 將 macOS 下載pkg至您選擇的目錄：

- 對於執行 Intel 處理器的 Mac，請選擇 x86_64 – [aws-sam-cli-macos-x86_64.pkg](#)
- 對於執行 Apple 晶片的 Mac，請選擇 arm64 – [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[選用：驗證安裝程式的 AWS SAM CLI 完整性](#)。

2. 執行您下載的檔案，並依照畫面上的指示繼續進行簡介、讀我檔案和授權步驟。
3. 針對目的地選取，選取此電腦所有使用者的安裝。
4. 針對安裝類型，選擇將要安裝 AWS SAM CLI 的位置，然後按安裝。建議的預設位置為 `/usr/local/aws-sam-cli`。

Note

若要使用 `sam` 命令叫用 AWS SAM CLI，安裝程式會自動在 `/usr/local/bin/sam` 和 `/usr/local/aws-sam-cli/sam` 之間建立符號連結，或是您選擇的安裝資料夾。

5. AWS SAM CLI 將安裝，並顯示安裝成功訊息。按關閉。

驗證成功安裝

- 執行下列 AWS SAM CLI 動作，確認已正確安裝，且您的符號連結已設定完成：

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

GUI - Current user**下載並安裝 AWS SAM CLI****Note**

如果您之前透過 AWS SAM CLI Homebrew 或安裝 pip，則需要先將其解除安裝。如需說明，請參閱[解除安裝 AWS SAM CLI](#)。

1. 將 macOS 下載 pkg 至您選擇的目錄：
 - 對於執行 Intel 處理器的 Mac，請選擇 x86_64 – [aws-sam-cli-macos-x86_64.pkg](#)
 - 對於執行 Apple 晶片的 Mac，請選擇 arm64 – [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[選用：驗證安裝程式的 AWS SAM CLI 完整性](#)。

2. 執行您下載的檔案，並依照畫面上的指示繼續進行簡介、讀我檔案和授權步驟。
3. 針對目的地選取，選取僅為我安裝。如果您沒有看到此選項，請前往下一個步驟。
4. 對於安裝類型，請執行下列動作：
 1. 選擇將要安裝 AWS SAM CLI 的位置。預設位置為 `/usr/local/aws-sam-cli`。選取您擁有寫入許可的位置。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. 如果您未在上一個步驟中取得僅為我選擇安裝的選項，請選取變更安裝位置 > 僅為我安裝，然後按繼續。
 3. 按下安裝。
5. AWS SAM CLI 將安裝，並顯示安裝成功訊息。按關閉。

建立符號連結

- 若要使用 AWS SAM CLI `sam` 命令叫用，您必須在程式與之間 AWS SAM CLI 手動建立符號連結 `$PATH`。修改並執行下列命令來建立您的符號連結：

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

- **sudo** – 如果您的使用者具有的寫入許可 `$PATH`，`sudo` 則不需要。否則，`sudo` 是必要的。
- **path-to** – 您安裝 AWS SAM CLI 程式的路徑。例如 `/Users/myUser/Desktop`。
- **path-to-symlink-directory** – 您的 `$PATH` 環境變數。預設位置為 `/usr/local/bin`。

驗證成功安裝

- 執行下列 AWS SAM CLI 動作，確認已正確安裝，且您的符號連結已設定完成：

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

Command line - All users

下載並安裝 AWS SAM CLI

Note

如果您之前透過 AWS SAM CLI Homebrew 或安裝 pip，則需要先將其解除安裝。如需說明，請參閱[解除安裝 AWS SAM CLI](#)。

1. 將 macOS 下載 pkg 至您選擇的目錄：

- 對於執行 Intel 處理器的 Mac，請選擇 x86_64 – [aws-sam-cli-macos-x86_64.pkg](#)
- 對於執行 Apple 晶片的 Mac，請選擇 arm64 – [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[選用：驗證安裝程式的 AWS SAM CLI 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

若要使用 AWS SAM CLI `sam` 命令叫用，安裝程式會自動在 `/usr/local/bin/sam` 和 `/usr/local/aws-sam-cli/sam` 之間建立符號連結。

驗證成功安裝

- 執行下列 AWS SAM CLI 動作，確認已正確安裝，且您的符號連結已設定完成：

```
$ which sam
```

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

Command line - Current user

下載並安裝 AWS SAM CLI

Note

如果您之前透過 AWS SAM CLI Homebrew 或安裝 pip，則需要先將其解除安裝。如需說明，請參閱[解除安裝 AWS SAM CLI](#)。

1. 將 macOS 下載 pkg 至您選擇的目錄：

- 對於執行 Intel 處理器的 Mac，請選擇 x86_64 – [aws-sam-cli-macos-x86_64.pkg](#)
- 對於執行 Apple 晶片的 Mac，請選擇 arm64 – [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[選用：驗證安裝程式的 AWS SAM CLI 完整性](#)。

- #### 2. 判斷您有寫入許可的安裝目錄。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，如果您將 *path-to-my-directory* 取代為 /Users/myUser/Desktop，則會在該處安裝 aws-sam-cli 程式資料夾。

```
<?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>
```

- 執行下列動作來儲存xml檔案並確認其有效：

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

輸出應會顯示將套用至程式的 AWS SAM CLI 偏好設定。

- 執行下列動作來安裝 AWS SAM CLI：

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

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

建立符號連結

- 若要使用 AWS SAM CLI 的 `aws-sam-cli` 命令，您必須在程式與 `aws-sam-cli` 之間手動建立符號連結 `$PATH`。修改並執行下列命令來建立您的符號連結：

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

- sudo** – 如果您的使用者具有 `$PATH` 的寫入許可，`sudo` 則不需要。否則，`sudo` 是必要的。
- path-to** – 您安裝 AWS SAM CLI 程式的路徑。例如 `/Users/myUser/Desktop`。
- path-to-symlink-directory** – 您的 `$PATH` 環境變數。預設位置為 `/usr/local/bin`。

驗證成功安裝

- 執行下列 AWS SAM CLI 動作，確認 已正確安裝，且您的符號連結已設定完成：

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

Windows

Windows Installer (MSI) 檔案是 Windows 作業系統的套件安裝程式檔案。

請依照下列步驟，使用 MSI 檔案安裝 AWS SAM CLI。

- 下載 AWS SAM CLI [64 位元](#)。
- (選用) 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[選用：驗證安裝程式的 AWS SAM CLI 完整性](#)。
- 驗證安裝。

完成安裝後，請開啟新的命令提示或 PowerShell 提示來驗證它。您應該能夠 sam 從命令列叫用。

```
sam --version
```

成功安裝 後 AWS SAM CLI，您應該會看到如下所示的輸出：

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

- 啟用長路徑 (僅限 Windows 10 和更新版本)。

Important

AWS SAM CLI 可能會與超過 Windows 路徑上限的檔案路徑互動。由於 Windows 10 MAX_PATH 限制 sam init，這可能會導致執行時發生錯誤。若要解決此問題，必須設定新的長路徑行為。

若要啟用長路徑，請參閱 Microsoft [Windows 應用程式開發文件中的在 Windows 10 版本 1607 和更新版本中啟用長路徑](#)。

5. 安裝 Git。

若要使用 `sam init` 命令下載範例應用程式，您還必須安裝 Git。如需說明，請參閱[安裝 Git](#)。

對安裝錯誤進行故障診斷

Linux

Docker 錯誤：「無法連線至 Docker 協助程式。是否在此主機上執行 docker 協助程式？」

在某些情況下，若要提供 `ec2-user` 存取 Docker 協助程式的許可，您可能需要重新啟動執行個體。如果您收到此錯誤，請嘗試重新啟動執行個體。

Shell 錯誤：「找不到命令」

如果您收到此錯誤，您的 shell 就無法在 AWS SAM CLI 路徑中找到可執行檔。驗證安裝 AWS SAM CLI 可執行檔的目錄位置，然後驗證目錄是否位於您的路徑上。

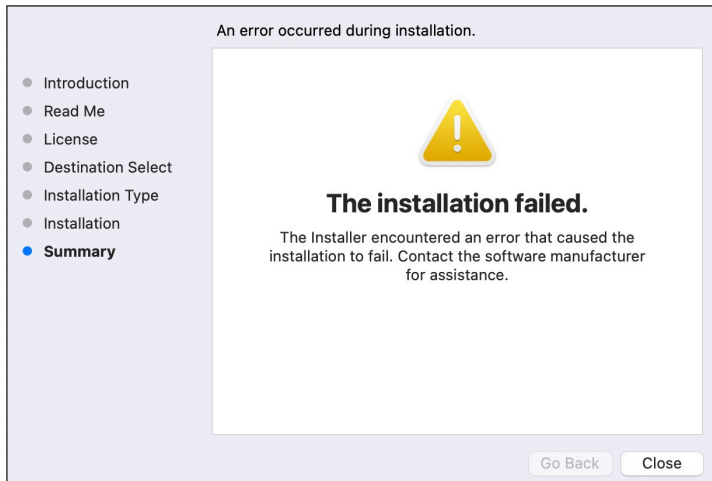
AWS SAM CLI 錯誤：`"/lib64/libc.so.6：找不到版本 `GLIBC_2.14'（需要 /usr/local/aws-sam-cli/dist/libz.so.1）"`

如果您收到此錯誤，表示您使用的是不支援的 Linux 版本，而且內建的 glibc 版本已過期。請嘗試下列其中一項：

- 將您的 Linux 主機升級至 CentOS、Fedora、Ubuntu 或 Amazon Linux 2 最近發行版本的 64 位元版本。
- 請遵循的指示[安裝 AWS SAM CLI](#)。

macOS

安裝失敗



如果您要為使用者安裝 AWS SAM CLI，並選取您沒有寫入許可的安裝目錄，則可能會發生此錯誤。請嘗試下列其中一項：

1. 選取您具有寫入許可的不同安裝目錄。
2. 刪除安裝程式。然後，下載並再次執行。

後續步驟

若要進一步了解 AWS SAM CLI 和 以開始建置您自己的無伺服器應用程式，請參閱以下內容：

- [教學課程：使用 部署 Hello World 應用程式 AWS SAM](#) – Step-by-step 說明。
- [完成 AWS SAM 研討會](#) – 旨在教導您許多 AWS SAM 提供的主要功能的研討會。
- [AWS SAM 範例應用程式和模式](#) – 社群作者的應用程式和模式範例，您可以進一步實驗。

選用：驗證安裝程式的 AWS SAM CLI 完整性

使用套件安裝程式安裝 AWS Serverless Application Model 命令列界面 (AWS SAM CLI) 時，您可以在安裝之前驗證其完整性。這是選用但強烈建議的步驟。

您有兩種可用的驗證選項：

- 驗證套件安裝程式簽章檔案。
- 驗證套件安裝程式雜湊值。

適用於您的平台時，建議您驗證簽章檔案選項。此選項提供額外的安全層，因為金鑰值會在此處發佈，並與儲存GitHub庫分開管理。

主題

- [驗證安裝程式簽章檔案](#)
- [驗證雜湊值](#)

驗證安裝程式簽章檔案

Linux

arm64 - 命令列安裝程式

AWS SAM 使用 [GnuPG](#) 簽署 AWS SAM CLI .zip 安裝程式。驗證會在下列步驟中執行：

1. 使用主要公有金鑰來驗證簽署者公有金鑰。
2. 使用簽署者公有金鑰來驗證 AWS SAM CLI 套件安裝程式。

驗證簽署者公有金鑰的完整性

1. 複製主要公有金鑰，並將其儲存為 .txt 檔案到您的本機電腦。例如 *primary-public-key.txt*。

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

mQINBGRuSzMBEADsqiw0y78w7F4+sshaMFRIwRGNRm94p5Qey2KMZBxekFtoryVD
D9jE0nvupx4tvhfBHz5EcUHCE0d14MTqdBy6vVAshozgxVb9RE8JpECn5lw7XC69
4Y7Gy1TKKQMEwtDXElkGxIFdUWvWjSnPlzfnoXwQYGeE93CUS3h5dImp22Yk1Ct6
eGghlcbg1X4L8EpFMj7GvcsU8f7ziVI/PyC1Xwy39Q8/I67ip5eU5ddx0/xHqrbL
YC7+8pJPbRMej2twT2LrcpWWYAbprMtRoA6WfE0/thoo3xhHpIMHdPFAA86ZNGIN
kRLjGUg7jnPTRW40in3pCc8nT4Tfc1QERkHm641gTC/jUvpmQsM6h/FUVP2i5iE/
JHpJcMuL2Mg6zDo3x+3gTCf+Wqz3rZzxB+wQT3yryZs6efcQy7nR0iRxYBxCSXX0
2cNYzsYLB/bYaW8yqWIHD5IQKhW269gp2E5Khs60zgS3CorMb5/xHgXjUCVgcu8a
a8ncdf9fj13WS5p0ohetPb02ZjWv+MaqrZ0mUIgKbA4RpWZ/fu97P5BW9ylwmIDB
sWy0cMxg8MlvSdLytPieogaM0qMg3u5qXRGBr6Wmevkty0qgnmpGGc5zPiUbt0E8
CnFFqyxBpj5I0nG0KZGVihvn+iRxrv6G07Ww092+Dc6m94U0EEiBR7Qi0wARAQAB
tDRBV1MgU0FNIENMSSBQcm1tYXJ5IDxhd3MtY2FtLWNsaS1wcm1tYXJ5QGFTYXpv
bi5jb20+iQI/BMBBCQApBQJkbksZAhsvBQkHhM4ABwsJCAcDAgEGFQgCCQoLBBYC
AwECHgECF4AACgkQQv1fen0tiFqTuhAAzi5+ju5UV0WqHKEv0JS008T4QB8HcqAE
```

```
SV03mY6/j29knkcL8ubZP/DbpV7QpHPI2PB5qSXsiDTP3IYPbeY78zHSDjljaIK3
njJLMScFeGPyfPpwMsuY4nztRIGAtXShPA8N/k4ZJcafnpNqKj7QnPxIC1KaIQWm
p0tVb8msUF3/s0UTa5Ys/1NRhVC0eGg32ogXGdojZA2kHZWdm9udLo4CDrDcrQT7
NtDcJASapXSQL63XfAS3snEc4e1941YxcjFYZ33rel8K9juyDZfi1slWR/L3AviI
QFIaqSHzy0tP1oinUkoVwL8ThevKD3Ag9CZf1ZLzNCV7yqlF8R1hEZ4zcE/3s9E1
WzCFsozb5HfE1AZonmrDh3Sy0EIBMCS6vG5dWnvJrAuSYv2rX38++K5Pr/MIAf0X
D0I1rtA+XDshNv9lSwSy0lt+iClawZAN09IXCiN1r0YcVQlwzDFwCNWDgkwd0qS0
g0A2f8NF91E5nBbeEuYquo0l1Vy8+ICbg0Fs9LoWZlnVh7/RyY6ssowiU9vGUUnHI
L8f9jqRspIz/Fm3JD86ntZxLVGkeZUz62FqErdohYfkFIVcv7G0NTEyrz5HL1npv
FJ0MR0HjrMrZrn0VZnwBKhpLocTsH+3t5It4ReYEX0f1DIOL/KRwPvjMvBVkXY5
hblRVDQo0Wc=
=d9oG
-----END PGP PUBLIC KEY BLOCK-----
```

2. 將主要公有金鑰匯入至 keyring。

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

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

3. 複製簽署者公有金鑰，並將其儲存為 .txt 檔案到您的本機電腦。例如 *signer-public-key.txt*。

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

mQINBGgrxIgBEADGCTudveeeVbWpZDGX9Ni57mBRMVSJwQJ6F/PC34jw0DozxTtd
H+ZPsXLvLwerN/DVXBk8E1qNZ5RGptak8j7MPz+MC3n4txibEJpB61vpjJJM+9cC
7whaMLDT/SbykHYXdrnHqa8KsUJ17rPLJcaRN722NSxvYVMI0A9ffVXV7cfEyZi5
MbYF2Gc9LNBKaknImIva7EKe2/wI6YCqC5yytyfWU5dL6oHXsgTnFL9mhziMxv
WhyzawyJG6EJZsJ3WL1bIKApN6XZSxyCx0v1BrebYZjD5v0nA+TJaQ7is8atjtOI
DGe0AViw7k08ChTpjA7YG/Uu7n/Fy7qLF/3Nz0b6cBNjemjBazQ3A3KNCpi5hqFM
Uo1WpoVLr5CXQnc0B3fBUntIoxi0Sk5MKjH9AbYxfqgEX0ZJB9hAlc6LIEy0Yru6
```

```
MMBrIHE86IM11nFE/DeLnCdPG23+1Pttyw0t3+9z5QwmPe3VPpEfCySPcdxHKZSP
rLile8qDznEvLPDvQ0qkBxdMtVa2yct5VJkdqy6UrN2xa0dpspHjRUjHh/EY/xMt
fwMUj0KohaZ/1pjotCcksAsZWUxCNcFvLYxuxeytVk4F09Es1hj4ihhLUI+43/ic
3DHSEiext7Q8/UccNArkSCT7U0vvL7QTuP+pjYTyiC8Vx6g/Y5Ht5+qywARAQAB
tDBBV1MgU0FNIENMSSBUZWFtIDxhd3Mtc2FtLWNsaS1zaWduZXJAYW1hem9uLmNv
bT6JAj8EEwEJACKFAMgrxIgCGy8FCQPCZwAHCwkIBwMCAQYVCAIJCgsEFgIDAQIe
AQIXgAAKCRBA1Kuxvt/atJo6EAC/5C8uJs76W5f5V5XNAMzwBFiZuYpop3DRReCo
P68ZZylokAC9ShRZnIOujpDJt1NS7T/G00BzmcpSPkYYE531ALaXcHWmb9XV0Ajg
J8iboAVBLY0C7mhL/cbJ3v9Q1pXXjyTuhexkJCV8rdHVX/0H8WqTZp1EaRuZ7p8q
PMxddg4ClwstYuh30/dmNd1Gqfb4Fqy8MnV1yGSXR5Jf+sD1N2U04mbpyk/mr1c
f/jFmx86IkCWJVvdXWCVTe2AFy3NHCdLtdnEvFhokCOQd9wibUWX0j9vq4cVRZT
qamnpAQa0lH3lX0wrjqo8b1AIPoRWSfMtCYvh6kA8MAJv4cAzanzXILSLt0E0mzaU
qp5qoy37wNIjeztX6c/q4wss05qTlJhnNu4s3nh5VHultooaYpmDxp+ala5TweuM
KZDI4KdAGF4z0Raif+N53nd0YIiXkY0goUbsPCnVrCwoK9PjjyoJncq7c14wN150
IQUZEjyYAQDGGqs5XSfY4zW2cCXatrfozKF7R1kSU14DfJwPUyksoNAQEQuezfXyq
kr0gfIWK1r2nMdqS7WgSx/ypS5kdyrHuPZdaYfEVtuezoT2lQQx0SZqq1p5hI4R
nqmPte53WXJhbC0tgTIJWn+Uy/d5Q/aSIfD6o8gNLS1BDs1j1ku0XKu1sFCHUCZG
aerdsIkCHAQQAQkABgUCaCvFeAAKCRBC/V96c62IWt3/D/9g0LzWtz62lqJRCsri
wcA/yz88ayKb/GUv3FCT5Nd9JZt8y1tW+AE3SPTdcpfZmt5UN2sRzlj061mpKJzp
eBvYQ9og/34ZrRQqeg8bz02u34LKY11gD0xY0bWtB7TGIXIZZYqZECOPR0Dp6ZzB
abzkRSsJkEk0vbZzJhFWFYs98qfp/G0suFSBE7908Am33DB2jQ/Sol1h1VmNE6Sv
EOgR6+2yEkS2D0+msJMa/V82v9gBTPnxS1NV1d8Dduvt9rbM3LoxiNXUgx/s52yY
U6H3bwUcQ3UY6uRe1UWo5QnMFcDwfg43+q5rmjB4xQyX/BaQyF5K0hZyG+42/pH1
EMw18qN617FTxo3hVQUi/cBah1hQ8EVYsGnHDVxLCisbq5iZvp7+XtmMy1Q417gT
EQRo8feJh31e1GWlccVR2pZgIm1PQ69dzzseHnnKkGhifik0bDGo5/IH2EgI1KFn
SG399RMU/qRz0PLVP3i+zSJmhMqG8cnZaUwE5V4P21vQSc1hhd2Hv/C4SVKNqA2i
+oZbHj2vAkuzTTL075AoANebEjPGqwsKZi5mWUE5Pa931JeiXxWZ1EB7rkgQ1PAB
fsDBhYLt4MxCWAhifLMA6uQ4BhXu2RuX0qNfSbqa8jVF6DB6cD8eAHGpPKfJ0130
LtZnq+n4SfeNbZjD2FQWZR4CrA==
=lHfs
-----END PGP PUBLIC KEY BLOCK-----
```

4. 將簽署者公有金鑰匯入至 keyring。

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

```
gpg: key 4094ABB1BEDFDAB4: 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
```

請記下輸出的索引鍵值。例如 **4094ABB1BEDFDAB4**。

5. 使用金鑰值來取得和驗證簽署者公有金鑰指紋。

```
$ gpg --fingerprint 4094ABB1BEDFDAB4

pub   rsa4096 2025-05-19 [SCEA] [expires: 2027-05-19]
      EF46 3E86 CA31 933B B688  CC1A 4094 ABB1 BEDF DAB4
uid           [ unknown] AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>
```

指紋應符合下列項目：

```
EF46 3E86 CA31 933B B688  CC1A 4094 ABB1 BEDF DAB4
```

如果指紋字串不相符，請勿使用 AWS SAM CLI 安裝程式。在 [aws-sam-cli GitHub 儲存庫](#) 中 [建立問題](#)，向 AWS SAM 團隊呈報。

6. 驗證簽署者公有金鑰的簽章：

```
$ gpg --check-sigs 4094ABB1BEDFDAB4

pub   rsa4096 2025-05-19 [SCEA] [expires: 2027-05-19]
      EF463E86CA31933BB688CC1A4094ABB1BEDFDAB4
uid           [ unknown] AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>
sig!3        4094ABB1BEDFDAB4 2025-05-19  [self-signature]
sig!         42FD5F7A73AD885A 2025-05-19 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，將主要和簽署者公有金鑰匯入至 keyring。

您應該會看到列出的主要公有金鑰和簽署者公有金鑰的金鑰值。

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

驗證套件安裝程式的 AWS SAM CLI 完整性

1. 取得 AWS SAM CLI 套件簽章檔案 – 使用下列命令下載套件安裝程式的 AWS SAM CLI 簽章檔案：

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

2. 驗證簽章檔案 – 將下載的 .sig 和 .zip 檔案做為參數傳遞至 gpg 命令。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 Mon 19 May 2025 01:21:57 AM UTC using RSA key ID
4094ABB1BEDFDAB4
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: EF46 3E86 CA31 933B B688 CC1A 4094 ABB1 BEDF DAB4
```

- 您可以忽略WARNING: This key is not certified with a trusted signature! 訊息。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。如需詳細資訊，請參閱[信任 Web](#)。
- 如果輸出包含片語 BAD signature，請檢查您是否正確執执行程序。如果您繼續取得此回應，請在 aws-sam-cli GitHub 儲存庫中[建立問題](#)，並避免使用下載的檔案，以向 AWS SAM 團隊呈報。

Good signature from "AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>"
訊息表示簽章已經過驗證，您可以繼續進行安裝。

x86_64 - 命令列安裝程式

AWS SAM 使用 [GnuPG](#) 簽署 AWS SAM CLI .zip 安裝程式。驗證會在下列步驟中執行：

1. 使用主要公有金鑰來驗證簽署者公有金鑰。
2. 使用簽署者公有金鑰來驗證 AWS SAM CLI 套件安裝程式。

驗證簽署者公有金鑰的完整性

1. 複製主要公有金鑰，並將其儲存為 .txt 檔案到您的本機電腦。例如 *primary-public-key.txt*。

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



```
mQINBGRuSzMBEADsqiw0y78w7F4+sshaMFRiWRGNRM94p5Qey2KMZBxekFtoryVD
D9jE0nvupx4tvhfBH5EcUHCE0d14MTqdBy6vVAshozgxVb9RE8JpECn5lw7XC69
4Y7Gy1TKKQMEwtDXElkGxIFdUWvWjSnPlzfnoXwQYGeE93CUS3h5dImp22Yk1Ct6
eGghlcbg1X4L8EpFMj7GvcsU8f7ziVI/PyC1Xwy39Q8/I67ip5eU5ddx0/xHqrbL
YC7+8pJPbRMej2twT2LrcpWWYAbprMtRoa6WfE0/thoo3xhHpIMHdPfAA86ZNGIN
kRLjGUg7jnPTRW40in3pCc8nT4Tfc1QERkHm641gTC/jUvpmQsM6h/FUVP2i5iE/
JHpJcMuL2Mg6zDo3x+3gTCf+Wqz3rZzxB+wQT3yryZs6efcQy7nR0iRxYBxCSXX0
2cNYzsYLB/bYaW8yqWIHD5IQKhW269gp2E5Khs60zgS3CorMb5/xHgXjUCVgcu8a
a8ncdf9fj13WS5p0ohetPb02ZjWv+MaqrZ0mUIgKbA4RpWZ/fU97P5BW9ylwmIDB
sWy0cMxg8MlvSdLytPieogaM0qMg3u5qXRGBr6Wmevkty0qgnmpGGc5zPiUbt0E8
CnFFqyxBpj5I0nG0KZGVihvn+iRrxv6G07WW092+Dc6m94U0EEiBR7Qi0wARAQAB
tDRBV1MgU0FNIENMSSBQcm1tYXJ5IDxhd3Mtc2FtLWNsaS1wcm1tYXJ5QGFTYXpv
bi5jb20+iQI/BBMBCQApBQJkbksZAhsvBQkHhM4ABwsJCAcDAgEGFQgCCQoLBBYC
AwECHgECF4AACgkQQv1fen0tiFqTuhAAzi5+ju5UV0WqHkev0JS008T4QB8HcqAE
SV03mY6/j29knkcL8ubZP/DbpV7QpHPi2PB5qSXsiDTP3IYPbeY78zHSDjljaIK3
njJLMScFeGPpyfPpwMsuY4nziRIgAtXShPA8N/k4ZJcafnpNqKj7QnPxIC1KaIQWm
p0tVb8msUF3/s0UTa5Ys/1NRhVC0eGg32ogXGdojZA2kHZWdm9udLo4CDrDcrQT7
NtDcJASapXSQL63XfAS3snEc4e1941YxcjFYZ33rel8K9juyDZfi1slWR/L3AviI
QFIaqSHzy0tP1oinUkoVwL8ThevKD3Ag9CZf1ZLzNCV7yq1F8R1hEZ4zcE/3s9E1
WzCFsozb5HfE1AZonmrDh3Sy0EIBMCS6vG5dWnvJrAuSYv2rX38++K5Pr/MIAf0X
D0I1rtA+XDShNV91SwSy01t+iClawZAN09IXCiN1r0YcVQ1wzDFwCNWDgkwd0qS0
g0A2f8NF91E5nBbeEuYquo011Vy8+ICbg0Fs9LoWZlnVh7/RyY6ssowiU9vGUnHI
L8f9jqRspIz/Fm3JD86ntZxLVGkeZUz62FqErdohYfkFIVcv7G0NTEyrz5HL1npv
FJ0MR0HjrMrZrn0VZnwBKhpBLocTsH+3t5It4ReYEX0f1DIOL/KRwPvjMvBVkXY5
hb1RVDQo0Wc=
=d9oG
-----END PGP PUBLIC KEY BLOCK-----
```

2. 將主要公有金鑰匯入至 keyring。

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

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

- 複製簽署者公有金鑰，並將其儲存為 `.txt` 檔案到您的本機電腦。例如 `signer-public-key.txt`。

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

mQINBGgrxIgBEADGCTudveeeVbWpZDGX9Ni57mBRMVSJwQJ6F/PC34jw0DozxTtd
H+ZPsXLvLwerN/DVXBk8E1qNZ5RGptak8j7MPz+MC3n4txibEJpB61vpjJJM+9cC
7whaMLDT/SbykHYXdrnHqa8KsUJ17rPLJcaRN722NSxvYVMI0A9ffVXV7cfEyZi5
MbYF2Gc9LnbKaknImIva7EKeeh2/wI6YCqC5yytyfWU5dL6oHXsgTnFL9mhziMxv
WhyzawyJG6EJZsJ3WL1bIKApN6XZSXYCx0v1BrebYZjD5v0nA+TJaQ7is8atjtOI
DGe0AViw7k08ChTpjA7YG/Uu7n/Fy7qLF/3Nz0b6cBNjemjBazQ3A3KNCpi5hqFM
Uo1WpovLr5CXQnc0B3fBUntIoxi0Sk5MKjH9AbYxfqgEX0ZJB9hAlc6LIEy0Yru6
MMBrIHE86IM11nFE/DeLnCdPG23+1Pttwy0t3+9z5QwmPe3VPPEfCySPcdxHKZSP
rLile8qDznEv1PDvQ0qkBxdMtVa2yct5VJkdqy6UrN2xa0dpspHjRUjHh/EY/xMt
fwMUj0KohaZ/1pjotCcksAsZWUxCNcFvLYxuxeytVk4F09Es1hj4ihhLUI+43/ic
3DHSEiext7Q8/UccNArkhSCT7U0vvL7QTuP+pjYTyiC8Vx6g/Y5Ht5+qywARAQAB
tDBBV1MgU0FNIENMSSBUZWFtIDxhd3Mt2FtLWNsaS1zaWduZXJAYW1hem9uLmNv
bT6JAj8EEwEJACKFAMgrxIgCGy8FCQPCZwAHCwkIBwMCAQYVCAIJCgsEFgIDAQIe
AQIXgAAKCRBA1Kuxvt/atJo6EAC/5C8uJs76W5f5V5XNAMzwBFiZuYpop3DRReCo
P68ZZylokAC9ShRZnI0ujpDJt1NS7T/G00BzmcpspkYYE531ALaXcHWmb9XV0Ajg
J8iboAVBLY0C7mhL/cbJ3v9Q1pXXjyTuhexkJCV8rdHVX/0H8WqTZp1EaRuZ7p8q
PMxddg4ClwstYuH30/dmNd1Gqfb4Fqy8MnV1yGSXRs5Jf+sD1N2U04mbpyk/mr1c
f/jFmx86IkCWJVvdXWCVTe2AFy3NHCdLtdnEvFhokC0Qd9wibUWX0j9vq4cVRZT
qamnpAQa0lH3lX0wrjqo8b1AIPoRWSfMtCYvh6kA8MAJv4cAznzXILSLt0E0mzaU
qp5qoy37wNIjeztX6c/q4wss05qTlJhnNu4s3nh5VHultooaYpmDxp+ala5TWeuM
KZDI4KdAGF4z0Raif+N53nd0YIiXkY0goUbsPCnVrCwoK9PjjyoJncq7c14wN150
IQUZEjyYAQDGZqs5XSfY4zW2cCXatrfozKF7R1kSU14DfJwPUyksoNAQEQezfXyq
kr0gfIWK1r2nMdqS7WgSx/ypS5kdyrHuPZdaYfEVtuezpoT2lQQx0SZqq1p5hI4R
nqmPte53WXJhbC0tgTIIJWn+Uy/d5Q/aSIFD6o8gNLS1BDs1j1ku0XKu1sFCHUcZG
aerdsIkCHAQQAQkABgUCaCvFeAAKCRBC/V96c62Iwt3/D/9g0LzWtz62lqJRCsri
wcA/yz88ayKb/GUv3FCT5Nd9JZt8y1tW+AE3SPTdcpfZmt5UN2sRzlj061mpKJzp
eBvYQ9og/34ZrRQqeg8bz02u34LKY1lgD0xY0bWtB7TGIxIZZYqZECOPR0Dp6ZzB
abzkRSsJkEk0vbZzJhfWFYs98qfp/G0suFSBE7908Am33DB2jQ/Sollh1VmNE6Sv
EOgR6+2yEkS2D0+msJMa/V82v9gBTPnxS1NV1d8Dduvt9rbM3LoxiNXUgx/s52yY
U6H3bwUCq3UY6uRe1UWo5QnMFcDwfg43+q5rmjB4xQyX/BaQyF5K0hZyG+42/pH1
EMw18qN617FTxo3hVQUi/cBah1hQ8EVYsGnHDVxLCisbq5iZvp7+XtmMy1Q417gT
EQRo8feJh31e1GLWlccVR2pZgIm1PQ69dzzseHnnKkGhifik0bDGo5/IH2EgI1KFn
SG399RMU/qRz0PLVP3i+zSJmhMqG8cnZaUwE5V4P21vQSc1hhd2Hv/C4SVKNqA2i
+oZbHj2vAkuzTTL075AoANebEjPGqwsKZi5mWUE5Pa931JeiXxWZ1EB7rkgQ1PAB
fsDBhYLt4MxCWAhifLMA6uQ4BhXu2RuX0qNfSbqa8jVF6DB6cD8eAHGpPKfJ0130
LtZnq+n4SfeNbZjD2FQWZR4CrA==
=1Hfs
```

```
-----END PGP PUBLIC KEY BLOCK-----
```

4. 將簽署者公有金鑰匯入至 keyring。

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

gpg: key 4094ABB1BEDFDAB4: 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
```

請記下輸出的索引鍵值。例如 **4094ABB1BEDFDAB4**。

5. 使用金鑰值來取得和驗證簽署者公有金鑰指紋。

```
$ gpg --fingerprint 4094ABB1BEDFDAB4

pub   rsa4096 2025-05-19 [SCEA] [expires: 2027-05-19]
      EF46 3E86 CA31 933B B688 CC1A 4094 ABB1 BEDF DAB4
uid           [ unknown] AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>
```

指紋應符合下列項目：

```
EF46 3E86 CA31 933B B688 CC1A 4094 ABB1 BEDF DAB4
```

如果指紋字串不相符，請勿使用 AWS SAM CLI 安裝程式。在 [aws-sam-cli GitHub 儲存庫](#) 中 [建立問題](#)，向 AWS SAM 團隊呈報。

6. 驗證簽署者公有金鑰的簽章：

```
$ gpg --check-sigs 4094ABB1BEDFDAB4

pub   rsa4096 2025-05-19 [SCEA] [expires: 2027-05-19]
      EF463E86CA31933BB688CC1A4094ABB1BEDFDAB4
uid           [ unknown] AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>
sig!3        4094ABB1BEDFDAB4 2025-05-19 [self-signature]
sig!         42FD5F7A73AD885A 2025-05-19 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，將主要和簽署者公有金鑰匯入至 keyring。

您應該會看到列出的主要公有金鑰和簽署者公有金鑰的金鑰值。

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

驗證套件安裝程式的 AWS SAM CLI 完整性

1. 取得 AWS SAM CLI 套件簽章檔案 – 使用下列命令下載套件安裝程式的 AWS SAM CLI 簽章檔案：

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

2. 驗證簽章檔案 – 將下載的 .sig 和 .zip 檔案做為參數傳遞至 gpg 命令。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 Mon 19 May 2025 01:21:57 AM UTC using RSA key ID
4094ABB1BEDFDAB4
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: EF46 3E86 CA31 933B B688 CC1A 4094 ABB1 BEDF DAB4
```

- 您可以忽略 WARNING: This key is not certified with a trusted signature! 訊息。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。如需詳細資訊，請參閱[信任 Web](#)。
- 如果輸出包含片語 BAD signature，請檢查您是否正確執执行程序。如果您繼續取得此回應，請在 aws-sam-cli GitHub 儲存庫中[建立問題](#)，並避免使用下載的檔案，以向 AWS SAM 團隊呈報。

Good signature from "AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>"
訊息表示簽章已經過驗證，您可以繼續進行安裝。

macOS

GUI 和命令列安裝程式

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

使用 pkgutil 驗證

1. 執行下列命令，提供本機電腦上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，找到適用於 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: 2026-01-28 07:39:16 +0000
  Certificate Chain:
    1. Developer ID Installer: AMZN Mobile LLC (94KV3E626L)
      Expires: 2030-09-26 00:18:06 +0000
      SHA256 Fingerprint:
        5C 45 BE 63 FD 52 10 07 2D 66 56 77 5C A9 FF 25 91 6D 3F 01 F7 0E
        9A 8A 05 F6 2D 62 B2 88 8D A9
      -----
    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：

```
5C 45 BE 63 FD 52 10 07 2D 66 56 77 5C A9 FF 25 91 6D 3F 01 F7 0E 9A 8A 05 F6 2D 62
B2 88 8D A9
```

如果指紋字串不相符，請勿使用 AWS SAM CLI 安裝程式。在 [aws-sam-cli GitHub 儲存庫](#) 中 [建立問題](#)，向 AWS SAM 團隊呈報。如果指紋字串確實相符，您可以使用套件安裝程式繼續使用。

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 Apple 軟體更新的真實性](#)。

Windows

AWS SAM CLI 安裝程式會封裝為 Windows 作業系統 MSI 的檔案。

驗證安裝程式的完整性

1. 在安裝程式上按一下滑鼠右鍵並開啟屬性視窗。
2. 選擇 數位簽章 索引標籤。
3. 從簽章清單中，選擇 Amazon Web Services, Inc.，然後選擇詳細資訊。
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. 選擇詳細資訊索引標籤，如果尚未選取，請在顯示下拉式清單中選擇全部。
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。如果值相符，請繼續進行安裝。如果沒有，請在 [aws-sam-cli GitHub 儲存庫](#) 中 [建立問題](#)，向 AWS SAM 團隊呈報。

```
6c 4e f0 59 67 db 0a 2c 69 87 17 31 ec 9e 51 5d 7c d6 e9 52
```

驗證雜湊值

Linux

x86_64 - 命令列安裝程式

使用下列命令產生雜湊值，驗證下載安裝程式檔案的完整性和真實性：

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

輸出應如下所示：

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

在的[AWS SAM CLI版本備註](#)中，將 64 個字元的 SHA-256 雜湊值與所需 AWS SAM CLI版本的雜湊值進行比較GitHub。

macOS

GUI 和命令列安裝程式

使用以下命令產生雜湊值，驗證下載安裝程式的完整性和真實性：

```
$ 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
```

將您的 64 個字元 SHA-256 雜湊值與[AWS SAM CLI版本備註](#)GitHub儲存庫中的對應值進行比較。

教學課程：使用 部署 Hello World 應用程式 AWS SAM

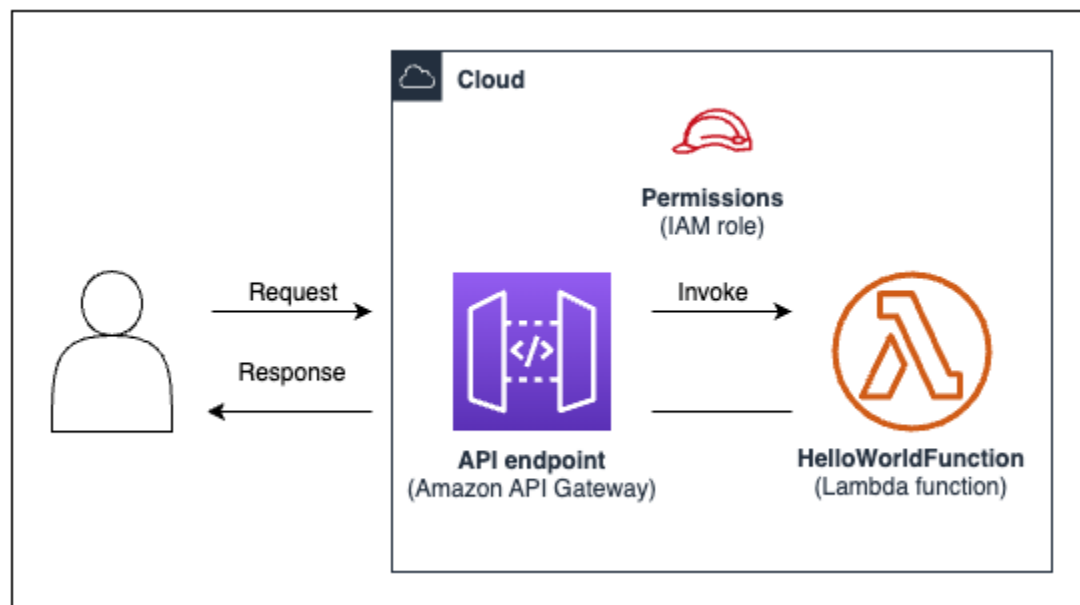
在本教學課程中，您可以使用 AWS Serverless Application Model 命令列界面 (AWS SAM CLI) 來完成下列操作：

- 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。
- 在開發主機上執行本機測試。
- 從 刪除範例應用程式 AWS 雲端。

範例 Hello World 應用程式實作基本 API 後端。它包含下列資源：

- Amazon API Gateway – 用來叫用函數的 API 端點。
- AWS Lambda – 處理 HTTP API GET 請求並傳回hello world訊息的函數。
- AWS Identity and Access Management (IAM) 角色 – 為服務佈建安全互動的許可。

下圖顯示此應用程式的組件：



主題

- [先決條件](#)
- [步驟 1：初始化範例 Hello World 應用程式](#)
- [步驟 2：建置您的應用程式](#)
- [步驟 3：將您的應用程式部署到 AWS 雲端](#)
- [步驟 4：執行您的應用程式](#)
- [步驟 5：與 中的 函數互動 AWS 雲端](#)
- [步驟 6：修改您的應用程式並將其同步至 AWS 雲端](#)
- [步驟 7：（選用）在本機測試您的應用程式](#)
- [步驟 8：從 刪除您的應用程式 AWS 雲端](#)
- [疑難排解](#)
- [進一步了解](#)

先決條件

確認您已完成下列操作：

- [AWS SAM 先決條件](#)
- [安裝 AWS SAM CLI](#)

Note

如果您偏好使用 Visual Studio Code 進行無伺服器開發，請參閱 AWS Toolkit for VS Code 使用者指南中的[建立無伺服器應用程式](#)，以取得使用 VS Code 界面step-by-step說明。

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 SAM CLI本機電腦上建立範例 Hello World 應用程式專案。

初始化範例 Hello World 應用程式

1. 在命令列中，從您選擇的開始目錄執行下列項目：

```
$ sam init
```

Note

此命令會初始化無伺服器應用程式並建立專案目錄。目錄包含數個檔案和資料夾，包括 `template.yaml` 檔案，也就是您的 AWS SAM 範本。

安裝的 Python 版本必須符合中指定的Runtime屬性`template.yaml`。對於生產工作負載，我們建議使用 AWS 環境中完全支援的 Python 3.12 或更早版本。如果 Python 版本不相符，您將會遇到建置錯誤。

若要解決版本相容性問題：

- 在初始化期間指定相容的[執行時間](#)：`sam init --runtime python3.9`
- 初始化`template.yaml`後在 中修改 Runtime 屬性

2. AWS SAM CLI 將引導您初始化新的應用程式。設定下列項目：

1. 選取 AWS Quick Start 範本以選擇啟動範本。
2. 選擇 Hello World 範例範本並下載。
3. 使用Python執行期和zip套件類型。
4. 在本教學課程中，選擇退出 AWS X-Ray 追蹤。若要進一步了解，請參閱[什麼是 AWS X-Ray？](#)《AWS X-Ray 開發人員指南》中的。

5. 在本教學課程中，選擇不使用 Amazon CloudWatch Application Insights 進行監控。若要進一步了解，請參閱《[Amazon CloudWatch 使用者指南](#)》中的 [Amazon CloudWatch Application Insights](#)。Amazon CloudWatch
6. 在此教學課程中，選擇不在您的 Lambda 函數上設定 JSON 格式的結構化記錄。
7. 將您的應用程式命名為 sam-app。

若要使用 AWS SAM CLI 互動式流程：

- 括號 ([]) 表示預設值。將您的答案保留空白以選取預設值。
- 輸入 **y** 表示是，輸入 **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 - Data processing
  3 - Hello World Example with Powertools for AWS Lambda
  4 - Multi-step workflow
  5 - Scheduled task
  6 - Standalone function
  7 - Serverless API
  8 - Infrastructure event management
  9 - Lambda Response Streaming
 10 - GraphQLApi Hello World Example
 11 - Full Stack
 12 - Lambda EFS example
 13 - Serverless Connector Hello World Example
 14 - Multi-step workflow with Connectors
 15 - DynamoDB Example
 16 - 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 AWS SAM CLI 下載您的啟動範本，並在本機電腦上建立應用程式專案目錄結構。以下是輸出的範例 AWS SAM CLI：

```
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。以下是 AWS SAM CLI 建立的範例：

 Note

如果tree命令未自動安裝，請執行此命令：`brew install tree`

```
$ 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
```

要強調的一些重要檔案：

- `hello_world/app.py` – 包含您的 Lambda 函數程式碼。
- `hello_world/requirements.txt` – 包含 Lambda 函數所需的任何Python相依性。
- `samconfig.toml` – 應用程式的組態檔案，存放使用的預設參數 AWS SAM CLI。
- `template.yaml` – 包含應用程式基礎設施程式碼的 AWS SAM 範本。

您現在已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，您會使用 AWS SAM CLI來建置應用程式並準備部署。當您建置時，AWS SAM CLI會建立 `.aws-sam` 目錄，並在該處組織您的函數相依性、專案程式碼和專案檔案。

建置您的應用程式

- 在命令列中，從 `sam-app` 專案目錄執行下列動作：

```
$ sam build
```

Note

如果您的 Python 本機電腦上沒有 `Docker`，請改用 `sam build --use-container` 命令。AWS SAM CLI 會建立 Docker 容器，其中包含函數的執行時間和相依性。此命令 `Docker` 在您的本機電腦上需要。若要安裝 `Docker`，請參閱 [安裝 Docker](#)。

以下是輸出的範例 AWS SAM CLI：

```
$ 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 CLI 所 AWS SAM 建立 .aws-sam 目錄的縮短範例：

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

要強調的一些重要檔案：

- build/HelloWorldFunction – 包含您的 Lambda 函數程式碼和相依性。會為應用程式中的每個函數 AWS SAM CLI 建立目錄。
- build/template.yaml – 包含在部署 CloudFormation 時由 參考的 AWS SAM 範本副本。
- build.toml – 組態檔案，用於存放建置和部署應用程式時 AWS SAM CLI 參考的預設參數值。

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 AWS 登入資料組態。如需詳細資訊，請參閱 [AWS SAM 先決條件](#) 中的 [步驟 2：使用 AWS CLI 設定 AWS 登入資料](#)。

在此步驟中 AWS SAM CLI，您可以使用 將應用程式部署到 AWS 雲端。AWS SAM CLI 將執行下列動作：

- 引導您設定應用程式設定以進行部署。
- 將您的應用程式檔案上傳至 Amazon Simple Storage Service (Amazon S3)。

- 將您的 AWS SAM 範本轉換為 CloudFormation 範本。接著會將您的範本上傳至 CloudFormation 服務，以佈建您的 AWS 資源。

部署您的 應用程式

1. 在命令列中，從sam-app專案目錄執行下列動作：

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

2. 遵循 AWS SAM CLI互動式流程來設定您的應用程式設定。設定下列項目：
 1. CloudFormation 堆疊名稱 – 堆疊是您可以單一單位管理的一組 AWS 資源。若要進一步了解，請參閱AWS CloudFormation 《使用者指南》中的[使用堆疊](#)。
 2. AWS 區域 要部署 CloudFormation 堆疊的。如需詳細資訊，請參閱AWS CloudFormation 《使用者指南》中的[CloudFormation 端點](#)。
 3. 在本教學課程中，選擇在部署之前不確認變更。
 4. 允許建立 IAM 角色 – 這可讓 AWS SAM 建立 API Gateway 資源和 Lambda 函數資源互動所需的 IAM 角色。
 5. 在本教學課程中，選擇不停用轉返。
 6. 允許 HelloWorldFunction 未定義授權 – 會顯示此訊息，因為您的 API Gateway 端點已設定為可公開存取，無需授權。由於這是 Hello World 應用程式的預期組態，請允許 AWS SAM CLI 繼續。如需設定授權的詳細資訊，請參閱 [使用 AWS SAM 範本控制 API 存取](#)。
 7. 將引數儲存至組態檔案 – 這會使用您的部署偏好設定來更新應用程式的 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 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. 會執行下列動作來 AWS SAM CLI部署您的應用程式：

- 會建立 Amazon S3 AWS SAM CLI儲存貯體並上傳您的 .aws-sam目錄。
- 會將您的 AWS SAM 範本轉換為 AWS SAM CLI，CloudFormation 並將其上傳至 CloudFormation 服務。
- CloudFormation 會佈建您的 資源。

在部署期間，AWS SAM CLI會顯示您的進度。以下是輸出範例：

```
Looking for resources needed for deployment:
```

```
Managed S3 bucket: aws-sam-cli-managed-default-samclisam-s3-demo-
bucket-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-samclisam-s3-demo-
bucket-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，您將傳送 GET 請求到您的 API 端點，並查看您的 Lambda 函數輸出。

取得您的 API 端點值

1. 從上一個步驟中 AWS SAM CLI顯示的資訊中，找到 Outputs區段。在本節中，找到您的HelloWorldApi資源以尋找您的 HTTP 端點值。以下是輸出範例：

```
-----
Outputs
-----
```

```

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

```

2. 或者，您可以使用 `sam list endpoints --output json` 命令來取得此資訊。以下是輸出範例：

```

$ sam list endpoints --output json
2023-03-15 12:39:19 Loading policies from IAM...
2023-03-15 12:39:25 Finished loading policies from IAM.
[
  {
    "LogicalResourceId": "HelloWorldFunction",
    "PhysicalResourceId": "sam-app-HelloWorldFunction-yQDNe17r9maD",
    "CloudEndpoint": "-",

```

```

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

```

叫用您的 函數

- 使用您的瀏覽器或命令列，將 GET 請求傳送至您的 API 端點。以下是使用 curl 命令的範例：

```

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

```

步驟 5：與 中的 函數互動 AWS 雲端

在此步驟中，您會使用在 AWS SAM CLI 中叫用 Lambda 函數 AWS 雲端。

在雲端中叫用 Lambda 事件函數

1. 請記下您函式LogicalResourceId上一個步驟的。它應該為 HelloWorldFunction。
2. 在命令列中，從sam-app專案目錄執行下列動作：

```

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

```

3. 會在雲端 AWS SAM CLI中叫用您的函數，並傳回回應。以下是輸出範例：

```

$ 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\"}}%
```

步驟 6：修改您的應用程式並將其同步至 AWS 雲端

在此步驟中，您可以使用 AWS SAM CLI `sam sync --watch` 命令將本機變更同步至 AWS 雲端。

使用 `sam` 同步

1. 在命令列中，從 `sam-app` 專案目錄執行下列動作：

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

2. 會 AWS SAM CLI 提示您確認是否同步開發堆疊。由於 `sam sync --watch` 命令會自動 AWS 雲端 即時將本機變更部署到，因此我們建議僅用於開發環境。

會在開始監控本機變更之前 AWS SAM CLI 執行初始部署。以下是輸出範例：

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

Confirm that you are synchronizing a development stack.

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

Build Succeeded

Successfully packaged artifacts and wrote output template to file /var/
folders/45/5ct135bx3fn2551_pt15g6_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-samclisam-s3-demo-
bucket-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		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-15GL0UR9LMT1W

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.

```

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

修改和同步您的應用程式

1. 在您所選的 IDE 中，開啟 `sam-app/hello_world/app.py` 檔案。
2. 變更 `message` 並儲存您的 檔案。以下是範例：

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

3. AWS SAM CLI 會偵測您的變更，並將您的應用程式同步至 AWS 雲端。以下是輸出範例：

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

4. 若要驗證您的變更，請再次傳送 GET 請求到您的 API 端點。

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

步驟 7：（選用）在本機測試您的應用程式

Note

此為選擇性步驟。

Important

此步驟 Docker 在本機電腦上需要。您必須 Docker 安裝 並設定為使用 AWS SAM CLI 進行本機測試。如需詳細資訊，請參閱 [安裝 Docker](#)。

在此步驟中，您可以使用 AWS SAM CLI `sam local` 命令在本機測試您的應用程式。為了達成此目的，AWS SAM CLI 會使用 建立本機環境 Docker。此本機環境模擬 Lambda 函數的雲端型執行環境。

您可執行下列項目：

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

在本機叫用 Lambda 事件函數

1. 在命令列中，從 `sam-app` 專案目錄執行下列動作：

```
$ sam local invoke
```

2. AWS SAM CLI 會建立本機 Docker 容器並叫用您的 函數。以下是輸出範例：

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

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

在本機託管您的 API

1. 在命令列中，從 `sam-app` 專案目錄執行下列動作：

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

2. 會為您的 Lambda AWS SAM CLI 函數建立本機 Docker 容器，並建立本機 HTTP 伺服器來模擬您的 API 端點。以下是輸出範例：

```
$ 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/.../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 AWS 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. 使用您的瀏覽器或命令列，將 GET 請求傳送至本機 API 端點。以下是使用 curl 命令的範例：

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

步驟 8：從 刪除您的應用程式 AWS 雲端

在此步驟中，您可以使用 AWS SAM CLI 的 `sam delete` 命令從 刪除您的應用程式 AWS 雲端。

從 刪除您的應用程式 AWS 雲端

1. 在命令列中，從 `sam-app` 專案目錄執行下列動作：

```
$ sam delete
```

2. AWS SAM CLI 會要求您確認。然後，它會刪除應用程式的 Amazon S3 儲存貯體和 CloudFormation 堆疊。以下是輸出範例：

```
$ sam delete
Are you sure you want to delete the stack sam-app in the region us-west-2 ? [y/
N]: y
```

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

Deleted successfully
```

疑難排解

若要對 進行故障診斷 AWS SAM CLI，請參閱 [AWS SAM CLI 故障診斷](#)。

進一步了解

若要繼續了解 AWS SAM，請參閱下列資源：

- [完整 AWS SAM 研討會](#) – 旨在教導您許多 AWS SAM 提供的主要功能的研討會。
- [使用 SAM 的工作階段](#) – 由我們的 AWS Serverless Developer Advocate 團隊在使用 時建立的影片系列 AWS SAM。
- [無伺服器土地](#) – 將最新資訊、部落格、影片、程式碼和學習資源整合在一起的網站 AWS。

將 Lambda 事件函數轉換為 AWS SAM 應用程式

如果您在 Lambda 主控台中有要以基礎設施形式在 VS 程式碼 (IaC) 中管理的函數，您可以將其轉移至 VS 程式碼，然後將其轉換為 AWS SAM 範本。將函數轉換為 AWS SAM 範本後，您就可以控制基礎設施的版本控制、自動化部署、遠端偵錯函數，以及在整個開發生命週期中維持一致的環境。透過本機和遠端偵錯功能，您也可以透過逐步完成程式碼、檢查變數，以及在本機和 AWS 雲端中設定中斷點，更有效地疑難排解問題。

如需從主控台移動函數和設定本機開發的指示，請參閱《AWS Lambda 開發人員指南》中的[使用 VS 程式碼在本機開發 Lambda 函數](#)。請依照這些指示，將 Lambda 事件函數從主控台移至 VS 程式碼，然後將其轉換為 AWS SAM 範本。執行此操作後，您可以：

- 在 AWS 雲端遠端偵錯您的 函數。如需詳細資訊，請參閱《AWS Lambda 開發人員指南》中的[使用 VS 程式碼遠端偵錯 Lambda 函數](#)。

- 將關聯的資源和許可作為基礎設施管理為程式碼。如需詳細資訊，請參閱《AWS Lambda 開發人員指南》中的[使用 Lambda 搭配基礎設施做為程式碼 \(IaC\)](#)
- 使用 部署和更新您的應用程式 AWS SAM CLI。如需詳細資訊，請參閱[AWS SAM CLI](#)。
- 使用 Infrastructure Composer 視覺化設計無伺服器架構。如需詳細資訊，請參閱 [Infrastructure Composer 開發人員指南](#)。

進一步了解

若要繼續了解 AWS SAM，請參閱下列資源：

- [完整 AWS SAM 研討會](#) – 旨在教導您許多 AWS SAM 提供的主要功能的研討會。
- [使用 SAM 的工作階段](#) – 由我們的 AWS Serverless Developer Advocate 團隊在使用 時建立的影片系列 AWS SAM。
- [無伺服器土地](#) – 將最新資訊、部落格、影片、程式碼和學習資源整合在一起的網站 AWS。

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 SAM CLI和 AWS SAM 範本和 AWS SAM 專案（即您的應用程式專案目錄）。您可以使用這些工具來：

1. [開發您的應用程式](#)（這包括初始化您的應用程式、定義您的資源，以及建置您的應用程式）。
2. [測試您的應用程式](#)。
3. [偵錯您的應用程式](#)。
4. [部署您的應用程式和資源](#)。
5. [監控您的應用程式](#)。

AWS SAM 在您執行 `sam init`命令並完成其後續工作流程後，會建立您的 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，您會撰寫程式碼來表達資源、事件來源映射，以及其他定義無伺服器應用程式的屬性。

AWS SAM CLI 包含您在 AWS SAM 專案上使用的命令儲存庫。更具體地說，AWS SAM CLI是您用來建置、轉換、部署、除錯、封裝、初始化和同步 AWS SAM 專案的。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。

如需這些工具的詳細資訊，請參閱下列章節：

[AWS SAM CLI](#) - 了解 AWS SAM CLI以及如何使用它來建置和執行無伺服器應用程式。

[AWS SAM 專案和 AWS SAM 範本](#) - 了解 AWS SAM 專案結構以及如何使用 AWS SAM 範本來定義無伺服器應用程式。

AWS SAM CLI

AWS Serverless Application Model 命令列界面 (AWS SAM CLI) 是一種命令列工具，用於本機開發和測試無伺服器應用程式。AWS SAM CLI 可讓您在本機建置、轉換、部署、除錯、封裝、初始化和同步無伺服器應用程式，然後再部署到雲端。

AWS SAM CLI 適用於使用不同架構和基礎設施做為程式碼 (IaC) 工具定義的無伺服器應用程式，支援程度不盡相同：

- AWS SAM 範本 – 透過完整的功能集提供原生支援，包括本機測試、偵錯、封裝和部署功能。
- AWS CDK 應用程式 – 在您使用 `cdk synth` 命令將 AWS CDK 應用程式合成至 CloudFormation 範本後，支援 Lambda 事件函數的本機測試。
- CloudFormation 範本 – 提供直接相容性，因為 AWS SAM 延伸 CloudFormation 支援標準 CloudFormation 範本中定義的無伺服器資源。
- Terraform 應用程式 – 為 Lambda 函數的建置和本機測試提供有限的支援。要求您產生代表 Terraform 定義 Lambda 函數的 AWS SAM 範本成品。

如需最全面的功能支援和簡化的開發人員體驗，建議使用原生 AWS SAM 範本。

主題

- [如何 AWS SAM CLI 記錄命令](#)
- [設定 AWS SAM CLI](#)
- [AWS SAM CLI 核心命令](#)
- [使用 進行本機測試 AWS SAM CLI](#)

如何 AWS SAM CLI 記錄命令

AWS SAM CLI 命令會以下列格式記錄：

- 提示 – 預設會記錄 Linux 提示，並顯示為 (\$)。對於 Windows 特定命令，會使用 (>) 做為提示。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。
- 可取代的文字 – 變數文字，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命令或命令中，鍵盤輸入也可以顯示為可取代的文字。例如，**ENTER**。

- 輸出 – 傳回做為命令回應的輸出格式為 `computer output`。

下列 `sam deploy` 命令和輸出為範例：

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

Configuring SAM deploy
=====

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

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

1. `sam deploy --guided --template template.yaml` 是您在命令列輸入的命令。
2. **`sam deploy --guided --template`** 應照原樣提供。
3. *`template.yaml`* 可以取代為您的特定檔案名稱。
4. 輸出從 開始 `Configuring SAM deploy`。
5. 在輸出中，*`ENTER`* 和 *`y`* 表示您提供的可取代值。

設定 AWS SAM CLI

的好處之一 AWS SAM 是，它透過移除重複性任務來最佳化開發人員的時間。AWS SAM CLI 包含名為 `samconfig` 的組態檔案。根據預設，AWS SAM CLI 不需要對 進行組態，但您可以更新組態檔

案，以允許參考組態檔案中的自訂參數 AWS SAM，以使用較少的參數執行命令。下表中的範例示範如何最佳化命令：

原始的	使用最佳化 samconfig
<code>sam build --cached --parallel --use-containers</code>	<code>sam build</code>
<code>sam local invoke --env-vars locals.json</code>	<code>sam local invoke</code>
<code>sam local start-api --env-vars locals.json --warm-containers EAGER</code>	<code>sam local start-api</code>

AWS SAM CLI 提供一組命令，協助開發人員建立、開發和部署無伺服器應用程式。這些命令都可以根據應用程式和開發人員的偏好設定，使用選用的旗標來設定。如需詳細資訊，請參閱 [AWS SAM CLI GitHub 中的內容](#)

本節中的主題說明如何建立 [AWS SAM CLI 組態檔案](#)並自訂其預設設定，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](#)
- [設定登入資料和基本設定](#)

如何建立您的組態檔案 (samconfig 檔案)

組態檔案 AWS SAM CLI (檔案名稱 `samconfig`) 是文字檔案，通常使用 TOML 結構，但也可以在 YAML 中。使用 AWS Quick Start 範本時，會在您執行 `sam init`命令時建立此檔案。您可以在使用 `sam deploy -l-guided`命令部署應用程式時更新此檔案。

部署完成後，default如果您使用預設值，`samconfig` 檔案會包含名為 `default` 的設定檔。當您重新執行 `deploy`命令時，會從此設定檔 AWS SAM 套用儲存的組態設定。

`samconfig` 檔案的優點是，除了部署命令之外，AWS SAM 還存放任何其他可用命令的組態設定。除了在新部署時建立的這些值之外，您還可以在 `samconfig` 檔案中設定許多屬性，以使用簡化開發人員工作流程的其他層面 AWS SAM CLI。

設定專案設定

您可以在組態檔案中指定專案特定的設定，例如 AWS SAM CLI 命令參數值，以搭配使用 AWS SAM CLI。如需此組態檔案的詳細資訊，請參閱 [AWS SAM CLI 組態檔案](#)。

使用組態檔案

組態檔案由環境、命令和參數值構成。如需詳細資訊，請參閱 [組態檔案基本概念](#)。

設定新的環境

1. 在組態檔案中指定您的新環境。

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` 選項來指定要使用的環境。

以下是範例：

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

設定參數值

1. 指定 AWS SAM CLI您要為其設定參數值的命令。若要設定所有 AWS SAM CLI命令的參數值，請使用 `global` 識別符。

以下是指定 `default` 環境 `sam deploy` 命令參數值的範例：

TOML

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

YAML

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

以下是為 `default` 環境中所有 AWS SAM CLI 命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. 您也可以指定參數值，並透過互動式流程修改您的組態檔案 AWS SAM CLI。

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[建立和修改組態檔案](#)。

範例

基本TOML範例

以下是samconfig.toml組態檔案的範例：

```
...
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samconfig.yaml組態檔案的範例：

```
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

```

設定登入資料和基本設定

使用 AWS Command Line Interface (AWS CLI) 來設定基本設定，例如 AWS 登入資料、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 搭配使用 AWS SAM CLI。若要進一步了解，請參閱 AWS Command Line Interface 《使用者指南》中的下列內容：

- [組態基本概念](#)
- [組態和登入資料檔案設定](#)
- [的具名設定檔 AWS CLI](#)
- [使用已啟用 IAM Identity Center 的具名設定檔](#)

如需快速設定說明，請參閱 [步驟 2：使用 AWS CLI 設定 AWS 登入資料](#)。

AWS SAM CLI 核心命令

AWS SAM CLI 有一些基本命令，您可用來建立、建置、測試、部署和同步無伺服器應用程式。下表列出這些命令，並提供每個命令的詳細資訊連結。

如需命令的完整清單 AWS SAM CLI，請參閱 [AWS SAM CLI 命令參考](#)。

命令	它的功能	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至 AWS 雲端。	<ul style="list-style-type: none"> • 使用 建置 簡介 AWS SAM • sam build
sam deploy	使用 將應用程式部署至 AWS 雲端 CloudFormation。	<ul style="list-style-type: none"> • 使用 部署簡介 AWS SAM • sam deploy

命令	它的功能	相關主題
sam init	提供初始化和建立新的無伺服器應用程式的選項。	<ul style="list-style-type: none"> • 在 中建立您的應用程式 AWS SAM • sam init
sam local	提供子命令以在本機測試無伺服器應用程式。	<ul style="list-style-type: none"> • 使用 sam local 命令進行測試的簡介 • sam local generate-event • sam local invoke • sam local start-api • sam local start-lambda
sam remote invoke	提供與 AWS 雲端中支援 AWS 的資源互動的方法。	<ul style="list-style-type: none"> • 在雲端使用 進行測試的簡介 sam remote invoke • sam remote invoke
sam remote test-event	提供存取和管理 AWS Lambda 事件函數可共用測試事件的方法。	<ul style="list-style-type: none"> • 使用 進行雲端測試的簡介 sam remote test-event • sam remote test-event
sam sync	提供快速將本機應用程式變更同步至 AWS 雲端的選項。	<ul style="list-style-type: none"> • 使用 sam sync 同步至 的簡介 AWS 雲端 • sam sync

使用 進行本機測試 AWS SAM CLI

可讓您在本機測試不同基礎設施的無伺服器應用程式，即程式碼 AWS SAM CLI(IaC) 工具。本指南說明如何使用 AWS SAM CLI 搭配各種 IaC 架構進行本機測試。

使用 AWS SAM CLI 進行本機測試的主要優點包括：

- 快速開發 - 在不部署至 的情況下測試程式碼變更 AWS
- 成本效益 - 開發和測試而不會產生 AWS 費用
- 離線功能 - 在沒有網際網路連線的情況下處理應用程式
- 簡化偵錯 - 使用偵錯工具在本機逐步完成 Lambda 事件函數程式碼

- 實際測試 - 使用 AWS 服務的本機模擬測試您的應用程式

在本機測試 AWS SAM 應用程式

如需有關測試使用 AWS SAM 範本定義的應用程式的資訊，請參閱本指南中的[測試和偵錯無伺服器應用程式](#)。

在本機測試 CloudFormation 範本

若要搭配使用 AWS SAM CLI 的本機測試功能 CloudFormation，請將 AWS SAM 轉換新增至您的 CloudFormation 範本。如需詳細資訊，請參閱 [AWS SAM 範本結構](#)：

- 將 AWS SAM 轉換新增至 CloudFormation 範本
- 了解範本相容性
- 探索無伺服器資源語法

在本機測試 AWS CDK 應用程式

您可以在使用 AWS cdk synth 命令將 CDK 應用程式合成至 CloudFormation 範本之後，使用 AWS SAM CLI 在本機測試 AWS CDK 應用程式。

如需詳細資訊，請參閱《[AWS CDK 開發人員指南](#)》中的使用 [本機測試和建置 CDK 應用程式 AWS SAM CLI](#)。AWS 其中包含：

- [本機測試入門](#)
- [使用 本機測試 AWS CDK 應用程式 AWS SAM](#)

在本機測試 Terraform 應用程式

支援 Terraform AWS SAM CLI 專案和 Terraform Cloud。您可以使用它來執行本機偵錯和測試：Lambda 函數和 layer，以及 Amazon API Gateway HTTP 和 REST APIs。

若要設定您的環境並了解所有可用的功能，請參閱本指南中的 [Terraform Support](#)。其中包含：

- [的 Terraform 支援入門 AWS SAM CLI](#)
- [使用 AWS SAM CLI 搭配 Terraform 進行本機偵錯和測試](#)

AWS SAM 範本

在您執行 `sam init` 命令並完成其後續工作流程後，會 AWS SAM 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。雖然您的 AWS SAM 專案包含一組檔案和資料夾，但您主要使用的檔案是您的 AWS SAM 範本（名為 `template.yaml`）。在此範本中，您會撰寫程式碼來表達資源、事件來源映射，以及定義無伺服器應用程式的其他屬性。

Note

AWS SAM 範本的關鍵元素是 AWS SAM 範本規格。此規格提供短期語法，相較於 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、事件來源映射、許可、APIs 和其他屬性。

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、內部函數和 API Gateway 延伸的詳細資訊。

AWS SAM 範本是 CloudFormation 範本的延伸，具有唯一的語法類型，使用比較少行程式碼的速記語法 CloudFormation。這可在建置無伺服器應用程式時加速您的開發。如需詳細資訊，請參閱 [AWS SAM 資源和屬性](#)。如需 CloudFormation 範本的完整參考，請參閱 AWS CloudFormation 《使用者指南》中的 [CloudFormation 範本參考](#)。

開發時，您通常會發現將應用程式程式碼分成不同的檔案，以便更好地組織和管理應用程式。其中一個基本範例是為您的 AWS Lambda 事件函數程式碼使用單獨的檔案，而不是在您的 AWS SAM 範本中擁有此程式碼。做法是在專案的子目錄中組織 Lambda 函數程式碼，並在您的 AWS Serverless Application Model (AWS SAM) 範本中參考其本機路徑。

主題

- [AWS SAM 範本結構](#)
- [AWS SAM 資源和屬性](#)
- [為產生 CloudFormation 資源 AWS SAM](#)
- [支援的資源屬性 AWS SAM](#)
- [的 API Gateway 擴充功能 AWS SAM](#)
- [的內部函數 AWS SAM](#)
- [CloudFormation 語言延伸支援](#)

AWS SAM 範本結構

AWS SAM 範本檔案嚴格遵循 CloudFormation 範本檔案的格式，如AWS CloudFormation 《使用者指南》中的[範本結構](#)所述。AWS SAM 範本檔案和 CloudFormation 範本檔案的主要差異如下：

- 轉換宣告。範本檔案Transform: AWS::Serverless-2016-10-31需要 AWS SAM 宣告。此宣告會將 CloudFormation 範本檔案識別為 AWS SAM 範本檔案。如需轉換的詳細資訊，請參閱AWS CloudFormation 《使用者指南》中的[轉換](#)。
- 全域區段。Globals 區段對 是唯一的 AWS SAM。它定義了所有無伺服器函數和 APIs屬性。所有 AWS::Serverless::Function、AWS::Serverless::Api、AWS::Serverless::CapacityProvider、AWS::Serverless::HttpApi和 AWS::Serverless::StateMachine 資源都會繼承 Globals區段中定義的屬性。如需本節的詳細資訊，請參閱 [AWS SAM 範本的全域區段](#)。
- 資源區段。在 AWS SAM 範本中，Resources區段可以包含 CloudFormation 資源和資源的組合 AWS SAM。如需 CloudFormation 資源的詳細資訊，請參閱AWS CloudFormation 《使用者指南》中的[AWS 資源和屬性類型參考](#)。如需 AWS SAM 資源的詳細資訊，請參閱 [AWS SAM 資源和屬性](#)。

AWS SAM 範本檔案的所有其他區段對應至相同名稱的 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 範本可以包含數個主要區段。只需要 Transform 和 Resources 區段。

您可以按任何順序包含範本區段。不過，如果使用語言延伸模組，您應該在無伺服器轉換 `AWS::LanguageExtensions` 之前新增（也就是在之前 `AWS::Serverless-2016-10-31`），如下列範例所示：

Transform:

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

當您建置範本時，使用下列清單中顯示的邏輯順序可能會有所幫助。這是因為一個區段中的值可能參照上一個區段的值。

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`。

其他轉換是選用的。如需轉換的詳細資訊，請參閱 AWS CloudFormation 《使用者指南》中的 [轉換](#)。

全域（選用）

所有無伺服器函數、APIs 屬性。所有

`AWS::Serverless::Function`、`AWS::Serverless::Api`、`AWS::Serverless::CapacityProvider`、`AWS::Serverless::HttpApi` 和 `AWS::Serverless::StateMachine` 資源都會繼承 `Globals` 區段中定義的屬性。

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

描述（選用）

說明範本的文字字串。

本節直接對應至 CloudFormation 範本的 Description 區段。

[Metadata \(選用\)](#)

提供範本其他資訊的物件。

本節直接對應至 CloudFormation 範本的 Metadata 區段。

[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。在 Parameters 區段中宣告的物件會導致 `sam deploy --guided` 命令向使用者顯示其他提示。

使用 `sam deploy` 命令的 `--parameter-overrides` 參數以及組態檔案中的項目傳入的值，優先於範本檔案中的項目 AWS SAM。如需 `sam deploy` 命令的詳細資訊，請參閱 [命令參考 sam deploy](#) 中的 AWS SAM CLI。如需組態檔案的詳細資訊，請參閱 [AWS SAM CLI 組態檔案](#)。

[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 [Fn::FindInMap](#) Resources 和 Outputs 區段中的內部 函數，將索引鍵與對應的值進行比對。

本節直接對應至 CloudFormation 範本的 Mappings 區段。

[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。

本節直接對應至 CloudFormation 範本的 Conditions 區段。

[Resources \(必要\)](#)

堆疊資源及其屬性，例如 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體或 Amazon Simple Storage Service (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 範本的 CloudFormation Resources 區段。在 AWS SAM 範本中，除了 AWS SAM 資源之外，本節還可以包含 CloudFormation 資源。

[Outputs \(選用\)](#)

每次檢視堆疊屬性時傳回的值。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。

本節直接對應至 CloudFormation 範本的 Outputs 區段。

後續步驟

若要下載和部署包含 AWS SAM 範本檔案的範例無伺服器應用程式，請參閱 [入門 AWS SAM](#) 並遵循中的指示[教學課程：使用 部署 Hello World 應用程式 AWS SAM](#)。

AWS SAM 範本的全域區段

有時，您在 AWS SAM 範本中宣告的資源具有常見的組態。例如，您可能有一個應用程式，其中包含多個具有相同 Runtime、Memory、VPCConfig、Environment 和 Cors 組態 `AWS::Serverless::Function` 的資源。您可以在 Globals 區段中宣告一次，並讓資源繼承這些資訊，而不是在每個資源中複製此資訊。

Globals 本節支援下列 AWS SAM 資源類型：

- `AWS::Serverless::Api`
- `AWS::Serverless::CapacityProvider`
- `AWS::Serverless::Function`
- `AWS::Serverless::HttpApi`
- `AWS::Serverless::SimpleTable`
- `AWS::Serverless::StateMachine`
- `AWS::Serverless::MicrovmImage`
- `AWS::Serverless::NetworkConnector`

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

```

在此範例中，HelloWorldFunction和 都會針對 ThumbnailFunction使用「nodejs12.x」Runtime、針對 使用「180」秒Timeout，以及針對 使用「index.handler」Handler。除了繼承的 TABLE_NAME 之外，還會HelloWorldFunction新增 MESSAGE 環境變數。會ThumbnailFunction繼承所有Globals屬性並新增 API 事件來源。

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:
    EndpointAccessMode:
    GatewayResponses:
    MethodSettings:
    MinimumCompressionSize:
    Name:
    OpenApiVersion:

```

PropagateTags:
SecurityPolicy:
TracingEnabled:
Variables:

CapacityProvider:
InstanceRequirements:
KmsKeyArn:
LoggingConfig:
ManagedResourceTags:
OperatorRole:
PropagateTags:
ScalingConfig:
Tags:
VpcConfig:

Function:
Architectures:
AssumeRolePolicyDocument:
AutoPublishAlias:
AutoPublishAliasAllProperties:
CapacityProviderConfig:
CodeSigningConfigArn:
CodeUri:
DeadLetterQueue:
DeploymentPreference:
Description:
DurableConfig:
Environment:
EphemeralStorage:
EventInvokeConfig:
FileSystemConfigs:
FunctionScalingConfig:
FunctionUrlConfig:
Handler:
KmsKeyArn:
Layers:
LoggingConfig:
MemorySize:
PermissionsBoundary:
PropagateTags:
ProvisionedConcurrencyConfig:
PublishToLatestPublished:
RecursiveLoop:

ReservedConcurrentExecutions:

RolePath:

Runtime:

RuntimeManagementConfig:

SnapStart:

SourceKMSKeyArn:

Tags:

TenancyConfig:

Timeout:

Tracing:

VersionDeletionPolicy:

VpcConfig:

HttpApi:

AccessLogSettings:

Auth:

PropagateTags:

StageVariables:

Tags:

SimpleTable:

SSESpecification:

StateMachine:

PropagateTags:

MicrovmImage:

BaseImageArn:

BaseImageVersion:

BuildRoleArn:

Hooks:

EnvironmentVariables:

EgressNetworkConnectors:

CpuConfigurations:

Resources:

AdditionalOsCapabilities:

Logging:

Tags:

PropagateTags:

NetworkConnector:

OperatorRole:

Tags:

PropagateTags:

Note

不支援任何未包含在上一個清單中的資源和屬性。不支援它們的一些原因包括：1) 它們開啟了潛在的安全問題，或 2) 它們使範本難以理解。

隱含 APIs

AWS SAM 當您在 Events 區段中宣告 APIs 時，會建立隱含 API。您可以使用 Globals 覆寫隱含 APIs 的所有屬性。

可覆寫的屬性

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映射，也可以覆寫全域宣告的變數。但是資源無法移除 Globals 區段中指定的屬性。

一般而言，Globals 區段會宣告所有資源共用的屬性。有些資源可以為全域宣告的屬性提供新值，但無法將其移除。如果某些資源使用屬性，但其他資源不使用，則不得在 Globals 區段中宣告它們。

下列各節說明覆寫對不同資料類型的運作方式。

取代基本資料類型

基本資料類型包括字串、數字、布林值等。

Resources 區段中指定的值會取代 Globals 區段中的值。

範例：

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

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

Runtime 的 MyFunction 設定為 python3.9。

映射已合併

映射也稱為字典或索引鍵/值對的集合。

Resources 區段中的映射項目會與全域映射項目合併。如果有重複項目，Resource區段項目會覆蓋Globals區段項目。

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MyFunction設定為下列：

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

清單是附加項目

清單也稱為陣列。

Globals 區段中的清單項目會附加至Resources區段中的清單。

範例：

```
Globals:
  Function:
```

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

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

MyFunction適用於 SecurityGroupIds 的 VpcConfig設定為下列項目：

```
[ "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 AWS SAM 速記語法定義這些資源和屬性。AWS SAM 也支援 CloudFormation 資源和屬性類型。如需所有 AWS 資源和屬性類型 CloudFormation 和 AWS SAM 支援的參考資訊，請參閱AWS CloudFormation 《使用者指南》中的[AWS 資源和屬性類型參考](#)。

主題

- [AWS::Serverless::Api](#)
- [AWS::Serverless::Application](#)
- [AWS::Serverless::CapacityProvider](#)
- [AWS::Serverless::Connector](#)
- [AWS::Serverless::Function](#)
- [AWS::Serverless::GraphQLApi](#)
- [AWS::Serverless::HttpApi](#)
- [AWS::Serverless::WebSocketApi](#)
- [AWS::Serverless::LayerVersion](#)
- [AWS::Serverless::MicrovmImage](#)
- [AWS::Serverless::NetworkConnector](#)

- [AWS::Serverless::SimpleTable](#)
- [AWS::Serverless::StateMachine](#)
- [AWS::Serverless::WebFunction](#)

AWS::Serverless::Api

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

[AWS::Serverless::Api](#) 資源不需要明確新增至無 AWS 伺服器應用程式定義範本。此類型的資源會隱含地從未參考 [AWS::Serverless::Api](#) 資源的範本中定義的 [AWS::Serverless::Function](#) 資源上定義的 Api 事件聯集建立。

應使用 [AWS::Serverless::Api](#) 資源來定義和記錄使用 OpenApi 的 API，以便更能夠設定基礎 Amazon API Gateway 資源。

我們建議您使用 CloudFormation 勾點或 IAM 政策來驗證 API Gateway 資源是否已連接授權方，以控制對它們的存取。

如需使用 CloudFormation 勾點的詳細資訊，請參閱 CloudFormation CLI 使用者指南中的 [註冊勾點](#) 和 [apigw-enforce-authorizer](#) GitHub 儲存庫。

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，會將您的 AWS SAM 資源 AWS SAM 轉換為 CloudFormation 資源。如需詳細資訊，請參閱 [為產生 CloudFormation 資源 AWS SAM](#)。

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](#)
[EndpointAccessMode](#): *String*
[EndpointConfiguration](#): [EndpointConfiguration](#)
[FailOnWarnings](#): *Boolean*
[GatewayResponses](#): *Map*
[MergeDefinitions](#): *Boolean*
[MethodSettings](#): [MethodSettings](#)
[MinimumCompressionSize](#): *Integer*
[Mode](#): *String*
[Models](#): *Map*
[Name](#): *String*
[OpenApiVersion](#): *String*
[PropagateTags](#): *Boolean*
[Policy](#): *JSON*
[SecurityPolicy](#): *String*
[StageName](#): *String*
[Tags](#): *Map*
[TracingEnabled](#): *Boolean*
[Variables](#): *Map*

Properties

AccessLogSetting

設定階段的存取日誌設定。

類型：[AccessLogSetting](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGateway::Stage` 資源的 [AccessLogSetting](#) 屬性。

AlwaysDeploy

即使未偵測到 API 的變更，也一律部署 API。

類型：布林值

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGateway::RestApi` 資源的 [ApiKeySourceType](#) 屬性。

Auth

設定授權以控制對 API Gateway API 的存取。

如需使用 設定存取權的詳細資訊，AWS SAM 請參閱 [使用 AWS SAM 範本控制 API 存取](#)。如需示範如何覆寫全域授權方的範例，請參閱 [the section called “覆寫 Amazon API Gateway REST API 的全域授權方”](#)。

類型：[ApiAuth](#)

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

BinaryMediaTypes

您的 API 可能傳回的 MIME 類型清單。使用此項目可啟用 APIs 的二進位支援。

類型：清單

必要：否

CloudFormation 相容性：此屬性類似於 `AWS::ApiGateway::RestApi` 資源的 [BinaryMediaTypes](#) 屬性。BinaryMediaTypes 的清單會同時新增至 CloudFormation 資源和 OpenAPI 文件。

CacheClusterEnabled

指出是否已為階段啟用快取。若要快取回應，您還必須在 `true` 下 `CachingEnabled` 將設定為 `MethodSettings`。

類型：布林值

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGateway::Stage` 資源的 [CacheClusterSize](#) 屬性。

CanarySetting

將 Canary 設定設定為一般部署的階段。

類型：[CanarySetting](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGateway::Stage` 資源的 [CanarySetting](#) 屬性。

Cors

管理所有 API Gateway APIs 的跨來源資源共用 (CORS)。指定要允許做為字串的網域，或指定具有其他 Cors 組態的字典。

Note

CORS 需要 AWS SAM 修改您的 OpenAPI 定義。在 中建立內嵌 OpenAPI 定義 `DefinitionBody` 以開啟 CORS。

如需 CORS 的詳細資訊，請參閱《[API Gateway 開發人員指南](#)》中的為 [API Gateway REST API 資源啟用 CORS](#)。

類型：字串 | [CorsConfiguration](#)

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

DefinitionBody

描述 API 的 OpenAPI 規格。如果既未指定DefinitionUri也DefinitionBody未指定，SAM 會根據您的範本組態DefinitionBody為您產生。

若要參考定義 API 的本機OpenAPI檔案，請使用 `AWS::Include`轉換。如需詳細資訊，請參閱 [如何 AWS SAM 上傳本機檔案](#)。

類型：JSON

必要：否

CloudFormation 相容性：此屬性類似於 `AWS::ApiGateway::RestApi` 資源的 [Body](#) 屬性。如果提供特定屬性，內容可能會在傳遞至 CloudFormation 之前插入或修改至 DefinitionBody。屬性包括對應 EventSource的 Auth、BinaryMediaTypes、Models、Cors GatewayResponses和 Api 類型的 `AWS::Serverless::Function`。

DefinitionUri

定義 API 之 OpenAPI 文件的 Amazon S3 Uri、本機檔案路徑或位置物件。此屬性參考的 Amazon S3 物件必須是有效的 OpenAPI 檔案。如果既未指定DefinitionUri也DefinitionBody未指定，SAM 會根據您的範本組態DefinitionBody為您產生。

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

參考的外部 OpenApi 檔案不支援內部函數DefinitionUri。使用 DefinitionBody 屬性搭配[包含轉換](#)，將 OpenApi 定義匯入範本。

類型：字串 | [ApiDefinition](#)

必要：否

CloudFormation 相容性：此屬性類似於 `AWS::ApiGateway::RestApi` 資源的 [BodyS3Location](#) 屬性。巢狀 Amazon S3 屬性的名稱不同。

Description

Api 資源的描述。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `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。根據預設，用戶端可以使用預設的叫用您的 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

CloudFormation 相容性：此屬性類似於 `AWS::ApiGateway::RestApi` 資源的 [DisableExecuteApiEndpoint](#) 屬性。它會直接傳遞至 [x-amazon-apigateway-endpoint-configuration](#) 延伸的 `disableExecuteApiEndpoint` 屬性，該延伸會新增至 `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

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

EndpointConfiguration

REST API 的端點類型。

類型：[EndpointConfiguration](#)

必要：否

CloudFormation 相容性：此屬性類似於 `AWS::ApiGateway::RestApi` 資源的 [EndpointConfiguration](#) 屬性。巢狀組態屬性的名稱不同。

EndpointAccessMode

REST API 端點的存取模式。使用增強型安全政策時為必要（字首為 的政策 `SecurityPolicy_`）。

有效值：STRICT 或 BASIC

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGateway::RestApi` 資源的 [EndpointAccessMode](#) 屬性。

FailOnWarnings

指定在遇到警告時是否要轉返 API 建立 (true) 與否 (false)。預設值為 false。

類型：布林值

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGateway::RestApi` 資源的 [FailOnWarnings](#) 屬性。

GatewayResponses

設定 API 的闡道回應。闡道回應是由 API Gateway 直接或透過使用 Lambda 授權方傳回的回應。如需詳細資訊，請參閱適用於 [闡道回應的 Api Gateway OpenApi 擴充功能](#) 文件。

類型：映射

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

MergeDefinitions

AWS SAM 會從 API 事件來源產生 OpenAPI 規格。指定 `AWS SAM true` 讓 將此合併到 `AWS::Serverless::Api` 資源中定義的內嵌 OpenAPI 規格。指定 `false` 不合併。

MergeDefinitions 需要AWS::Serverless::Api定義的 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

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

MethodSettings

設定 API 階段的所有設定，包括記錄、指標、CacheTTL、調節。

類型：[MethodSetting](#) 的清單

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::ApiGateway::Stage 資源的
[MethodSettings](#) 屬性。

MinimumCompressionSize

允許根據用戶端的 Accept-Encoding 標頭壓縮回應內文。當回應內文大小大於或等於您設定的閾值時，就會觸發壓縮。主體大小閾值上限為 10 MB (10,485,760 個位元組)。- 支援下列壓縮類型：gzip、deflate 和 identity。

類型：整數

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGateway::RestApi` 資源的 [Mode](#) 屬性。

Models

您的 API 方法要使用的結構描述。您可以使用 JSON 或 YAML 描述這些結構描述。如需範例模型，請參閱此頁面底部的範例一節。

類型：映射

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

Name

API Gateway RestApi 資源的名稱

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGateway::RestApi` 資源的 [Name](#) 屬性。

OpenApiVersion

要使用的 OpenApi 版本。這可以是 2.0 用於 Swagger 規格，或其中一個 OpenApi 3.0 版本，例如 3.0.1。如需 OpenAPI 的詳細資訊，請參閱 [OpenAPI 規格](#)。

Note

AWS SAM 根據預設，會建立名為 Stage 的階段。將此屬性設定為任何有效值，將導致無法建立階段 Stage。

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

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。指定 True 在產生的資源中傳播標籤。

類型：布林值

必要：否

預設：False

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

Policy

包含 API 許可的政策文件。若要設定政策的 ARN，請使用 !Join 內部函數搭配 "" 做為分隔符號，並使用值 "execute-api:/" 和 "*"。

類型：JSON

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::ApiGateway::RestApi 資源的 [政策](#) 屬性。

SecurityPolicy

REST API 的 TLS 版本加上密碼套件。使用增強型安全政策（字首為 的政策 SecurityPolicy>）時，您還必須設定 EndpointAccessMode。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::ApiGateway::RestApi 資源的 [SecurityPolicy](#) 屬性。

StageName

API Gateway 做為叫用統一資源識別符 (URI) 中第一個路徑區段的階段名稱。

若要參考階段資源，請使用 *<api-logical-id>.Stage*。如需參考指定資源時產生的 [AWS::Serverless::Api](#) 資源的詳細資訊，請參閱 [CloudFormation 指定 AWS::Serverless::Api 時產生的資源](#)。如需所產生 CloudFormation 資源的一般資訊，請參閱 [為產生 CloudFormation 資源 AWS SAM](#)。

類型：字串

必要：是

CloudFormation 相容性：此屬性類似於 `AWS::ApiGateway::Stage` 資源的 [StageName](#) 屬性。在 SAM 中為必要，但在 API Gateway 中為非必要

其他備註：隱含 API 的階段名稱為 "Prod"。

Tags

映射（字串到字串），指定要新增至此 API Gateway 階段的標籤。如需標籤有效金鑰和值的詳細資訊，請參閱 AWS CloudFormation 《使用者指南》中的 [資源標籤](#)。

類型：映射

必要：否

CloudFormation 相容性：此屬性類似於 `AWS::ApiGateway::Stage` 資源的 [Tags](#) 屬性。SAM 中的標籤屬性包含 Key : Value 對；在 CloudFormation 中包含標籤物件清單。

TracingEnabled

指出是否已啟用階段的 X-Ray 主動追蹤。如需 X-Ray 的詳細資訊，請參閱《[APIs Gateway 開發人員指南](#)》中的 [使用 X-Ray 追蹤使用者對 REST API 的請求](#)。

類型：布林值

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGateway::Stage` 資源的 [TracingEnabled](#) 屬性。

Variables

定義階段變數的映射（字串到字串），其中變數名稱是索引鍵，而變數值是值。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：映射

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGateway::Stage` 資源的 [Variables](#) 屬性。

傳回值

Ref

當將此資源的邏輯 ID 提供給Ref內部 函數時，它會傳回基礎 API Gateway API 的 ID。

如需使用 Ref函數的詳細資訊，請參閱AWS CloudFormation 《使用者指南[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。

如需使用 的詳細資訊Fn::GetAtt，請參閱AWS CloudFormation 《使用者指南[Fn::GetAtt](#)》中的 。

RootResourceId

RestApi 資源的根資源 ID，例如：a0bc123d4e。

範例

SimpleApiExample

Hello World AWS SAM 範本檔案，其中包含具有 API 端點的 Lambda 事件函數。這是運作中無伺服器應用程式的完整 AWS SAM 範本檔案。

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

具有外部 Swagger 檔案中定義之 API 的 AWS SAM 範本程式碼片段，以及 Lambda 整合和 CORS 組態。這只是 AWS SAM 範本檔案的一部分，顯示 [AWS::Serverless::Api](#) 定義。

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 API 使用 Amazon Cognito 對 API 授權請求。這只是 AWS SAM 範本檔案的一部分，顯示 [AWS::Serverless::Api](#) 定義。

YAML

```

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

```

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

ApiModelsExample

包含模型結構描述之 API 的 AWS SAM 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 [AWS::Serverless::Api](#) 的定義。

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
```

快取範例

Hello World AWS SAM 範本檔案，其中包含具有 API 端點的 Lambda 函數。API 已針對一個資源和方法啟用快取。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}
```

具有私有 API 範例的自訂網域

Hello World AWS SAM 範本檔案，其中包含與私有網域對應之 API 端點的 Lambda 函數。範本會在 VPC 端點與私有網域之間建立網域存取關聯。如需詳細資訊，請參閱 [API Gateway 中私有 APIs 的自訂網域名稱](#)。

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: AWS SAM template configured with a custom domain using a private API

Parameters:
  DomainName:
    Type: String
    Default: mydomain.example.com
  CertificateArn:
    Type: String
  HostedZoneId:
    Type: String
  VpcEndpointId:
    Type: String
  VpcEndpointDomainName:
    Type: String
  VpcEndpointHostedZoneId:
    Type: String

Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      InlineCode: |
        def handler(event, context):
            return {'body': 'Hello World!', 'statusCode': 200}
      Handler: index.handler
      Runtime: python3.13
      Events:
        Fetch:
          Type: Api
          Properties:
            RestApiId:
```

```
        Ref: MyApi
        Method: Get
        Path: /get
MyApi:
  Type: AWS::Serverless::Api
  Properties:
    StageName: Prod
    EndpointConfiguration:
      Type: PRIVATE
      VPCEndpointIds:
        - !Ref VpcEndpointId
  Policy:
    Version: '2012-10-17'
    Statement:
      - Effect: Allow
        Principal: '*'
        Action: execute-api:Invoke
        Resource: execute-api/*
      - Effect: Deny
        Principal: '*'
        Action: execute-api:Invoke
        Resource: execute-api/*
        Condition:
          StringNotEquals:
            aws:SourceVpce: !Ref VpcEndpointId
  Domain:
    DomainName: !Ref DomainName
    CertificateArn: !Ref CertificateArn
    EndpointConfiguration: PRIVATE
    BasePath:
      - /
  Route53:
    HostedZoneId: !Ref HostedZoneId
    VpcEndpointDomainName: !Ref VpcEndpointDomainName
    VpcEndpointHostedZoneId: !Ref VpcEndpointHostedZoneId
  AccessAssociation:
    VpcEndpointId: !Ref VpcEndpointId
  Policy:
    Version: '2012-10-17'
    Statement:
      - Effect: Allow
        Principal: '*'
        Action: execute-api:Invoke
        Resource: execute-api/*
```

```

- Effect: Deny
  Principal: '*'
  Action: execute-api:Invoke
  Resource: execute-api:/*
  Condition:
    StringNotEquals:
      aws:SourceVpce: !Ref VpcEndpointId

```

ApiAuth

設定授權以控制對 API Gateway API 的存取。

如需使用 設定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 |
AWS_IAM
DefaultAuthorizer: String
InvokeRole: String
ResourcePolicy: ResourcePolicyStatement
UsagePlan: ApiUsagePlan

```

Note

Authorizers 屬性包含 AWS_IAM，但 不需要額外的組態AWS_IAM。如需範例，請參閱 [AWS IAM](#)。

Properties

AddApiKeyRequiredToCorsPreflight

如果已設定 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

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

AddDefaultAuthorizerToCorsPreflight

如果已設定 DefaultAuthorizer 和 Cors 屬性，則設定 AddDefaultAuthorizerToCorsPreflight 會導致預設授權方新增至 OpenAPI 區段中的 Options 屬性。

類型：布林值

必要：否

預設：True

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

ApiKeyRequired

如果設定為 true，則所有 API 事件都需要 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 [使用 API 金鑰建立和使用用量計劃](#)。

類型：布林值

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

Authorizers

用於控制 API Gateway API 存取的授權方。

如需詳細資訊，請參閱 [使用 AWS SAM 範本控制 API 存取](#)。

類型：[CognitoAuthorizer](#) | [LambdaTokenAuthorizer](#) | [LambdaRequestAuthorizer](#) | AWS_IAM

必要：否

預設：無

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

其他備註：SAM 會將授權方新增至 Api 的 OpenApi 定義。

DefaultAuthorizer

指定 API Gateway API 的預設授權方，該 API Gateway API 預設將用於授權 API 呼叫。

Note

如果與此 API 相關聯之函數的 Api EventSource 設定為使用 IAM 許可，則此屬性必須設定為 AWS_IAM，否則會產生錯誤。

類型：字串

必要：否

預設：無

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

InvokeRole

將所有資源和方法的整合登入資料設定為此值。

CALLER_CREDENTIALS 會映射到 `arn:aws:iam::<user>/`，使用呼叫者登入資料來叫用端點。

有效值：CALLER_CREDENTIALS、NONE、IAMRoleArn

類型：字串

必要：否

預設：CALLER_CREDENTIALS

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

ResourcePolicy

設定 API 上所有方法和路徑的資源政策。

類型：[ResourcePolicyStatement](#)

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

其他備註：您也可以 `AWS::Serverless::Function` 使用 在個別 上定義此設定 [ApiFunctionAuth](#)。對於具有 APIs，這是必要的 `EndpointConfiguration: PRIVATE`。

UsagePlan

設定與此 API 相關聯的用量計劃。如需用量計劃的詳細資訊，請參閱《[API Gateway 開發人員指南](#)》中的 [使用 API 金鑰建立和使用用量計劃](#)。

此 AWS SAM 屬性會在設定此屬性時產生三個額外的 CloudFormation 資源：[AWS::ApiGateway::UsagePlan](#)、[AWS::ApiGateway::UsagePlanKey](#)和 [AWS::ApiGateway::ApiKey](#)。如需此案例的資訊，請參閱 [已指定 UsagePlan 屬性](#)。如需所產生 CloudFormation 資源的一般資訊，請參閱 [為 產生 CloudFormation 資源 AWS SAM](#)。

類型：[ApiUsagePlan](#)

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

範例

CognitoAuth

Cognito 驗證範例

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"
      }
    }
  }]
  IpRangeDenylist:
    - "10.20.30.40"

```

AWS IAM

AWS IAM 範例

YAML

```

Auth:
  Authorizers: AWS_IAM

```

ApiUsagePlan

設定 API Gateway API 的用量計劃。如需用量計劃的詳細資訊，請參閱《[API Gateway 開發人員指南](#)》中的[使用 API 金鑰建立和使用用量計劃](#)。

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*

Properties

CreateUsagePlan

決定此用量計劃的設定方式。有效值為 PER_API、SHARED 和 NONE。

PER_API 會建立此 API 特有的 [AWS::ApiGateway::ApiKey](#)、[AWS::ApiGateway::UsagePlan](#) 和 [AWS::ApiGateway::UsagePlanKey](#) 資源。這些資源的邏輯 IDs *<api-logical-id>*UsagePlanKey 分別為 *<api-logical-id>*UsagePlan、*<api-logical-id>*ApiKey 和 。

SHARED 會建立 [AWS::ApiGateway::UsagePlan](#)、[AWS::ApiGateway::ApiKey](#) 和 [AWS::ApiGateway::UsagePlanKey](#) 資源，這些資源在相同 AWS SAM 範本 CreateUsagePlan: SHARED 中也有 的任何 API 之間共用。這些資源的邏輯 IDs ServerlessUsagePlanKey 分別為 ServerlessUsagePlan、ServerlessApiKey 和 。如果您使用此選項，建議您僅針對一個 API 資源新增此用量計劃的其他組態，以避免衝突的定義和不確定狀態。

NONE 會停用與此 API 建立或關聯用量計劃。只有在 中指定 PER_API SHARED 或 時，才需要這樣做 [AWS SAM 範本的全域區段](#)。

有效值：PER_API、SHARED 與 NONE

類型：字串

必要：是

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

Description

用量計劃的描述。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 [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

CloudFormation 相容性：此屬性會直接傳遞至 `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)。

此屬性使用 [CloudFormation 標籤類型](#)。

類型：清單

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGateway::UsagePlan` 資源的 [Throttle](#) 屬性。

UsagePlanName

用量計劃的名稱。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGateway::UsagePlan` 資源的 [UsagePlanName](#) 屬性。

範例

UsagePlan

以下是用量計劃範例。

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 使用者集區授權方。

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 | List
```

Properties

AuthorizationScopes

此授權方的授權範圍清單。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

Identity

此屬性可用於在 授權方的傳入請求IdentitySource中指定。

類型：[CognitoAuthorizationIdentity](#)

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

UserPoolArn

用於授權的 Amazon Cognito 使用者集區 ARN (s)。將單一 ARN 指定為字串，或將多個 ARNs 指定為清單，以使用多個使用者集區。

類型：字串 | 清單

必要：是

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

範例

CognitoAuth

Cognito 驗證範例

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 授權方的傳入請求中指定 IdentitySource。如需 IdentitySource 的詳細資訊，請參閱 [ApiGateway 授權方 OpenApi 延伸](#)。

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
```

Properties

Header

在 OpenApi 定義中指定授權的標頭名稱。

類型：字串

必要：否

預設：授權

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

ReauthorizeEvery

存活期 (TTL) 期間 (秒)，指定 API Gateway 快取授權方結果的時間。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：300

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

ValidationExpression

指定驗證表達式以驗證傳入的 Identity

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

範例

CognitoAuthIdentity

YAML

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

LambdaRequestAuthorizer

設定 Lambda 授權方，以使用 Lambda 事件函數控制對 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*

Properties

DisableFunctionDefaultPermissions

指定 `true` AWS SAM 以防止 自動建立 `AWS::Lambda::Permissions` 資源，在您的 `AWS::Serverless::Api` 資源和授權方 `Lambda` 函數之間佈建許可。

預設值：false

類型：布林值

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

FunctionArn

指定提供 API 授權的 `Lambda` 函數的函數 ARN。

Note

AWS SAM 當 `FunctionArn` 針對 指定 時，會自動建立 `AWS::Lambda::Permissions` 資源 `AWS::Serverless::Api`。 `AWS::Lambda::Permissions` 資源會在您的 API 和授權方 `Lambda` 函數之間佈建許可。

類型：字串

必要：是

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

FunctionInvokeRole

將授權方憑證新增至 `Lambda` 授權方的 `OpenApi` 定義。

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

FunctionPayloadType

此屬性可用來定義 API 的 Lambda 授權方類型。

有效值：TOKEN 或 REQUEST

類型：字串

必要：否

預設：TOKEN

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

Identity

此屬性可用於在 授權方的傳入請求IdentitySource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

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

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 授權方的傳入請求中指定 IdentitySource。如需 IdentitySource 的詳細資訊，請參閱 [ApiGateway 授權方 OpenApi 延伸](#)。

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
QueryString: List
ReauthorizeEvery: Integer
StageVariables: List
```

Properties

Context

將指定的內容字串轉換為格式 的映射表達式 `context.contextString`。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

Headers

將標頭轉換為格式 映射表達式的逗號分隔字串 `method.request.header.name`。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

QueryString

將指定的查詢字串轉換為格式 映射表達式的逗號分隔字串 `method.request.querystring.queryString`。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

ReauthorizeEvery

存活期 (TTL) 期間 (秒)，指定 API Gateway 快取授權方結果的時間。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：300

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

StageVariables

將指定的階段變數轉換為以逗號分隔的格式映射表達式字串 `stageVariables.stageVariable`。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

範例

LambdaRequestIdIdentity

YAML

```
Identity:
```

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

LambdaTokenAuthorizer

設定 Lambda 授權方，以使用 Lambda 事件函數控制對 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
```

Properties

DisableFunctionDefaultPermissions

指定 true AWS SAM 以防止 自動建立 `AWS::Lambda::Permissions` 資源，在您的 `AWS::Serverless::Api` 資源和授權方 Lambda 函數之間佈建許可。

預設值：false

類型：布林值

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

FunctionArn

指定提供 API 授權的 Lambda 函數的函數 ARN。

Note

AWS SAM 當 FunctionArn 針對 指定 時，會自動建立 `AWS::Lambda::Permissions` 資源 `AWS::Serverless::Api`。AWS::Lambda::Permissions 資源會在您的 API 和授權方 Lambda 函數之間佈建許可。

類型：字串

必要：是

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

FunctionInvokeRole

將授權方憑證新增至 Lambda 授權方的 OpenApi 定義。

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

FunctionPayloadType

此屬性可用來定義 Api 的 Lambda 授權方類型。

有效值：TOKEN 或 REQUEST

類型：字串

必要：否

預設：TOKEN

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

Identity

此屬性可用於在 授權方的傳入請求 IdentitySource 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

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

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 授權方的傳入請求中指定 IdentitySource。如需 IdentitySource 的詳細資訊，請參閱 [ApiGateway 授權方 OpenApi 延伸](#)。

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
```

Properties

Header

在 OpenApi 定義中指定授權的標頭名稱。

類型：字串

必要：否

預設：授權

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

ReauthorizeEvery

存活期 (TTL) 期間 (秒)，指定 API Gateway 快取授權方結果的時間。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：300

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

ValidationExpression

指定驗證表達式以驗證傳入的 Identity。

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

範例

LambdaTokenIdentity

YAML

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

ResourcePolicyStatement

為 API 的所有方法和路徑設定資源政策。如需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 資源政策控制對 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
```

Properties

AwsAccountBlacklist

要封鎖 AWS 的帳戶。

類型：字串的清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

AwsAccountWhitelist

要允許 AWS 的帳戶。如需此屬性的範例使用方式，請參閱此頁面底部的範例區段。

類型：字串的清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

CustomStatements

要套用至此 API 的自訂資源政策陳述式清單。如需此屬性的範例使用方式，請參閱此頁面底部的範例區段。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

IntrinsicVpcBlacklist

要封鎖的虛擬私有雲端 (VPCs) 清單，其中每個 VPC 指定為參考，例如[動態參考](#)或Ref[內部函數](#)。如需此屬性的範例使用方式，請參閱此頁面底部的範例區段。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

IntrinsicVpcWhitelist

要允許的 VPCs 清單，其中每個 VPC 指定為參考，例如[動態參考](#)或Ref[內部函數](#)。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

IntrinsicVpceBlacklist

要封鎖的 VPC 端點清單，其中每個 VPC 端點指定為參考，例如[動態參考](#)或Ref [內部函數](#)。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

IntrinsicVpceWhitelist

要允許的 VPC 端點清單，其中每個 VPC 端點指定為參考，例如[動態參考](#)或Ref [內部函數](#)。如需此屬性的範例使用方式，請參閱此頁面底部的範例區段。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

IpRangeBlacklist

要封鎖的 IP 地址或地址範圍。如需此屬性的範例使用方式，請參閱此頁面底部的範例區段。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

IpRangeWhitelist

要允許的 IP 地址或地址範圍。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

SourceVpcBlacklist

要封鎖的來源 VPC 或 VPC 端點。來源 VPC 名稱必須以 開頭"vpc-"，來源 VPC 端點名稱必須以 開頭"vpce-"。如需此屬性的範例使用方式，請參閱此頁面底部的範例區段。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

SourceVpcWhitelist

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

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

範例

資源政策範例

下列範例會封鎖兩個 IP 地址和來源 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

定義 API 的 OpenAPI 文件。

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
Key: String
Version: String
```

Properties

Bucket

存放 OpenAPI 檔案的 Amazon S3 儲存貯體名稱。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGateway::RestApi` `S3Location` 資料類型的 [Bucket](#) 屬性。

Key

OpenAPI 檔案的 Amazon S3 金鑰。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGateway::RestApi` `S3Location` 資料類型的 [Key](#) 屬性。

Version

對於版本控制的物件，則為 OpenAPI 檔案的版本。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGateway::RestApi` `S3Location` 資料類型的 [Version](#) 屬性。

範例

定義 Uri 範例

API 定義範例

YAML

```
DefinitionUri:
  Bucket: amzn-s3-demo-bucket-name
  Key: mykey-name
  Version: 121212
```

CorsConfiguration

管理 API Gateway APIs 的跨來源資源共用 (CORS)。指定要允許做為字串的網域，或指定具有其他 Cors 組態的字典。

Note

CORS 需要 AWS SAM 修改您的 OpenAPI 定義。在 中建立內嵌 OpenAPI 定義 `DefinitionBody` 以開啟 CORS。如果在 OpenAPI 定義和屬性層級設定 `CorsConfiguration`，會 AWS SAM 合併它們。屬性層級優先於 OpenAPI 定義。

如需 CORS 的詳細資訊，請參閱《[API Gateway 開發人員指南](#)》中的為 `API Gateway REST API` 資源啟用 CORS。

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
```

Properties

AllowCredentials

布林值，指出是否允許請求包含登入資料。

類型：布林值

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

AllowHeaders

要允許的標頭字串。

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

AllowMethods

包含要允許之 HTTP 方法的字串。

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

AllowOrigin

要允許的原始伺服器字串。這可以是字串格式的逗號分隔清單。

類型：字串

必要：是

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

MaxAge

包含快取 CORS 預檢請求秒數的字串。

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

範例

CorsConfiguration

CORS 組態範例。這只是 AWS SAM 範本檔案的一部分，顯示已設定 CORS 和 [AWS::Serverless::Api](#) 的定義 [AWS::Serverless::Function](#)。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: "'https://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': 'https://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

```

AccessAssociation: DomainAccessAssociation
BasePath: List
CertificateArn: String
DomainName: String
EndpointAccessMode: String
EndpointConfiguration: String
MutualTlsAuthentication: MutualTlsAuthentication

```

```
NormalizeBasePath: Boolean  
OwnershipVerificationCertificateArn: String  
Policy: Json  
Route53: Route53Configuration  
SecurityPolicy: String
```

Properties

AccessAssociation

產生 `AWS::ApiGateway::DomainNameAccessAssociation` 資源所需的組態。

AWS SAM 會在設定此屬性時產生 [AWS::ApiGateway::DomainNameAccessAssociation](#) 資源。如需所產生 CloudFormation 資源的資訊，請參閱 [為產生 CloudFormation 資源 AWS SAM](#)。

類型：[DomainAccessAssociation](#)

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

BasePath

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：/

CloudFormation 相容性：此屬性類似於 `AWS::ApiGateway::BasePathMapping` 資源的 [BasePath](#) 屬性。會 AWS SAM 建立多個 `AWS::ApiGateway::BasePathMapping` 資源，此屬性中 `BasePath` 指定的每個資源一個。

CertificateArn

此網域名稱端點 AWS 受管憑證的 Amazon Resource Name (ARN)。AWS Certificate Manager 是唯一支援的來源。

類型：字串

必要：是

CloudFormation 相容性：此屬性類似於 `AWS::ApiGateway::DomainName` 資源的 [CertificateArn](#) 屬性。如果 `EndpointConfiguration` 設定為 `REGIONAL` (預設值)，則會 `CertificateArn` 映射至 中的 [RegionalCertificateArn](#) `AWS::ApiGateway::DomainName`。如果 `EndpointConfiguration` 設定為 `EDGE`，會 `CertificateArn` 映射至 中的 [CertificateArn](#) `AWS::ApiGateway::DomainName`。如果 `EndpointConfiguration` 設定為 `PRIVATE`，則此屬性會傳遞至 [AWS::ApiGateway::DomainNameV2](#) 資源。

其他備註：對於 `EDGE` 端點，您必須在 `us-east-1` AWS 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 會在設定此屬性時產生 [AWS::ApiGateway::DomainName](#) 資源。如需此案例的資訊，請參閱 [已指定 DomainName 屬性](#)。如需所產生 CloudFormation 資源的資訊，請參閱 [為產生 CloudFormation 資源 AWS SAM](#)。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGateway::DomainName` 資源的 [DomainName](#) 屬性，或在 `EndpointConfiguration` 設定為 [AWS::ApiGateway::DomainNameV2](#) 時傳遞至 `PRIVATE`。

EndpointConfiguration

定義要映射到自訂網域的 API Gateway 端點類型。此屬性的值會決定 `CertificateArn` 屬性的映射方式 CloudFormation。

有效值：`EDGE`、`REGIONAL` 或 `PRIVATE`

類型：字串

必要：否

預設：`REGIONAL`

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等的。

EndpointAccessMode

自訂網域名稱端點的存取模式。使用增強型安全政策 (字首為 `SecurityPolicy_`) 時需要。

有效值：STRICT 或 BASIC

類型：字串

必要：有條件

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGateway::DomainName` 資源的 [EndpointAccessMode](#) 屬性，或在 `EndpointConfiguration` 設定為 [AWS::ApiGateway::DomainNameV2](#) 時傳遞至 `PRIVATE`。

MutualTlsAuthentication

自訂網域名稱的相互 Transport Layer Security (TLS) 身分驗證組態。

類型：[MutualTlsAuthentication](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGateway::DomainName` 資源的 [MutualTlsAuthentication](#) 屬性。

NormalizeBasePath

指出 `BasePath` 屬性定義的 `Basepaths` 中是否允許非英數字元。設為 `True`，會從 `basepaths` 中移除非英數字元。

`NormalizeBasePath` 搭配 `BasePath` 屬性使用。

類型：布林值

必要：否

預設：True

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等的。

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。只有在您設定交互 TLS 並指定的 ACM 匯入或私有 CA 憑證 ARN 時才需要 `CertificateArn`。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGateway::DomainName` 資源的 [OwnershipVerificationCertificateArn](#) 屬性。

Policy

要連接至 API Gateway 網域名稱的 IAM 政策。僅適用於 `EndpointConfiguration` 設定為 `PRIVATE`。

類型：Json

必要：否

CloudFormation 相容性：當 `EndpointConfiguration` 設定為 `PRIVATE` 時，此屬性會直接傳遞至 `AWS::ApiGateway::DomainNameV2` 資源的 `Policy` 屬性 `PRIVATE`。如需有效政策文件的範例，請參閱 [AWS::ApiGateway::DomainNameV2](#)。

Route53

定義 Amazon Route 53 組態。

類型：[Route53Configuration](#)

必要：否

CloudFormation 相容性：此屬性對 `Route53Configuration` 是唯一的 AWS SAM，並且沒有 CloudFormation 同等的。

SecurityPolicy

此網域名稱的 TLS 版本加上密碼套件。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGateway::DomainName` 資源的 [SecurityPolicy](#) 屬性，或在 `EndpointConfiguration` 設定為 `PRIVATE` 時傳遞至 `AWS::ApiGateway::DomainNameV2`。對於 `PRIVATE` 端點，僅支援 `TLS_1_2`。

範例

DomainName

DomainName 範例

YAML

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

DomainAccessAssociation

設定存取關聯來源與私有自訂網域名稱之間的網域存取關聯。唯一支援的存取關聯來源是 VPC 端點 ID。如需詳細資訊，請參閱 [API Gateway 中私有 APIs 的自訂網域名稱](#)。

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

```
VpcEndpointId: String
```

Properties

VpcEndpointId

與 API Gateway VPC 服務相關聯之 VPC 介面端點的端點 ID。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至
AWS::ApiGateway::DomainNameAccessAssociation 資源的
[AccessAssociationSource](#) 屬性。

Route53Configuration

設定 API 的 Route53 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  
VpcEndpointDomainName: String  
VpcEndpointHostedZoneId: String
```

Properties

DistributionDomainName

設定 API 自訂網域名稱的自訂分佈。

類型：字串

必要：否

預設：使用 API Gateway 分佈。

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Route53::RecordSetGroup` `AliasTarget` 資源的 [DNSName](#) 屬性。

其他備註：[CloudFront 分佈](#)的網域名稱。

EvaluateTargetHealth

當 `EvaluateTargetHealth` 為 `true` 時，別名記錄會繼承參考 AWS 資源的運作狀態，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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Route53::RecordSetGroup` `AliasTarget` 資源的 [EvaluateTargetHealth](#) 屬性。

其他備註：當別名目標為 CloudFront 分佈時，您無法將 EvaluateTargetHealth 設定為 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

CloudFormation 相容性：此屬性會直接傳遞至 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

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Route53::RecordSetGroup RecordSet 資源的 [HostedZoneName](#) 屬性。

IpV6

設定此屬性時，會 AWS SAM 建立AWS::Route53::RecordSet資源，並將提供的 HostedZone AAAA的 [Type](#) 設定為 。

類型：布林值

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

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。資源通常是 AWS 資源，例如 EC2 執行個體或 ELB 負載平衡器，並且根據記錄類型由 IP 地址或 DNS 網域名稱參考。

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

CloudFormation 相容性：此屬性會直接傳遞至

AWS::Route53::RecordSetGroupRecordSet 資料類型的 [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

CloudFormation 相容性：此屬性會直接傳遞至

AWS::Route53::RecordSetGroupRecordSet 資料類型的 [SetIdentifier](#) 屬性。

VpcEndpointDomainName

與 API Gateway VPC 服務相關聯之 VPC 介面端點的 DNS 名稱。只有私有網域才需要此屬性。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Route53::RecordSet AliasTarget` 欄位的 [DNSName](#) 屬性。

VpcEndpointHostedZoneId

與 API Gateway VPC 服務相關聯之 VPC 介面端點的託管區域 ID。只有私有網域才需要此屬性。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Route53::RecordSet AliasTarget` 欄位的 [HostedZoneId](#) 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: ABCDEFGHIJKLMN0P
```

EndpointConfiguration

REST 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

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

Properties

IpAddressType

可叫用 API (RestApi) 的 IP 地址類型。

有效值：ipv4 或 dualstack

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::ApiGateway::RestApi EndpointConfiguration 資料類型的 [IpAddressType](#) 屬性。

Type

REST API 的端點類型。

有效值：EDGE 或 REGIONAL 或 PRIVATE

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::ApiGateway::RestApi EndpointConfiguration 資料類型的 [Types](#) 屬性。

VPCEndpointIds

要建立 Route53 別名之 REST API 的 VPC 端點 IDs 清單。

類型：清單

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGateway::RestApi` `EndpointConfiguration` 資料類型的 [VpcEndpointIds](#) 屬性。

範例

EndpointConfiguration

端點組態範例

YAML

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

AWS::Serverless::Application

將來自 [AWS Serverless Application Repository](#) 或來自 Amazon S3 儲存貯體的無伺服器應用程式內嵌為巢狀應用程式。巢狀應用程式會部署為巢狀 [AWS::CloudFormation::Stack](#) 資源，其中可包含多個其他資源，包括其他 [AWS::Serverless::Application](#) 資源。

Note

當您部署到時 AWS CloudFormation，會將您的 AWS SAM 資源 AWS SAM 轉換為 CloudFormation 資源。如需詳細資訊，請參閱 [為產生 CloudFormation 資源 AWS SAM](#)。

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
```

Properties

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，其必須遵循 [CloudFormation TemplateUrl 文件](#) 中指定的格式，並包含有效的 CloudFormation 或 SAM 範本。[ApplicationLocationObject](#) 可用來指定已發佈至 的應用程式 [AWS Serverless Application Repository](#)。

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

類型：字串 | [ApplicationLocationObject](#)

必要：是

CloudFormation 相容性：此屬性類似於 `AWS::CloudFormation::Stack` 資源的 [TemplateURL](#) 屬性。CloudFormation 版本不會採用 從 [ApplicationLocationObject](#) 擷取應用程式 AWS Serverless Application Repository。

NotificationARNs

現有 Amazon SNS 主題的清單，其中會傳送堆疊事件的通知。

類型：清單

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::CloudFormation::Stack` 資源的 [NotificationARNs](#) 屬性。

Parameters

應用程式參數值。

類型：映射

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::CloudFormation::Stack` 資源的 [Parameters](#) 屬性。

Tags

指定要新增到此應用程式之標籤的映射（字串到字串）。金鑰和值僅限於英數字元。金鑰長度可以是 1 到 127 個 Unicode 字元，且字首不可為 `aws:`。值長度可以是 1 到 255 個 Unicode 字元。

類型：映射

必要：否

CloudFormation 相容性：此屬性類似於 `AWS::CloudFormation::Stack` 資源的 [Tags](#) 屬性。SAM 中的標籤屬性包含 Key : Value 對；在 CloudFormation 中包含標籤物件清單。建立堆疊時，SAM 會自動將 `lambda:createdBy: SAM` 標籤新增至此應用程式。此外，如果此應用程式來自 AWS Serverless Application Repository，則 SAM 也會自動新增兩個標籤 `serverlessrepo:applicationId: ApplicationId` 和 `serverlessrepo:semanticVersion: SemanticVersion`。

TimeoutInMinutes

CloudFormation 等待巢狀堆疊達到 `CREATE_COMPLETE` 狀態的時間長度，以分鐘為單位。預設值為無逾時。當 CloudFormation 偵測到巢狀堆疊已達到 `CREATE_COMPLETE` 狀態時，它會將巢狀堆疊資源標記為父堆疊 `CREATE_COMPLETE` 中，並繼續建立父堆疊。如果逾時期間在巢狀堆疊達到之前過期 `CREATE_COMPLETE`，會將巢狀堆疊 CloudFormation 標記為失敗，並復原巢狀堆疊和父堆疊。

類型：整數

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::CloudFormation::Stack` 資源的 [TimeoutInMinutes](#) 屬性。

傳回值

Ref

當將此資源的邏輯 ID 提供給 `Ref` 內部函數時，它會傳回基礎 `AWS::CloudFormation::Stack` 資源的資源名稱。

如需使用 `Ref` 函數的詳細資訊，請參閱 AWS CloudFormation 《使用者指南[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。

如需使用 的詳細資訊Fn::GetAtt，請參閱AWS CloudFormation 《使用者指南[Fn::GetAtt](#)》中的。

Outputs.ApplicationOutputName

名稱為 的堆疊輸出值*ApplicationOutputName*。

範例

SAR 應用程式

使用 Serverless Application Repository 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
```

一般應用程式

S3 url 的應用程式

YAML

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

ApplicationLocationObject

已發佈至 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
```

Properties

ApplicationId

應用程式的 Amazon Resource Name (ARN)。

類型：字串

必要：是

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

SemanticVersion

應用程式的語義版本。

類型：字串

必要：是

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

範例

my-application

應用程式位置物件範例

YAML

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

SemanticVersion: 1.0.0

AWS::Serverless::CapacityProvider

為 AWS Lambda 事件函數建立容量提供者，以便在客戶擁有的 Amazon Elastic Compute Cloud 執行個體上執行 Lambda 受管執行個體。此資源是 Lambda 受管執行個體功能的一部分，可透過使用 Amazon EC2 定價模型，為大規模 Lambda 工作負載提供成本最佳化。

容量提供者會管理 Amazon EC2 執行個體的生命週期，並為 Lambda 函數提供必要的基礎設施，以在客戶擁有的運算資源上執行，同時維護無伺服器程式設計模型。

Note

當您部署到時 AWS CloudFormation，會將您的 AWS SAM 資源 `AWS::Serverless::CapacityProvider` 轉換為 `AmazonEC2::CapacityProvider` 資源。如需詳細資訊，請參閱 [為產生 CloudFormation 資源 AWS SAM](#)。

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::CapacityProvider
Properties:
  CapacityProviderName: String
  VpcConfig: VpcConfig
  OperatorRole: String
  Tags: Map
  PropagateTags: Boolean
  InstanceRequirements: InstanceRequirements
  ScalingConfig: ScalingConfig
  LoggingConfig: CapacityProviderLoggingConfig
  KmsKeyArn: String
  ManagedResourceTags: ManagedResourceTags
```

Properties

CapacityProviderName

容量提供者的名稱。此名稱在 AWS 您的帳戶和區域中必須是唯一的。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::CapacityProvider` 資源的 [CapacityProviderName](#) 屬性。

VpcConfig

容量提供者的 VPC 組態。指定要啟動 Amazon EC2 執行個體的 VPC 子網路和安全群組。

類型：[VpcConfig](#)

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::CapacityProvider` 資源的 [VpcConfig](#) 屬性。

OperatorRole

Lambda 運算子角色的 ARN，具有在客戶帳戶中建立和管理 Amazon EC2 執行個體和相關資源的許可。如果未提供，AWS SAM 會自動產生具有必要許可的角色。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::CapacityProvider` 資源 [PermissionsConfig](#) 的 [CapacityProviderOperatorRoleArn](#) 屬性。

Tags

要套用至容量提供者及其相關聯資源的鍵/值對映射。

類型：映射

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::CapacityProvider` 資源的 [Tags](#) 屬性。中的 Tags 屬性 AWS SAM 包含鍵值對（而 CloudFormation 此屬性包含標籤物件清單）。此外，AWS SAM 會自動將 `lambda:createdBy: SAM` 標籤新增至此 Lambda 函數，以及新增至為此函數產生的預設角色。

PropagateTags

指出是否要將標籤從標籤屬性傳遞至您AWS::Serverless::CapacityProvider產生的資源。將此設為 True以在產生的資源中傳播標籤。

類型：布林值

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

InstanceRequirements

容量提供者可以使用之運算執行個體類型的規格。這包括架構需求和 allowed或 excluded 執行個體類型。

類型：[InstanceRequirements](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Lambda::CapacityProvider 資源的 [InstanceRequirements](#) 屬性。

ScalingConfig

容量提供者的擴展組態。定義容量提供者如何根據需求擴展 Amazon EC2 執行個體。

類型：[ScalingConfig](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Lambda::CapacityProvider 資源的 [CapacityProviderScalingConfig](#) 屬性。

LoggingConfig

使用 LoggingConfig 設定容量提供者的系統記錄。系統日誌會擷取擴展活動和操作事件，並以可設定的日誌層級將其傳送至 Amazon CloudWatch 日誌群組。

類型：[CapacityProviderLoggingConfig](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Lambda::CapacityProvider 資源 [TelemetryConfig](#) 的 [LoggingConfig](#) 屬性。

KmsKeyArn

用於加密容量提供者靜態和傳輸中資料的 AWS KMS 金鑰 ARN。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::CapacityProvider` 資源的 [KmsKeyArn](#) 屬性。

ManagedResourceTags

將明確標籤套用至容量提供者代您佈建的 Amazon Elastic Compute Cloud 執行個體和其他受管資源。

類型：[ManagedResourceTags](#)

必要：否

CloudFormation compatibility：AWS SAM 使用此屬性來建構 `AWS::Lambda::CapacityProvider` 資源的 [PropagateTags](#) 屬性。

傳回值

Ref

當將此資源的邏輯 ID 提供給 `Ref` 內部函數時，它會傳回容量提供者的名稱。

如需使用 `Ref` 函數的詳細資訊，請參閱 AWS CloudFormation 《使用者指南[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。

如需使用的詳細資訊 `Fn::GetAtt`，請參閱 AWS CloudFormation 《使用者指南[Fn::GetAtt](#)》中的。

Arn

容量提供者的 ARN。

範例

基本容量提供者

下列範例會使用 VPC 組態建立基本容量提供者。

```
MyCapacityProvider:
  Type: AWS::Serverless::CapacityProvider
  Properties:
    CapacityProviderName: my-capacity-provider
    VpcConfig:
      SubnetIds:
        - subnet-12345678
        - subnet-87654321
      SecurityGroupIds:
        - sg-12345678
    Tags:
      Environment: Production
      Team: ServerlessTeam
```

具有擴展功能的進階容量提供者

下列範例會建立具有自訂執行個體需求和擴展組態的容量提供者。

```
AdvancedCapacityProvider:
  Type: AWS::Serverless::CapacityProvider
  Properties:
    CapacityProviderName: advanced-capacity-provider
    VpcConfig:
      SubnetIds:
        - subnet-12345678
        - subnet-87654321
      SecurityGroupIds:
        - sg-12345678
    OperatorRole: arn:aws:iam::123456789012:role/MyCapacityProviderRole
    PropagateTags: true
    InstanceRequirements:
      Architectures:
        - x86_64
      ExcludedTypes:
        - t2.micro
    ScalingConfig:
      MaxVCpuCount: 10
```

```
AverageCPUUtilization: 70.0
KmsKeyArn: arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012
LoggingConfig:
  LogGroup: /aws/lambda/capacity-provider/advanced-capacity-provider
  SystemLogLevel: INFO
ManagedResourceTags:
  Tags:
    CostCenter: cc-1234
    Environment: Production
Tags:
  Environment: Production
  CostCenter: Engineering
```

VpcConfig

設定容量提供者的 VPC 設定，包括將啟動 EC2 執行個體的子網路和安全群組。

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

```
SubnetIds: List
SecurityGroupIds: List
```

Properties

SubnetIds

將啟動 EC2 執行個體IDs 清單。至少必須指定一個子網路。

類型：清單

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Lambda::CapacityProvider 資源 [VpcConfig](#) 的 [SubnetIds](#) 屬性。

SecurityGroupIds

要與 EC2 執行個體建立關聯的安全群組 IDs 清單。如果未指定，則會使用 VPC 的預設安全群組。

類型：清單

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::CapacityProvider` 資源 `SecurityGroupIds` 的 `VpcConfig` 屬性。

範例

VPC 組態

下列範例顯示具有多個子網路和安全群組的 VPC 組態。

```
VpcConfig:
  SubnetIds:
    - subnet-12345678
    - subnet-87654321
  SecurityGroupIds:
    - sg-12345678
    - sg-87654321
```

InstanceRequirements

指定容量提供者將啟動之 EC2 執行個體的需求，包括架構和執行個體類型限制。

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

```
Architectures: List
AllowedTypes: List
ExcludedTypes: List
```

Note

您可以選擇在定義容量提供者的執行個體需求 `ExcludedTypes` 時指定 `AllowedTypes` 或，但不能同時指定兩者。

Properties

Architectures

容量提供者執行個體的指示集架構。

有效值：x86_64 或 arm64

類型：清單

必要：否

預設：x86_64

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::CapacityProvider` 資源 [InstanceRequirements](#) 的 [Architectures](#) 屬性。

AllowedTypes

容量提供者執行個體允許的 EC2 執行個體類型清單。

類型：清單

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::CapacityProvider` 資源 [InstanceRequirements](#) 的 [AllowedInstanceTypes](#) 屬性。

ExcludedTypes

要從容量提供者排除的 EC2 執行個體類型清單。

類型：清單

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::CapacityProvider` 資源 [InstanceRequirements](#) 的 [ExcludedInstanceTypes](#) 屬性。

範例

執行個體需求組態

下列範例顯示具有特定架構和執行個體類型限制條件的執行個體需求。

```
InstanceRequirements:
  Architectures:
    - x86_64
  ExcludedTypes:
    - t2.micro
```

ScalingConfig

指定容量提供者如何根據需求擴展 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體。您可以設定容量提供者可佈建vCPUs 數量上限，以及觸發擴展的目標平均 CPU 使用率。

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

```
MaxVCpuCount: Integer
AverageCPUUtilization: Double
```

Properties

MaxVCpuCount

容量提供者可在所有運算執行個體中佈建vCPUs 數量上限。

類型：整數

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::CapacityProvider` 資源 [CapacityProviderScalingConfig](#) 的 [MaxVCpuCount](#) 屬性。

AverageCPUUtilization

擴展決策的目標平均 CPU 使用率百分比 (0-100)。當平均 CPU 使用率超過此閾值時，容量提供者將擴展 Amazon EC2 執行個體。指定時，[CapacityProviderScalingConfig](#) 會 AWS SAM 建構 `AWS::Lambda::CapacityProvider` 資源，並將 [ScalingMode](#) 設定為 'Manual'，並將 [ScalingPolicies](#) 設定為 `[[{PredefinedMetricType: 'LambdaCapacityProviderAverageCPUUtilization', TargetValue: <this value>}]]`。

類型：Double

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

範例

擴展組態

下列範例顯示具有最大 VCpu 計數和平均 CPU 使用率的擴展組態。

```
ScalingConfig:
  MaxVCpuCount: 10
  AverageCPUUtilization: 70.0
```

ManagedResourceTags

將明確標籤套用至容量提供者代您佈建和管理的 Amazon Elastic Compute Cloud 執行個體和其他資源。

Note

ManagedResourceTags 屬性可控制 Amazon EC2 執行個體的標記，以及 Lambda 為容量提供者管理的其他資源。此屬性與 [PropagateTags](#) 屬性不同，指定 [Tags](#) 屬性是否 AWS SAM 傳播至從範本 AWS SAM 產生的 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

```
Tags: Map
```

Properties

支援下列屬性ManagedResourceTags。

Tags

要套用至 Amazon EC2 執行個體和容量提供者管理之其他資源的金鑰/值對映射。

類型：映射

必要：否

CloudFormation compatibility：AWS SAM 使用此屬性來建構

AWS::Lambda::CapacityProvider resource. AWS SAM sets 的 [PropagateTags](#) 屬性 Mode, Explicit 並將鍵值對轉換為 中的標籤物件清單 ExplicitTags。

範例

將明確標籤套用至受管資源

下列範例會將一組明確的標籤套用至容量提供者管理的 Amazon EC2 執行個體。

```
ManagedResourceTags:
  Tags:
    CostCenter: cc-1234
    Environment: Production
```

AWS::Serverless::Connector

設定兩個資源之間的許可。如需連接器的簡介，請參閱 [使用 AWS SAM 連接器管理資源許可](#)。

如需所產生 AWS CloudFormation 資源的詳細資訊，請參閱 [CloudFormation 當您指定 時產生的資源 AWS::Serverless::Connector](#)。

若要提供連接器的意見回饋，請在 serverless-application-model AWS GitHub 儲存庫 [提交新問題](#)。

Note

當您部署到 時 AWS CloudFormation，會將您的 AWS SAM 資源 AWS SAM 轉換為 CloudFormation 資源。如需詳細資訊，請參閱 [為 產生 CloudFormation 資源 AWS SAM](#)。

語法

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

 Note

對於大多數使用案例，我們建議使用內嵌連接器語法。內嵌在來源資源中可讓您隨著時間更輕鬆地讀取和維護。當您需要參考不在相同 AWS SAM 範本內的來源資源時，例如巢狀堆疊中的資源或共用資源，請使用 `AWS::Serverless::Connector` 語法。

內嵌連接器

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

AWS::Serverless::Connector

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

Properties

Destination

目的地資源。

類型：[ResourceReference](#) | [ResourceReference](#) 的清單

必要：是

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

Permissions

允許來源資源在目的地資源上執行的許可類型。

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

Write 包含允許啟動資料並將其寫入資源的 IAM 動作。

有效值：Read 或 Write

類型：清單

必要：是

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

Source

來源資源。使用 `AWS::Serverless::Connector` 語法時為必要。

類型：[ResourceReference](#)

必要：有條件

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

SourceReference

來源資源。

Note

定義來源資源的其他屬性時，請使用 搭配內嵌連接器語法。

類型：[SourceReference](#)

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

範例

內嵌連接器

下列範例使用內嵌連接器來定義 AWS Lambda 事件函數和 Amazon DynamoDB 資料表之間的 Write 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 許可：

```
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

下列範例使用 [AWS::Serverless::Connector](#) 資源從 讀取和寫入 Amazon DynamoDB 資料表的 AWS Lambda 函數：

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

下列範例使用 [AWS::Serverless::Connector](#) 資源將 Lambda 函數寫入 Amazon SNS 主題，而這兩個資源位於相同的範本中：

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

- Write

下列範例使用 [AWS::Serverless::Connector](#) 資源將 Amazon SNS 主題寫入 Lambda 函數，然後寫入 Amazon DynamoDB 資料表，其中包含相同範本中的所有資源：

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

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

  Table:
    Type: AWS::Serverless::SimpleTable

  TopicToFunctionConnector:
    Type: AWS::Serverless::Connector
    Properties:
      Source:
        Id: Topic
```

```

Destination:
  Id: Function
Permissions:
  - Write

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

```

以下是上述範例中的轉換 AWS CloudFormation 範本：

```

"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",

```


 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
```

Properties

Arn

資源的 ARN。

類型：字串

必要：有條件

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

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 推斷與這些政策相關聯的 IAM 角色Id。Id 未指定 時，請為連接器提供 RoleName 資源，以將產生的 IAM 政策連接至 IAM 角色。

類型：字串

必要：有條件

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

Name

資源的名稱。

類型：字串

必要：有條件

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

Qualifier

縮小其範圍的資源限定詞。會Qualifier取代資源限制條件 ARN 結尾*的值。如需範例，請參閱 [API Gateway 叫用 Lambda 事件函數](#)。

 Note

限定詞定義因資源類型而異。如需支援的來源和目的地資源類型清單，請參閱 [AWS SAM 連接器參考](#)。

類型：字串

必要：有條件

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

QueueUrl

Amazon SQS 佇列 URL。此屬性僅適用於 Amazon SQS 資源。

類型：字串

必要：有條件

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

ResourceId

資源的 ID。例如，API Gateway API ID。

類型：字串

必要：有條件

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

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 推斷與這些政策相關聯的 IAM 角色 Id。Id 未指定 時，請為連接器提供 RoleName 資源，以將產生的 IAM 政策連接至 IAM 角色。

類型：字串

必要：有條件

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

Type

資源的 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

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

範例

API Gateway 叫用 Lambda 事件函數

下列範例使用 [AWS::Serverless::Connector](#) 資源來允許 Amazon API Gateway 叫用 AWS Lambda 事件函數。

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[AWS::Serverless::Connector](#)類型使用的來源資源參考。

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

```
Qualifier: String
```

Properties

Qualifier

資源的限定詞縮小其範圍。會Qualifier取代資源限制條件 ARN 結尾*的值。

Note

限定詞定義因資源類型而異。如需支援的來源和目的地資源類型清單，請參閱 [AWS SAM 連接器參考](#)。

類型：字串

必要：有條件

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有同等 CloudFormation 屬性。

範例

下列範例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

`AWS::Serverless::Function` 資源會建立 Lambda 事件函數、(IAM) AWS Identity and Access Management 執行角色，以及觸發函數的事件來源映射。

Event Function 是原始 Lambda 函數類型的文件名稱，先前稱為 Lambda 函數。AWS::Serverless::Function 資源類型、產生的AWS::Lambda::Function資源及其屬性仍會使用 函數，因此您不需要變更範本。

[AWS::Serverless::Function](#) 資源也支援 Metadata 資源屬性，因此您可以指示 AWS SAM 建置應用程式所需的自訂執行時間。如需建置自訂執行時間的詳細資訊，請參閱 [在中使用自訂執行時間建置 Lambda 事件函數 AWS SAM](#)。

Note

當您部署到 時 AWS CloudFormation，會將您的 AWS SAM 資源 AWS SAM 轉換為 CloudFormation 資源。如需詳細資訊，請參閱 [為產生 CloudFormation 資源 AWS SAM](#)。

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
  CapacityProviderConfig: CapacityProviderConfig
  CodeSigningConfigArn: String
  CodeUri: String | FunctionCode

```

DeadLetterQueue: *Map* | [DeadLetterQueue](#)
DeploymentPreference: [DeploymentPreference](#)
Description: *String*
DurableConfig: [DurableConfig](#)
Environment: [Environment](#)
EphemeralStorage: [EphemeralStorage](#)
EventInvokeConfig: [EventInvokeConfiguration](#)
Events: [EventSource](#)
FileSystemConfigs: *List*
FunctionName: *String*
FunctionScalingConfig: [FunctionScalingConfig](#)
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*
PublishToLatestPublished: *Boolean*
PropagateTags: *Boolean*
ProvisionedConcurrencyConfig: [ProvisionedConcurrencyConfig](#)
RecursiveLoop: *String*
ReservedConcurrentExecutions: *Integer*
Role: *String*
RolePath: *String*
Runtime: *String*
RuntimeManagementConfig: [RuntimeManagementConfig](#)
SnapStart: [SnapStart](#)
SourceKMSKeyArn: *String*
Tags: *Map*
TenancyConfig: [TenancyConfig](#)
Timeout: *Integer*
Tracing: *String*
VersionDescription: *String*
VersionDeletionPolicy: *String*
VpcConfig: [VpcConfig](#)

Properties

Architectures

函數的指示集架構。

如需此屬性的詳細資訊，請參閱《AWS Lambda 開發人員指南》中的 [Lambda 指令集架構](#)。

有效值：x86_64 或 arm64

類型：清單

必要：否

預設：x86_64

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::Function` 資源的 [Architectures](#) 屬性。

AssumeRolePolicyDocument

為此 Role 函數建立的預設 新增 AssumeRolePolicyDocument。如果未指定此屬性，會為此函數 AWS SAM 新增預設擔任角色。

類型：JSON

必要：否

CloudFormation 相容性：此屬性類似於 `AWS::IAM::Role` 資源的 [AssumeRolePolicyDocument](#) 屬性。會將此屬性 AWS SAM 新增至此函數產生的 IAM 角色。如果此函數提供角色的 Amazon Resource Name (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](#)。如需使用此屬性的範例，請參閱 [使用 逐步部署無伺服器應用程式 AWS SAM](#)。

AWS SAM 會在設定此屬性時產生 [AWS::Lambda::Alias](#) [AWS::Lambda::Version](#) 和資源。如需此案例的資訊，請參閱 [已指定 AutoPublishAlias 屬性](#)。如需所產生 CloudFormation 資源的一般資訊，請參閱 [為 產生 CloudFormation 資源 AWS SAM](#)。

版本發佈和內部函數

AWS SAM 透過在部署之間比較範本中的屬性值，決定是否發佈新版本。對於透過內部函數提供的值，AWS SAM 會比較未解析的參考，而不是解析的值。其中一個範例是將 範本參數的環境變數設定為 `!Ref`。

如果部署僅變更堆疊參數的值，則範本本身不會變更。在此情況下，AWS SAM 不會發佈新版本或更新別名。AWS SAM 只會將變更套用至函數的 `$LATEST` 版本。如需解決方法，請參閱使用 [AWS :: LanguageExtensions](#) 的範例。

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

AutoPublishAliasAllProperties

指定何時建立新的 [AWS::Lambda::Version](#)。當時 `true`，會在修改 Lambda 函數中的任何屬性時建立新的 Lambda 版本。當時 `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 `CodeDict`、`ImageUri` 或 `InlineCode`。

參數變更和版本發佈

AWS SAM 只有在範本中的屬性值變更時，才會發佈新版本。如果屬性值來自內部函數，AWS SAM 請參閱未解析的參考。其中一個範例是 範本參數的 `!Ref` 所定義的環境變數。僅變更參數值的部署不會發佈新版本，即使使用 `AutoPublishAliasAllProperties: true`。`$LATEST` 只會將變更 AWS SAM 套用至。

當只有參數值變更時，使用下列其中一個解決方法 AWS SAM 強制 發佈新版本：

- 標準 AWS SAM 和 CloudFormation 部署：在範本的 `Transform` 區段 `AWS::Serverless-2016-10-31` 中新增 之前的 [AWS :: LanguageExtensions](#) 轉換。在 AWS SAM 運算版本雜湊之前，CloudFormation 解析對其實際值的參考。然後，參數值的變更會發佈新版本並更新別名：

```
Transform:
```

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

- AWS Serverless Application Repository 應用程式：僅 AWS Serverless Application Repository 支援AWS::Serverless-2016-10-31轉換，並拒絕宣告 的範本AWS::LanguageExtensions。對於您透過 發佈的應用程式 AWS Serverless Application Repository，請在範本中直接提供值作為常值，而不是參考參數。這會在 AWS SAM 運算版本雜湊時顯示變更。

此屬性AutoPublishAlias需要定義。

如果也指定 AutoPublishCodeSha256，則其行為優先於 AutoPublishAliasAllProperties: true。

類型：布林值

必要：否

預設值：false

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

AutoPublishCodeSha256

使用時，此字串會與 CodeUri值搭配使用，以判斷是否需要發佈新的 Lambda 版本。此屬性通常用於解決下列部署問題：部署套件會存放在 Amazon S3 位置，並以具有更新 Lambda 函數程式碼的新部署套件取代，但 CodeUri 屬性保持不變（而不是將新的部署套件上傳至新的 Amazon S3 位置，並將 CodeUri 變更為新的位置）。

此問題由具有下列特性的 AWS SAM 範本標記：

- DeploymentPreference 物件已設定為逐步部署（如中所述[使用 逐步部署無伺服器應用程式 AWS SAM](#)）
- AutoPublishAlias 屬性已設定，不會在部署之間變更
- CodeUri 屬性已設定，不會在部署之間變更。

在此案例中，更新AutoPublishCodeSha256會導致成功建立新的 Lambda 版本。不過，部署到 Amazon S3 的新函數程式碼將無法辨識。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。為您的 Lambda 函數指定 Version 屬性，並將儲存貯體設定為一律使用最新的部署套件。

在此案例中，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

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

CapacityProviderConfig

設定將附加發佈版本函數的容量提供者。這可讓函數在由 Lambda 受管執行個體管理的客戶擁有 EC2 執行個體上執行。

類型：[CapacityProviderConfig](#)

必要：否

CloudFormation 相容性：SAM 會扁平化傳遞至 `AWS::Lambda::Function` 資源 屬性的 [CapacityProviderConfig](#) 屬性，並重建巢狀結構。

CodeSigningConfigArn

[AWS::Lambda::CodeSigningConfig](#) 資源的 ARN，用於啟用此函數的程式碼簽署。如需有關程式碼簽署的詳細資訊，請參閱[為您的 AWS SAM 應用程式設定程式碼簽署](#)。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::Function` 資源的 [CodeSigningConfigArn](#) 屬性。

CodeUri

函數的程式碼。接受的值包括：

- 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，請使用在 AWS SAM CLI 部署時上傳本機檔案。如需詳細資訊，請參閱 [如何在部署時 AWS SAM 上傳本機檔案](#)。

如果您在 CodeUri 屬性中使用內部函數，AWS SAM 將無法正確剖析值。請考慮改用 [AWS::LanguageExtensions](#) 轉換。

類型：【字串 | [FunctionCode](#)】

必要：有條件限制。當 PackageType 設為 Zip，InlineCode 需要其中一個 CodeUri 或。

CloudFormation 相容性：此屬性類似於 AWS::Lambda::Function 資源的 [Code](#) 屬性。巢狀 Amazon S3 屬性的名稱不同。

DeadLetterQueue

設定 Amazon Simple Notification Service (Amazon SNS) 主題或 Amazon Simple Queue Service (Amazon SQS) 佇列，其中 Lambda 會傳送無法處理的事件。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](#)。

Note

如果您的 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](#)，而非事件來源佇列。

類型：映射 | [DeadLetterQueue](#)

必要：否

CloudFormation 相容性：此屬性類似於 AWS::Lambda::Function 資源的 [DeadLetterConfig](#) 屬性。在 CloudFormation 類型衍生自 TargetArn，而在 AWS SAM 中，您必須將類型與一起傳遞 TargetArn。

DeploymentPreference

啟用逐步 Lambda 部署的設定。

如果指定 DeploymentPreference 物件，會 AWS SAM 建立 [AWS::CodeDeploy::Application](#) 稱為的 ServerlessDeploymentApplication (每個堆疊一個) *<function-logical-id>*DeploymentGroup、[AWS::CodeDeploy::DeploymentGroup](#) 稱為的和 [AWS::IAM::Role](#) 稱為的 CodeDeployServiceRole。

類型：[DeploymentPreference](#)

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

另請參閱：如需此屬性的詳細資訊，請參閱[使用 逐步部署無伺服器應用程式 AWS SAM](#)。

Description

函數的敘述。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Lambda::Function 資源的 [Description](#) 屬性。

DurableConfig

耐用函數的組態。使用自動檢查點和重播功能來啟用有狀態執行。

類型：[DurableConfig](#)

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

Environment

執行時間環境的組態。

類型：[Environment](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Lambda::Function 資源的 [Environment](#) 屬性。

EphemeralStorage

指定 中 Lambda 函數可用磁碟空間的物件，以 MB 為單位/tmp。

如需此屬性的詳細資訊，請參閱《AWS Lambda 開發人員指南》中的 [Lambda 執行環境](#)。

類型：[EphemeralStorage](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::Function` 資源的 [EphemeralStorage](#) 屬性。

EventInvokeConfig

描述事件在 Lambda 函數上叫用組態的物件。

類型：[EventInvokeConfiguration](#)

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

Events

指定觸發此函數的事件。事件由類型和一組屬性組成，這些屬性取決於類型。

類型：[EventSource](#)

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

FileSystemConfigs

指定 Amazon Elastic File System (Amazon EFS) 檔案系統或 Amazon S3 檔案檔案系統的連線設定的 [FileSystemConfig](#) 物件清單。您可以連接 Amazon EFS 存取點或 S3 檔案存取點，但不能同時連接兩者。

每個 `FileSystemConfig` 物件都包含 `Arn` (存取點 ARN) 和 `LocalMountPath` (在函數中掛載檔案系統的路徑)。對於 Amazon EFS，ARN 是 Amazon EFS 存取點 ARN。對於 S3 檔案，ARN 是 `AWS::S3Files::AccessPoint` ARN。

如果您的範本包含 [AWS::EFS::MountTarget](#) 資源，您還必須指定 `DependsOn` 資源屬性，以確保掛載目標在函數之前已建立或更新。同樣地，如果您的範本包含 `AWS::S3Files::MountTarget` 資源，則必須指定 S3 檔案掛載目標的 `DependsOn` 屬性。

類型：清單

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::Function` 資源的 [FunctionName](#) 屬性。

FunctionScalingConfig

為在容量提供者上執行的 Lambda 函數設定擴展行為。定義執行環境的最小和最大數量。

類型：[FunctionScalingConfig](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::Function` 資源的 [FunctionScalingConfig](#) 屬性。

FunctionUrlConfig

描述函數 URL 的物件。函數 URL 是 HTTPS 端點，可用來叫用函數。

如需詳細資訊，請參閱《AWS Lambda 開發人員指南》中的[函數 URLs](#)。

類型：[FunctionUrlConfig](#)

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

Handler

您程式碼中的 函數，稱為 以開始執行。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

CloudFormation 相容性：此屬性會直接傳遞至 `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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::Function` 資源的 [ImageConfig](#) 屬性。

ImageUri

Lambda 函數容器映像的 Amazon Elastic Container Registry (Amazon ECR) 儲存庫的 URI。此屬性僅適用於 `PackageType` 屬性設為 `Image`，否則會予以忽略。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 `PackageType` 屬性設定為 `Image`，則需要 `ImageUri`，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[使用的預設建置 AWS SAM](#)。

使用必要Metadata項目建置應用程式優先於 `ImageUri`，因此如果您指定兩者，`ImageUri`則會忽略。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::Function` 資料類型的 [ImageUri](#) 屬性。

InlineCode

直接寫入範本中的 Lambda 函數程式碼。此屬性僅適用於 `PackageType` 屬性設為 `Zip`，否則會予以忽略。

 Note

如果 `PackageType` 屬性設定為 `Zip` (預設)，`InlineCode` 則需要其中一個 `CodeUri` 或。

類型：字串

必要：有條件

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::Function` Code 資料類型的 [ZipFile](#) 屬性。

KmsKeyArn

Lambda 用來加密和解密函數環境變數之 AWS Key Management Service (AWS KMS) 金鑰的 ARN。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::Function` 資源的 [KmsKeyArn](#) 屬性。

Layers

此函數應使用的 `LayerVersion` ARNs 清單。此處指定的順序是在執行 Lambda 函數時匯入的順序。版本是完整的 ARN，包括版本或 `LayerVersion` 資源的參考。例如，的參考 `LayerVersion` 將是 `!Ref MyLayer` 而包含版本的完整 ARN 將是 `arn:aws:lambda:region:account-id:layer:layer-name:version`。

類型：清單

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::Function` 資源的 [Layers](#) 屬性。

LoggingConfig

該功能的 Amazon CloudWatch Logs 組態設定。

類型：[LoggingConfig](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::Function` 資源的 [LoggingConfig](#) 屬性。

MemorySize

每次叫用函數時配置的記憶體大小，以 MB 為單位。

類型：整數

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `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. 如果此屬性設為 Zip (預設)，則會 `InlineCode` 套用 `CodeUri` 或，並忽略 `ImageUri`。
2. 如果此屬性設為 Image，則只會 `ImageUri` 套用，且 `InlineCode` 會忽略 `CodeUri` 和。儲存函數容器映像所需的 Amazon ECR 儲存庫可由自動建立 AWS SAM CLI。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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::Function` 資源的 [PackageType](#) 屬性。

PermissionsBoundary

用於此函數執行角色之許可界限的 ARN。此屬性只有在為您產生角色時才有效。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::IAM::Role` 資源的 [PermissionsBoundary](#) 屬性。

Policies

此函數的許可政策。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 政策範本](#)。
- [AWS 受管政策](#)或[客戶受管政策](#)ARN的。
- 下列[清單中](#)受 AWS 管政策的名稱。
- 在 中格式化YAML為映射的[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

CloudFormation 相容性：此屬性類似於 `AWS::IAM::Role` 資源的 [Policies](#) 屬性。

PublishToLatestPublished

指定是否要在更新函數時發佈最新的函數版本。

類型：布林值

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::Function` 資源的 [PublishToLatestPublished](#) 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。指定 True 在產生的資源中傳播標籤。

類型：布林值

必要：否

預設：False

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

ProvisionedConcurrencyConfig

函數別名的佈建並行組態。

Note

ProvisionedConcurrencyConfig 只有在設定 AutoPublishAlias 時，才能指定。否則會產生錯誤結果。

類型：[ProvisionedConcurrencyConfig](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Lambda::Alias 資源的 [ProvisionedConcurrencyConfig](#) 屬性。

RecursiveLoop

函數遞迴迴圈偵測組態的狀態。

當此值設為 Allow 且 Lambda 偵測到您的函數被調用為遞迴迴圈的一部分時，它不會採取任何動作。

當此值設為 Terminate 且 Lambda 偵測到您的函數被調用為遞迴迴圈的一部分時，它會停止您的函數被調用並通知您。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Lambda::Function 資源的 [RecursiveLoop](#) 屬性。

ReservedConcurrentExecutions

您要保留給函數的並行執行數目上限。

如需此屬性的詳細資訊，請參閱《AWS Lambda 開發人員指南》中的 [Lambda 函數擴展](#)。

類型：整數

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::Function` 資源的 [ReservedConcurrentExecutions](#) 屬性。

Role

做為此函數執行角色的 IAM 角色 ARN。

類型：字串

必要：否

CloudFormation 相容性：此屬性類似於 `AWS::Lambda::Function` 資源的 [Role](#) 屬性。這是在 CloudFormation 的必要項目，但不是在 AWS SAM。如果未指定角色，則會為您建立一個邏輯 ID 為 `<function-logical-id>Role` 的角色。

RolePath

函數 IAM 執行角色的路徑。

為您產生角色時，請使用此屬性。使用 `Role` 屬性指定角色時，請勿使用。

類型：字串

必要：有條件

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::IAM::Role` 資源的 [Path](#) 屬性。

Runtime

函數的 [執行時間](#) 的識別符。只有在 `PackageType` 屬性設定為 `Zip` 時，才需要此屬性。

Note

如果您為此屬性指定 `provided` 識別符，您可以使用 `Metadata` 資源屬性來指示 AWS SAM 建置此函數所需的自訂執行期。如需建置自訂執行時間的詳細資訊，請參閱 [在中使用自訂執行時間建置 Lambda 事件函數 AWS SAM](#)。

類型：字串

必要：有條件

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::Function` 資源的 [Runtime](#) 屬性。

RuntimeManagementConfig

為您的 Lambda 函數設定執行時間管理選項，例如執行時間環境更新、復原行為，以及選取特定的執行時間版本。若要進一步了解，請參閱《AWS Lambda 開發人員指南》中的 [Lambda 執行期更新](#)。

類型：[RuntimeManagementConfig](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `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快取狀態，包括其所有相依性。函數只會初始化一次，快取狀態會重複使用於所有未來的調用，藉由減少必須初始化函數的次數來改善應用程式效能。若要進一步了解，請參閱《AWS Lambda 開發人員指南》中的 [使用 Lambda SnapStart 改善啟動效能](#)。

類型：[SnapStart](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::Function` 資源的 [SnapStart](#) 屬性。

SourceKMSKeyArn

代表用來加密客戶 ZIP 函數程式碼的 KMS 金鑰 ARN。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::FunctionCode` 資料類型的 [SourceKMSKeyArn](#) 屬性。

Tags

指定新增至此函數之標籤的映射（字串到字串）。如需標籤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，以及此函數產生的預設角色。

類型：映射

必要：否

CloudFormation 相容性：此屬性類似於 `AWS::Lambda::Function` 資源的 [Tags](#) 屬性。中的 Tags 屬性 AWS SAM 包含鍵值對（而 CloudFormation 此屬性包含 Tag 物件清單）。此外，AWS SAM 會自動將`lambda:createdBy:SAM`標籤新增至此 Lambda 函數，以及新增至為此函數產生的預設角色。

TenancyConfig

Lambda 租用戶隔離模式的組態。確保執行環境永遠不會在不同租用戶 IDs 之間共用，為多租用戶應用程式提供運算層級隔離。

類型：[TenancyConfig](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::Function` 資源的 [TenancyConfig](#) 屬性。

Timeout

函數在停止之前可以執行的時間上限，以秒為單位。

類型：整數

必要：否

預設：3

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::Function` 資源的 [Timeout](#) 屬性。

Tracing

指定函數 X-Ray 追蹤模式的字串。

- Active – 啟用函數的 X-Ray 追蹤。
- Disabled – 停用函數的 X-Ray。
- PassThrough – 啟用函數的 X-Ray 追蹤。抽樣決策會委派給下游服務。

如果指定為 Active 或 PassThrough 且未設定 Role 屬性，會將 `arn:aws:iam::aws:policy/AWSXrayWriteOnlyAccess` 政策 AWS SAM 新增至其為您建立的 Lambda 執行角色。

如需 X-Ray 的詳細資訊，請參閱《AWS Lambda 開發人員指南》中的[使用 AWS Lambda 搭配 AWS X-Ray](#)。

有效值：【Active|Disabled|PassThrough】

類型：字串

必要：否

CloudFormation 相容性：此屬性類似於 `AWS::Lambda::Function` 資源的 [TracingConfig](#) 屬性。

VersionDescription

指定在新的 Lambda 版本資源上新增 Description 的欄位。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::Version` 資源的 [Description](#) 屬性。

VersionDeletionPolicy

指定設定時所建立 Lambda AutoPublishAlias 版本資源的刪除政策。這會控制在刪除堆疊時保留或刪除版本資源。

有效值：Delete、Retain 或 Snapshot

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。它會在產生的 `AWS::Lambda::Version` 資源上設定 DeletionPolicy 屬性。

VpcConfig

可讓此函數存取虛擬私有雲端 (VPC) 內私有資源的組態。

類型：[VpcConfig](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::Function` 資源的 [VpcConfig](#) 屬性。

傳回值

Ref

當將此資源的邏輯 ID 提供給 `Ref` 內部函數時，它會傳回基礎 Lambda 函數的資源名稱。

如需使用 `Ref` 函數的詳細資訊，請參閱 AWS CloudFormation 《使用者指南[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。

如需使用的詳細資訊 `Fn::GetAtt`，請參閱 AWS CloudFormation 《使用者指南[Fn::GetAtt](#)》中的。

Arn

基礎 Lambda 函數的 ARN。

範例

簡單函數

以下是 Amazon S3 儲存貯體中套件類型 `Zip` (預設) 和函數程式碼 [AWS::Serverless::Function](#) 資源的基本範例。

YAML

```
Type: AWS::Serverless::Function
Properties:
```

```

Handler: index.handler
Runtime: python3.9
CodeUri: s3://bucket-name/key-name

```

函數屬性範例

以下是使用 InlineCode、Layers、Policies Amazon EFS 檔案系統和 Api 事件來源 [AWS::Serverless::Function](#) 的套件類型 Zip (預設) Tracing 的範例。

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:::sam-s3-demo-bucket/*'
  Events:
    ApiEvent:
      Type: Api
      Properties:
        Path: /path
        Method: get

```

ImageConfig 範例

以下是套件類型 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 範例

設定為根據目前行為更新其執行時間環境的 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時間環境的 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時間環境的 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 範例

為未來版本開啟 SnapStart 的 Lambda 函數範例：

```
TestFunc
  Type: AWS::Serverless::Function
  Properties:
    ...
    SnapStart:
      ApplyOn: PublishedVersions
```

TenancyConfig 範例

開啟租用戶隔離模式的 Lambda 函數範例：

```
TestFunction
  Type: AWS::Serverless::Function
  Properties:
    ...
    TenancyConfig:
      TenantIsolationMode: PER_TENANT
```

S3 檔案系統範例

下列範例會建立掛載 Amazon S3 Files 檔案系統的 Lambda 函數。範本會建立 S3 儲存貯體、該儲存貯體支援的 S3 檔案系統、VPC 子網路中的掛載目標，以及存取點。函數會將存取點掛載在 `/mnt/s3files`，並且可以讀取和寫入與 S3 儲存貯體同步的檔案。

Note

您可以將 Amazon EFS 檔案系統或 S3 檔案系統連接至 Lambda 函數，但不能同時連接兩者。

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: Lambda function with S3 Files file system

Resources:
  # VPC and networking
  MyVpc:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: true
      EnableDnsHostnames: true

  MySubnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref MyVpc
      CidrBlock: 10.0.1.0/24
      AvailabilityZone: !Select [0, !GetAZs '']

  MySecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Security group for Lambda and S3 Files
      VpcId: !Ref MyVpc
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 2049
          ToPort: 2049
          CidrIp: 10.0.0.0/16

  # S3 bucket for file storage
  MyS3Bucket:
    Type: AWS::S3::Bucket

  # IAM role for S3 Files to access the bucket
  S3FilesRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
```

```

    - Effect: Allow
    Principal:
      Service: elasticfilesystem.amazonaws.com
    Action: sts:AssumeRole
Policies:
  - PolicyName: S3FilesBucketAccess
    PolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Effect: Allow
          Action:
            - s3:GetObject
            - s3:PutObject
            - s3:DeleteObject
            - s3:ListBucket
          Resource:
            - !GetAtt MyS3Bucket.Arn
            - !Sub '${MyS3Bucket.Arn}/*'

```

```
# S3 Files resources
```

```
MyS3FilesFileSystem:
```

```
  Type: AWS::S3Files::FileSystem
```

```
  Properties:
```

```
    Bucket: !GetAtt MyS3Bucket.Arn
```

```
    RoleArn: !GetAtt S3FilesRole.Arn
```

```
MyS3FilesMountTarget:
```

```
  Type: AWS::S3Files::MountTarget
```

```
  Properties:
```

```
    FileSystemId: !Ref MyS3FilesFileSystem
```

```
    SubnetId: !Ref MySubnet
```

```
    SecurityGroups:
```

```
      - !Ref MySecurityGroup
```

```
MyS3FilesAccessPoint:
```

```
  Type: AWS::S3Files::AccessPoint
```

```
  Properties:
```

```
    FileSystemId: !Ref MyS3FilesFileSystem
```

```
    PosixUser:
```

```
      Uid: '1000'
```

```
      Gid: '1000'
```

```
    RootDirectory:
```

```
      Path: /lambda
```

```
    CreationInfo:
```

```
OwnerUid: '1000'
OwnerGid: '1000'
Permissions: '750'
```

```
# Lambda function with S3 Files mount
```

```
MyFunction:
```

```
  Type: AWS::Serverless::Function
```

```
  DependsOn: MyS3FilesMountTarget
```

```
  Properties:
```

```
    Handler: index.handler
```

```
    Runtime: python3.12
```

```
    Timeout: 120
```

```
    VpcConfig:
```

```
      SecurityGroupIds:
```

```
        - !Ref MySecurityGroup
```

```
      SubnetIds:
```

```
        - !Ref MySubnet
```

```
    FileSystemConfigs:
```

```
      - Arn: !GetAtt MyS3FilesAccessPoint.AccessPointArn
```

```
        LocalMountPath: /mnt/s3files
```

```
    Policies:
```

```
      - Version: '2012-10-17'
```

```
        Statement:
```

```
          - Effect: Allow
```

```
            Action:
```

```
              - s3files:ClientMount
```

```
              - s3files:ClientWrite
```

```
            Resource: !GetAtt MyS3FilesAccessPoint.AccessPointArn
```

```
  InlineCode: |
```

```
    import os
```

```
    def handler(event, context):
```

```
      # Write a file to the S3 Files mount
```

```
      with open('/mnt/s3files/hello.txt', 'w') as f:
```

```
        f.write('Hello from Lambda!')
```

```
      # List files at the mount path
```

```
      files = os.listdir('/mnt/s3files')
```

```
      return {'files': files}
```


DeadLetterQueue

指定 SQS 佇列或 SNS 主題，當事件無法處理時，AWS Lambda (Lambda) 會將事件傳送至該主題。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 會自動為您的 Lambda 事件函數執行角色新增適當的許可，讓 Lambda 服務存取資源。sqs : SendMessage 會針對 SQS 佇列新增，sns : Publish for SNS 主題新增。

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
```

Properties

TargetArn

Amazon SQS 佇列或 Amazon SNS 主題的 Amazon Resource Name (ARN)。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Lambda::Function DeadLetterConfig 資料類型的 [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

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

範例

DeadLetterQueue

SNS 主題的無效字母佇列範例。

YAML

```
DeadLetterQueue:
  Type: SNS
  TargetArn: arn:aws:sns:us-east-2:123456789012:my-topic
```

DeploymentPreference

指定要啟用逐步 Lambda 部署的組態。如需設定逐步 Lambda 部署的詳細資訊，請參閱 [使用 逐步部署 無伺服器應用程式 AWS SAM](#)。

Note

您必須在 AutoPublishAlias 中指定 [AWS::Serverless::Function](#) 以使用 DeploymentPreference 物件，否則會產生錯誤。

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
```

Properties

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` 內部函數。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

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

Enabled

此部署偏好設定是否已啟用。

類型：布林值

必要：否

預設：True

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

Hooks

驗證流量轉移前後執行的 Lambda 事件函數。

類型：[勾點](#)

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

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產生的 CodeDeploy 資源。一般而言，您應該將此設定為 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

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

Role

CodeDeploy 將用於流量轉移的 IAM 角色 ARN。如果提供此功能，則不會建立 IAM 角色。

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

TriggerConfigurations

您要與部署群組建立關聯的觸發組態清單。用來通知生命週期事件的 SNS 主題。

類型：清單

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `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 [使用 逐步部署無伺服器應用程式 AWS SAM](#)。

類型：字串

必要：是

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

範例

具有流量前後掛鉤的 DeploymentPreference。

包含流量前後掛鉤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
```

具有 Fn::If 內部函數的 DeploymentPreference

Fn::If 用於設定警示的範例部署偏好設定。在此範例中，如果 MyCondition 是 true，Alarm1 則會設定 true，如果 MyCondition 是 false，Alarm2Alarm5 則會設定 false。

YAML

```
DeploymentPreference:
  Enabled: true
  Type: Canary10Percent10Minutes
  Alarms:
    Fn::If:
      - MyCondition
      - - Alarm1
        - Alarm2
        - Alarm5
```

Hooks

驗證流量轉移前後執行的 Lambda 事件函數。

Note

此屬性中參考的 Lambda 函數會設定產生 [AWS::Lambda::Alias](#) 資源的 CodeDeployLambdaAliasUpdate 物件。如需詳細資訊，請參閱 AWS CloudFormation 《使用者指南》中的 [CodeDeployLambdaAliasUpdate 政策](#)。

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
```

Properties

PostTraffic

在流量轉移後執行的 Lambda 函數。

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

PreTraffic

在流量轉移之前執行的 Lambda 函數。

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

範例

勾點

掛鉤函數範例

YAML

```
Hooks:
  PreTraffic:
    Ref: PreTrafficLambdaFunction
  PostTraffic:
    Ref: PostTrafficLambdaFunction
```

DurableConfig

設定 AWS Lambda 函數的耐久執行設定。耐用的函數可以執行長達一年，並自動檢查點進度，啟用長時間執行的工作流程和容錯應用程式。如需耐用函數的詳細資訊，請參閱《AWS Lambda 開發人員指南》中的 [Lambda 耐用函數](#)。

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

```
ExecutionTimeout: Integer
RetentionPeriodInDays: Integer
```

Properties

ExecutionTimeout

Lambda 允許耐用函數在停止之前執行的時間量（以秒為單位）。上限為一年 366 天或 31,622,400 秒。

類型：整數

必要：是

下限：1

上限：31622400

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::Function DurableConfig` 資料類型的 [ExecutionTimeout](#) 屬性。

RetentionPeriodInDays

持久性執行關閉後的天數，Lambda 會保留其歷史記錄，從 1 天到 90 天。預設值是 14 天。

類型：整數

必要：否

預設：14

下限：1

上限：90

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::Function DurableConfig` 資料類型的 [RetentionPeriodInDays](#) 屬性。

範例

DurableConfig

具有 1 小時執行逾時和 7 天保留期的函數的耐用組態範例。

YAML

```
DurableConfig:
  ExecutionTimeout: 3600
```

```
RetentionPeriodInDays: 7
```

EventInvokeConfiguration

[非同步](#) Lambda 別名或版本調用的組態選項。

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
```

Properties

DestinationConfig

組態物件，指定在 Lambda 處理過後事件的目標。

類型：[EventInvokeDestinationConfiguration](#)

必要：否

CloudFormation 相容性：此屬性類似於 `AWS::Lambda::EventInvokeConfig` 資源的 [DestinationConfig](#) 屬性。SAM 需要不在 CloudFormation 中的額外參數 "Type"。

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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventInvokeConfig` 資源的 [MaximumEventAgeInSeconds](#) 屬性。

MaximumRetryAttempts

在函數傳回錯誤之前重試的次數上限。

類型：整數

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventInvokeConfig` 資源的 [MaximumRetryAttempts](#) 屬性。

範例

MaximumEventAgeInSeconds

MaximumEventAgeInSeconds 範例

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
```

Properties

OnFailure

處理失敗事件的目標。

類型：[OnFailure](#)

必要：否

CloudFormation 相容性：此屬性類似於 `AWS::Lambda::EventInvokeConfig` 資源的 [OnFailure](#) 屬性。需要 `Type`，這是額外的僅限 SAM 屬性。

OnSuccess

已順利處理的事件目標。

類型：[OnSuccess](#)

必要：否

CloudFormation 相容性：此屬性類似於 `AWS::Lambda::EventInvokeConfig` 資源的 [OnSuccess](#) 屬性。需要 `Type`，這是額外的僅限 SAM 屬性。

範例

OnSuccess

OnSuccess 範例

YAML

```
EventInvokeConfig:
  DestinationConfig:
    OnSuccess:
      Type: SQS
      Destination: arn:aws:sqs:us-west-2:012345678901:my-queue
    OnFailure:
      Type: Lambda
      Destination: !GetAtt DestinationLambda.Arn
```

OnFailure

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
```

Properties

Destination

目標資源的 Amazon Resource Name (ARN)。

類型：字串

必要：有條件

CloudFormation 相容性：此屬性類似於 `AWS::Lambda::EventInvokeConfig` 資源的 [OnFailure](#) 屬性。SAM 會將任何必要的許可新增至與此函數相關聯的自動產生 IAM 角色，以存取此屬性中參考的資源。

其他備註：如果類型為 `Lambda/EventBridge`，則需要目的地。

Type

目的地中參考的資源類型。支援的類型為 `SQS`、`SNS`、`Lambda`、`S3` 和 `EventBridge`。

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

其他備註：如果類型為 `SQS/SNS` 且 `Destination` 屬性為空白，則 SAM 會自動產生 `SQS/SNS` 資源。若要參考資源，請 `<function-logical-id>.DestinationQueue` 針對 `SQS` 使用，或 `<function-logical-id>.DestinationTopic` 針對 `SNS` 使用。如果類型為 `Lambda/EventBridge`，`Destination` 則為必要項目。

範例

使用 SQS 和 Lambda 目的地的 EventInvoke 組態範例

在此範例中，`SQS OnSuccess` 組態不會指定目的地，因此 SAM 會隱含地建立 `SQS` 佇列並新增任何必要的許可。此外，在此範例中，範本檔案中宣告的 `Lambda` 資源目的地是在 `OnFailure` 組態中指定，因此 SAM 會將必要的許可新增至此 `Lambda` 事件函數，以呼叫目的地 `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.
```

使用 SNS 目的地的 EventInvoke 組態範例

在此範例中，會為 OnSuccess 組態的範本檔案中宣告的 SNS 主題指定目的地。

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
```

Properties

Destination

目標資源的 Amazon Resource Name (ARN)。

類型：字串

必要：有條件

CloudFormation 相容性：此屬性類似於 `AWS::Lambda::EventInvokeConfig` 資源的 [OnSuccess](#) 屬性。SAM 會將任何必要的許可新增至與此函數相關聯的自動產生 IAM 角色，以存取此屬性中參考的資源。

其他備註：如果類型為 `Lambda/EventBridge`，則需要目的地。

Type

目的地中參考的資源類型。支援的類型為 `SQS`、`SNS`、`Lambda`、`S3`和 `EventBridge`。

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

其他備註：如果類型為 `SQS/SNS` 且 `Destination` 屬性為空白，則 SAM 會自動產生 `SQS/SNS` 資源。若要參考 資源，請 `<function-logical-id>.DestinationQueue` 針對 `SQS` 使用，或 `<function-logical-id>.DestinationTopic` 針對 `SNS` 使用。如果類型為 `Lambda/EventBridge`，`Destination` 則為必要項目。

範例

使用 `SQS` 和 `Lambda` 目的地的 `EventInvoke` 組態範例

在此範例中，`SQS OnSuccess` 組態不會指定目的地，因此 SAM 會隱含地建立 `SQS` 佇列並新增任何必要的許可。此外，在此範例中，範本檔案中宣告的 `Lambda` 資源目的地是在 `OnFailure` 組態中指定，因此 SAM 會將必要的許可新增至此 `Lambda` 事件函數，以呼叫目的地 `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.
```

使用 SNS 目的地的 EventInvoke 組態範例

在此範例中，會為 OnSuccess 組態的範本檔案中宣告的 SNS 主題指定目的地。

YAML

```
EventInvokeConfig:
  DestinationConfig:
    OnSuccess:
      Type: SNS
      Destination:
        Ref: DestinationSNS          # Arn of an SNS topic declared in the template 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取決於該類型的屬性。如需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
```

Properties

Properties

描述此事件映射屬性的物件。一組屬性必須符合定義的類型。

類型：[AlexaSkill](#) | [Api](#) | [CloudWatchEvent](#) | [CloudWatchLogs](#) | [Cognito](#) | [DocumentDB](#) | [DynamoDB](#) | [EventBridgeRule](#) | [HttpApi](#) | [IoTRule](#) | [Kinesis](#) | [MQ](#) | [MSK](#) | [S3](#) | [Schedule](#) | [ScheduleV2](#) | [SelfManagedKafka](#) | [SNS](#) | [SQS](#)

必要：是

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

Properties

SkillId

Alexa Skill 的 Alexa Skill ID。如需技能 ID 的詳細資訊，請參閱 [Alexa Skills Kit 文件中的設定 Lambda 函數的觸發](#)。

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

範例

AlexaSkillTrigger

Alexa 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 操作。

如果[AWS::Serverless::Api](#)未定義任何，則函數輸入和輸出代表 HTTP 請求和 HTTP 回應。

例如，使用 JavaScript API，可以透過傳回具有金鑰 statusCode 和內文的物件來控制回應statusCode 和內文。

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

```
Auth: ApiFunctionAuth
Method: String
Path: String
```



```
RequestModel: RequestModel  
RequestParameters: List of [ String | RequestParameter ]  
RestApiId: String  
ResponseTransferMode: String  
TimeoutInMillis: Integer
```

Properties

Auth

此特定 Api+Path+Method 的身分驗證組態。

用於在未DefaultAuthorizer指定或覆寫預設設定時，覆寫個別路徑上 API DefaultAuthorizer ApiKeyRequired的設定身分驗證組態。

類型：[ApiFunctionAuth](#)

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

Method

叫用此函數的 HTTP 方法。選項包括 DELETE、GET、HEAD、OPTIONS、PATCH、PUT、POST和 ANY。如需詳細資訊，請參閱《[API Gateway 開發人員指南](#)》中的設定 HTTP 方法。

類型：字串

必要：是

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

Path

叫用此函數的 Uri 路徑。必須以 開頭/。

類型：字串

必要：是

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

RequestModel

請求使用此特定 Api+Path+Method 的模型。這應該參考 [AWS::Serverless::Api](#) 資源 Models區段中指定的模型名稱。

類型：[RequestModel](#)

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

RequestParameters

請求此特定 Api+Path+Method 的參數組態。所有參數名稱都必須以開頭 `method.request.header` , `method.request` 且必須限制為 `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：【 String | [RequestParameter](#) 】 的清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

RestApiId

RestApi 資源的識別符，必須包含具有指定路徑和方法的操作。一般而言，這會設定為參考此範本中定義的 [AWS::Serverless::Api](#) 資源。

如果您未定義此屬性，會使用產生的 `OpenApi` 文件 AWS SAM 建立預設 [AWS::Serverless::Api](#) 資源。該資源包含由相同範本中未指定 之 `Api` 事件定義的所有路徑和方法的聯集 `RestApiId`。

這無法參考另一個範本中定義的 [AWS::Serverless::Api](#) 資源。

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

ResponseTransferMode

Lambda 函數整合的回應傳輸模式。設定為 `RESPONSE_STREAM` 以透過 API Gateway 啟用 Lambda 回應串流，允許函數將回應串流回用戶端。設為 `時RESPONSE_STREAM`，API Gateway 會使用 `Lambda InvokeWithResponseStreaming` API。

類型：字串

必要：否

有效值：BUFFERED | RESPONSE_STREAM

CloudFormation 相容性：此屬性會直接傳遞至的 [ResponseTransferMode](#) 屬性AWS::ApiGateway::Method Integration。

TimeoutInMillis

自訂介於 50 和 29,000 毫秒之間的逾時。

 Note

當您指定此屬性時，會 AWS SAM 修改您的 OpenAPI 定義。OpenAPI 定義必須使用 DefinitionBody 屬性內嵌指定。

類型：整數

必要：否

預設：29,000 毫秒或 29 秒

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有同等 CloudFormation 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

為特定 API、路徑和方法設定事件層級的授權。

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
```

Properties

ApiKeyRequired

此 API、路徑和方法需要 API 金鑰。

類型：布林值

必要：否

CloudFormation 相容性：此屬性對 AWS SAM 是唯一的，並且沒有 CloudFormation 同等屬性。

AuthorizationScopes

要套用至此 API、路徑和方法的授權範圍。

如果您已指定 DefaultAuthorizer 屬性套用的任何範圍，您指定的範圍將會覆寫它。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 AWS SAM 是唯一的，並且沒有 CloudFormation 同等屬性。

Authorizer

特定函數Authorizer的。

如果您為 `AWS::Serverless::Api` 資源指定了全域授權方，則可以將 `Authorizer` 設定為 來覆寫授權方 `NONE`。如需範例，請參閱 [覆寫 Amazon API Gateway REST API 的全域授權方](#)。

 Note

如果您使用 `AWS::Serverless::Api` 資源的 `DefinitionBody` 屬性來描述 API，則必須使用 `OverrideApiAuth` 搭配 `Authorizer` 來覆寫您的全域授權方。如需詳細資訊，請參閱 [OverrideApiAuth](#)。

有效值：AWS_IAM、NONE 或 AWS SAM 範本中定義之任何授權方的邏輯 ID。

類型：字串

必要：否

CloudFormation 相容性：此屬性對 AWS SAM 是唯一的，並且沒有 CloudFormation 同等屬性。

InvokeRole

指定 `InvokeRole` 用於 AWS_IAM 授權的。

類型：字串

必要：否

預設：CALLER_CREDENTIALS

CloudFormation 相容性：此屬性對 AWS SAM 是唯一的，並且沒有 CloudFormation 同等屬性。

其他備註：CALLER_CREDENTIALS 映射至 `arn:aws:iam::<user>/`，這會使用呼叫者登入資料來叫用端點。

OverrideApiAuth

將指定為 `true` 以覆寫 `AWS::Serverless::Api` 資源的全域授權方組態。只有在您指定全域授權方並使用 `AWS::Serverless::Api` 資源的屬性來描述 API 時，才需要此 `DefinitionBody` 屬性。

 Note

當您將指定 `OverrideApiAuth` 為 `true` 時，AWS SAM 會使用為 `ApiKeyRequired`、`Authorizer` 或 提供的任何值來覆寫您的全域授權方 `ResourcePolicy`。因此，使用時，

至少也必須指定其中一個屬性`OverrideApiAuth`。如需範例，請參閱 [指定 `DefinitionBody` for `AWS::Serverless::Api` 時覆寫全域授權方](#)。

類型：布林值

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

ResourcePolicy

在 API 上設定此路徑的資源政策。

類型：[ResourcePolicyStatement](#)

必要：否

CloudFormation 相容性：此屬性對 AWS SAM 是唯一的，並且沒有 CloudFormation 同等屬性。

範例

Function-Auth

下列範例會在函數層級指定授權。

YAML

```
Auth:
  ApiKeyRequired: true
  Authorizer: NONE
```

覆寫 Amazon API Gateway REST API 的全域授權方

您可以為您的`AWS::Serverless::Api`資源指定全域授權方。以下是設定全域預設授權方的範例：

```
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 事件函數的預設授權方，您可以將 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 for AWS::Serverless::Api 時覆寫全域授權方

使用 DefinitionBody 屬性描述AWS::Serverless::Api資源時，先前的覆寫方法無法運作。以下是將 DefinitionBody 屬性用於 AWS::Serverless::Api 資源的範例：

```

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

```

若要覆寫全域授權方，請使用 `OverrideApiAuth` 屬性。以下是使用 `OverrideApiAuth` 以提供的值覆寫全域授權方的範例 `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 的所有方法和路徑設定資源政策。如需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 資源政策控制對 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
```

Properties

AwsAccountBlacklist

要封鎖 AWS 的帳戶。

類型：字串的清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

AwsAccountWhitelist

要允許 AWS 的帳戶。如需此屬性的範例使用方式，請參閱此頁面底部的範例區段。

類型：字串的清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

CustomStatements

要套用至此 API 的自訂資源政策陳述式清單。如需此屬性的範例使用方式，請參閱此頁面底部的範例區段。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

IntrinsicVpcBlacklist

要封鎖的虛擬私有雲端 (VPCs) 清單，其中每個 VPC 指定為參考，例如[動態參考](#)或Ref [內部函數](#)。如需此屬性的範例使用方式，請參閱此頁面底部的範例區段。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

IntrinsicVpcWhitelist

要允許的 VPCs 清單，其中每個 VPC 指定為參考，例如[動態參考](#)或Ref [內部函數](#)。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

IntrinsicVpceBlacklist

要封鎖的 VPC 端點清單，其中每個 VPC 端點指定為參考，例如[動態參考](#)或Ref [內部函數](#)。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

IntrinsicVpceWhitelist

要允許的 VPC 端點清單，其中每個 VPC 端點指定為參考，例如[動態參考](#)或Ref [內部函數](#)。如需此屬性的範例使用方式，請參閱此頁面底部的範例區段。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

IpRangeBlacklist

要封鎖的 IP 地址或地址範圍。如需此屬性的範例使用方式，請參閱此頁面底部的範例區段。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

IpRangeWhitelist

要允許的 IP 地址或地址範圍。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

SourceVpcBlacklist

要封鎖的來源 VPC 或 VPC 端點。來源 VPC 名稱必須以 開頭"vpc-"，來源 VPC 端點名稱必須以 開頭"vpce-"。如需此屬性的範例使用方式，請參閱此頁面底部的範例區段。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

SourceVpcWhitelist

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

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

範例

資源政策範例

下列範例會封鎖兩個 IP 地址和來源 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

設定特定 Api+Path+Method 的請求模型。

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
```

Properties

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

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

Required

在指定 API 端點的 OpenApi 定義的參數區段中新增required屬性。

類型：布林值

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

ValidateBody

指定 API Gateway 是否使用 Model 驗證請求內文。如需詳細資訊，請參閱《[API Gateway 開發人員指南](#)》中的在 [API Gateway 中啟用請求驗證](#)。

類型：布林值

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

ValidateParameters

指定 API Gateway 是否使用 Model 驗證請求路徑參數、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 中啟用請求驗證](#)。

類型：布林值

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

範例

請求模型

請求模型範例

YAML

```
RequestModel:
  Model: User
  Required: true
  ValidateBody: true
  ValidateParameters: true
```

RequestParameter

設定特定 Api+Path+Method 的請求參數。

請求參數需要指定 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
```

Properties

Caching

將 `cacheKeyParameters` 區段新增至 API Gateway OpenApi 定義

類型：布林值

必要：有條件

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

Required

此欄位指定是否需要參數

類型：布林值

必要：有條件

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

範例

請求參數

設定請求參數的範例

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) 會在設定此事件類型時產生 [AWS::Events::Rule](#) 資源。

重要注意事項：[EventBridgeRule](#)是要使用的慣用事件來源類型，而不是 CloudWatchEvent。EventBridgeRule 和 CloudWatchEvent會使用相同的基礎服務、API CloudFormation 和資源。不過，AWS SAM 只會將新功能的支援新增至 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
```

Properties

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 屬性，但不能同時指定兩者。

類型：布林值

必要：否

CloudFormation 相容性：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：預設事件匯流排

CloudFormation 相容性：此屬性會直接傳遞至 `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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Events::Rule Target` 資源的 [Input](#) 屬性。

InputPath

當您不想將整個相符事件傳遞至目標時，請使用 `InputPath` 屬性來描述要傳遞的事件部分。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `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](#)》中的 [EventBridge 中的事件和事件模式](#)。EventBridge

類型：[EventPattern](#)

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 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 屬性，但不能同時指定兩者。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 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條件，並將其與指定的日誌群組建立關聯。

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
```

Properties

FilterPattern

篩選表達式，限制交付至目的地 AWS 資源的項目。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

CloudFormation 相容性：此屬性會直接傳遞至 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。如果篩選條件模式符合日誌事件，則上傳至此日誌群組的所有日誌事件都會經過篩選，並交付至指定的 AWS 資源。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Logs::SubscriptionFilter 資源的 [LogGroupName](#) 屬性。

範例

Cloudwatchlogs 訂閱篩選條件

Cloudwatchlogs 訂閱篩選條件範例

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
```

Properties

Trigger

新使用者集區的 Lambda 觸發組態資訊。

類型：清單

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Cognito::UserPool` 資源的 [LambdaConfig](#) 屬性。

UserPool

相同範本中定義的 UserPool 參考

類型：字串

必要：是

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

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[搭配使用 AWS Lambda 與 Amazon DocumentDB](#)。

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
KmsKeyArn: String
MaximumBatchingWindowInSeconds: Integer
SecretsManagerKmsKeyId: String
SourceAccessConfigurations: List
StartingPosition: String
StartingPositionTimestamp: Double
```

Properties

BatchSize

要在單一批次中擷取的最大項目數。

類型：整數

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [BatchSize](#) 屬性。

Cluster

Amazon DocumentDB 叢集的 Amazon Resource Name (ARN)。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMappingDocumentDBEventSourceConfig` 資料類型的 [CollectionName](#) 屬性。

DatabaseName

Amazon DocumentDB 叢集內要使用的資料庫名稱。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至

AWS::Lambda::EventSourceMappingDocumentDBEventSourceConfig資料類型的 [DatabaseName](#) 屬性。

Enabled

如果為 true，則事件來源映射為作用中。若要暫停輪詢和調用，請將設定為 false。

類型：布林值

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Lambda::EventSourceMapping 資源的 [Enabled](#) 屬性。

FilterCriteria

定義判斷 Lambda 是否應處理事件之條件的物件。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

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Lambda::EventSourceMapping 資源的 [FilterCriteria](#) 屬性。

FullDocument

決定 Amazon DocumentDB 在文件更新操作期間傳送到事件串流的內容。如果設定為 UpdateLookup，Amazon DocumentDB 會傳送描述變更的差異，以及整個文件的副本。否則，Amazon DocumentDB 只會傳送包含變更的部分文件。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至

AWS::Lambda::EventSourceMappingDocumentDBEventSourceConfig資料類型的 [FullDocument](#) 屬性。

KmsKeyArn

金鑰的 Amazon Resource Name (ARN)，用於加密與此事件相關的資訊。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [KmsKeyArn](#) 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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [MaximumBatchingWindowInSeconds](#) 屬性。

SecretsManagerKmsKeyId

AWS Secrets Manager 的客戶受管金鑰的 AWS Key Management Service (AWS KMS) 金鑰 ID。當您使用 Secrets Manager 的客戶受管金鑰搭配不包含 `kms:Decrypt` 許可的 Lambda 執行角色時，此為必要項目。

此屬性的值是 UUID。例如：1abc23d4-567f-8ab9-cde0-1fab234c5d67。

類型：字串

必要：有條件

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有同等 CloudFormation 的。

SourceAccessConfigurations

身分驗證通訊協定或虛擬主機的陣列。使用 [SourceAccessConfigurations](#) 資料類型指定此項目。

對於 DocumentDB 事件來源類型，唯一的有效組態類型為 BASIC_AUTH。

- BASIC_AUTH – 存放代理程式登入資料的 Secrets Manager 秘密。對於此類型，登入資料必須採用下列格式：`{"username": "your-username", "password": "your-password"}`。僅允許一個 類型的物件 BASIC_AUTH。

類型：清單

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [SourceAccessConfigurations](#) 屬性。

StartingPosition

要從中開始讀取的串流位置。

- `AT_TIMESTAMP` – 指定從中開始讀取記錄的時間。
- `LATEST` – 唯讀新記錄。
- `TRIM_HORIZON` – 處理所有可用的記錄。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `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` `StartingPosition` 何時指定為 `AT_TIMESTAMP`。

類型：Double

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [StartingPositionTimestamp](#) 屬性。

範例

Amazon DocumentDB 事件來源

```
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[搭配使用 AWS Lambda 與 Amazon DynamoDB](#)。

AWS SAM 會在設定此事件類型時產生 [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
KmsKeyArn: String
MaximumBatchingWindowInSeconds: Integer
MaximumRecordAgeInSeconds: Integer
MaximumRetryAttempts: Integer
MetricsConfig: MetricsConfig
ParallelizationFactor: Integer
StartingPosition: String
StartingPositionTimestamp: Double
Stream: String
TumblingWindowInSeconds: Integer

```

Properties

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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [BatchSize](#) 屬性。

下限：1

上限：1000

BisectBatchOnFunctionError

如果函數傳回錯誤，請將批次分割為兩個，然後重試。

類型：布林值

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [BisectBatchOnFunctionError](#) 屬性。

DestinationConfig

捨棄記錄的 Amazon Simple Queue Service (Amazon SQS) 佇列或 Amazon Simple Notification Service (Amazon SNS) 主題目的地。

類型：[DestinationConfig](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [Enabled](#) 屬性。

FilterCriteria

定義判斷 Lambda 是否應處理事件之條件的物件。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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [FilterCriteria](#) 屬性。

FunctionResponseTypes

目前套用至事件來源映射的回應類型清單。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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [FunctionResponseTypes](#) 屬性。

KmsKeyArn

金鑰的 Amazon Resource Name (ARN)，用於加密與此事件相關的資訊。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [KmsKeyArn](#) 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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [MaximumBatchingWindowInSeconds](#) 屬性。

MaximumRecordAgeInSeconds

Lambda 傳送給函數進行處理的記錄最長存留期。

類型：整數

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [MaximumRetryAttempts](#) 屬性。

MetricsConfig

選擇加入組態，以取得擷取每個處理階段之事件來源映射的增強指標。如需範例，請參閱 [MetricsConfig 事件](#)。

類型：[MetricsConfig](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [MetricsConfig](#) 屬性。

ParallelizationFactor

要同時處理每個碎片的批次數量。

類型：整數

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [ParallelizationFactor](#) 屬性。

StartingPosition

要從中開始讀取的串流位置。

- `AT_TIMESTAMP` – 指定從中開始讀取記錄的時間。
- `LATEST` – 唯讀新記錄。
- `TRIM_HORIZON` – 處理所有可用的記錄。

有效值：`AT_TIMESTAMP` | `LATEST` | `TRIM_HORIZON`

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `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` `StartingPosition` 何時指定為 `AT_TIMESTAMP`。

類型：Double

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [StartingPositionTimestamp](#) 屬性。

Stream

DynamoDB 串流的 Amazon Resource Name (ARN)。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [EventSourceArn](#) 屬性。

TumblingWindowInSeconds

處理時段的持續時間，以秒為單位。有效範圍為 1 到 900 (15 分鐘)。

如需詳細資訊，請參閱《AWS Lambda 開發人員指南》中的[輪轉視窗](#)。

類型：整數

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [TumblingWindowInSeconds](#) 屬性。

範例

MetricsConfig 事件

以下是資源的範例，該資源使用 `MetricsConfig` 屬性來擷取其事件來源映射的每個處理階段。

```
Resources:
  FilteredEventsFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: s3://sam-demo-bucket/metricsConfig.zip
      Handler: index.handler
      Runtime: nodejs16.x
      Events:
        KinesisStream:
          Type: Kinesis
          Properties:
            Stream: !GetAtt KinesisStream.Arn
            StartingPosition: LATEST
            MetricsConfig:
              Metrics:
                - EventCount
```

現有 DynamoDB 資料表的 DynamoDB 事件來源

AWS 帳戶中已存在的 DynamoDB 資料表的 DynamoDB 事件來源。

YAML

```
Events:
  DDBEvent:
    Type: DynamoDB
    Properties:
      Stream: arn:aws:dynamodb:us-east-1:123456789012:table/TestTable/
stream/2016-08-11T21:21:33.291
      StartingPosition: TRIM_HORIZON
      BatchSize: 10
      Enabled: false
```

範本中宣告的 DynamoDB 資料表的 DynamoDB 事件

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，會將無伺服器函數設定為 Amazon EventBridge 規則的目標。如需詳細資訊，請參閱《Amazon EventBridge 使用者指南》中的[什麼是 Amazon EventBridge ?](#)。

AWS SAM 會在設定此事件類型時產生 [AWS::Events::Rule](#) 資源。AWS SAM 也會建立 資源AWS::Lambda::Permission，這是必要的，以便 EventBridgeRule可以呼叫 Lambda。

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
```

Properties

DeadLetterConfig

設定 EventBridge 在目標呼叫失敗後傳送事件的 Amazon Simple Queue Service (Amazon SQS) 佇列。例如，將事件傳送至不存在的 Lambda 事件函數時，或 EventBridge 沒有足夠的許可來叫用 Lambda 函數時，叫用可能會失敗。如需詳細資訊，請參閱《Amazon EventBridge 使用者指南》中的[事件重試政策和使用無效字母佇列](#)。

Note

[AWS::Serverless::Function](#) 資源類型具有類似的資料類型 `DeadLetterQueue`，可處理成功調用目標 Lambda 函數後發生的失敗。這些失敗類型的範例包括 Lambda 限流，或 Lambda 目標函數傳回的錯誤。如需函數 `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

CloudFormation 相容性：此屬性類似於 `AWS::Events::RuleTarget` 資料類型的 [DeadLetterConfig](#) 屬性。此屬性的 AWS SAM 版本包含其他子屬性，以防您想要為您 AWS SAM 建立無效字母佇列。

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：預設事件匯流排

CloudFormation 相容性：此屬性會直接傳遞至 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

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Events::Rule Target 資源的 [Input](#) 屬性。

InputPath

當您不想將整個相符事件傳遞至目標時，請使用 InputPath 屬性來描述要傳遞的事件部分。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 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](#)。 EventBridge

類型：[InputTransformer](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Events::RuleTarget 資料類型的 [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》中的 Amazon [EventBridge 事件](#) 和 [EventBridge 事件模式](#)。 EventBridge

類型：[EventPattern](#)

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 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 EventBridge 使用者指南》中的 [事件重試政策和使用無效字母佇列](#)。

類型：[RetryPolicy](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Events::Rule Target 資料類型的 [RetryPolicy](#) 屬性。

RuleName

規則的名稱。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Events::Rule 資源的 [Name](#) 屬性。

State

規則的狀態。

接受的值：DISABLED | ENABLED | ENABLED_WITH_ALL_CLOUDTRAIL_MANAGEMENT_EVENTS

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Events::Rule 資源的 [State](#) 屬性。

Target

EventBridge 在觸發規則時叫用 AWS 的資源。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

CloudFormation 相容性：此屬性類似於 AWS::Events::Rule 資源的 [Targets](#) 屬性。Amazon EC2 RebootInstances API call 是目標屬性的範例。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

用來指定 Amazon Simple Queue Service (Amazon SQS) 佇列的物件，其中 EventBridge 會在失敗的目標調用後傳送事件。例如，將事件傳送至不存在的 Lambda 事件函數時，呼叫可能會失敗，或沒有

足夠的許可來叫用 Lambda 函數。如需詳細資訊，請參閱《Amazon EventBridge 使用者指南》中的[事件重試政策和使用無效字母佇列](#)。

Note

[AWS::Serverless::Function](#) 資源類型具有類似的資料類型，DeadLetterQueue可處理成功調用目標 Lambda 函數後發生的失敗。此類故障的範例包括 Lambda 限流，或 Lambda 目標函數傳回的錯誤。如需 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
```

Properties

Arn

指定為無效字母佇列目標之 Amazon SQS 佇列的 Amazon Resource Name (ARN)。

Note

指定 Type 屬性或 Arn 屬性，但不能同時指定兩者。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Events::Rule DeadLetterConfig` 資料類型的 [Arn](#) 屬性。

QueueLogicalId

Type 指定 時 AWS SAM 建立的無效字母佇列自訂名稱。

 Note

如果未設定 `Type` 屬性，則會忽略此屬性。

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

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，並連接必要的[資源型政策](#)，以授予將事件傳送至佇列的規則資源許可。

 Note

指定 `Type` 屬性或 `Arn` 屬性，但不能同時指定兩者。

有效值：SQS

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

範例

DeadLetterConfig

DeadLetterConfig

YAML

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

Target

設定觸發規則時 EventBridge 叫用 AWS 的資源。

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
```

Properties

Id

目標的邏輯 ID。

的值Id可以包含英數字元、句點 (.)、連字號 (-) 和底線 ()_。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Events::Rule Target` 資料類型的 [Id](#) 屬性。

範例

Target

YAML

```
EBRule:
  Type: EventBridgeRule
  Properties:
    Target:
      Id: MyTarget
```

HttpApi

描述具有 HttpApi 類型的事件來源的物件。

如果 API 上存在指定路徑和方法的 OpenApi 定義，SAM 會為您新增 Lambda 整合和安全性區段（如適用）。

如果 API 上沒有指定路徑和方法的 OpenApi 定義，SAM 會為您建立此定義。

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
```

Properties

ApiId

此範本中定義之 [AWS::Serverless::HttpApi](#) 資源的識別符。

如果未定義，則會 ServerlessHttpApi 使用產生的 OpenApi 文件來建立稱為 的預設 [AWS::Serverless::HttpApi](#) 資源，該文件包含此範本中定義之 Api 事件所定義的所有路徑和方法的聯集，這些未指定 ApiId。

這無法參考另一個範本中定義的 [AWS::Serverless::HttpApi](#) 資源。

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

Auth

此特定 Api+Path+Method 的驗證組態。

用於在未指定 DefaultAuthorizer 時覆寫 API 的 DefaultAuthorizer 或對個別路徑設定身分驗證組態。

類型：[HttpApiFunctionAuth](#)

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

Method

叫用此函數的 HTTP 方法。

如果未指定 Method Path和 ，SAM 會建立預設 API 路徑，將任何未對應至不同端點的請求路由至此 Lambda 事件函數。每個 API 只能存在其中一個預設路徑。

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

Path

叫用此函數的 Uri 路徑。必須以 開頭/。

如果未指定 Method Path和 ，SAM 會建立預設 API 路徑，將任何未對應至不同端點的請求路由至此 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

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

PayloadFormatVersion

為傳送至整合的承載指定格式。

注意：PayloadFormatVersion 需要 SAM 修改您的 OpenAPI 定義，因此僅適用於 DefinitionBody 屬性中定義的內嵌 OpenApi。

類型：字串

必要：否

預設：2.0

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

RouteSettings

此 HTTP API 的每個路由路由設定。如需路由設定的詳細資訊，請參閱 API Gateway 開發人員指南中的 [AWS::ApiGatewayV2::Stage RouteSettings](#)。

注意：如果在 HttpApi 資源和事件來源中都指定 RouteSettings ，會將它們與優先的事件來源屬性 AWS SAM 合併。

類型：[RouteSettings](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 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 需要 SAM 修改您的 OpenAPI 定義，因此僅適用於 DefinitionBody 屬性中定義的內嵌 OpenApi。

類型：整數

必要：否

預設：5000

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

範例

預設 HttpApi 事件

使用預設路徑的 HttpApi 事件。此 API 上所有未映射的路徑和方法都會路由至此端點。

YAML

```
Events:
  HttpApiEvent:
```

Type: HttpApi

HttpApi

使用特定路徑和方法的 HttpApi 事件。

YAML

```
Events:
  HttpApiEvent:
    Type: HttpApi
    Properties:
      Path: /
      Method: GET
```

HttpApi 授權

使用 授權方的 HttpApi 事件。

YAML

```
Events:
  HttpApiEvent:
    Type: HttpApi
    Properties:
      Path: /authenticated
      Method: GET
      Auth:
        Authorizer: OpenIdAuth
        AuthorizationScopes:
          - scope1
          - scope2
```

HttpApiFunctionAuth

在事件層級設定授權。

設定特定 API + 路徑 + 方法的驗證

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
```

Properties

AuthorizationScopes

要套用至此 API、路徑和方法的授權範圍。

DefaultAuthorizer 如果存在，此處列出的範圍會覆寫 套用的任何範圍。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

Authorizer

特定函數Authorizer的。若要使用 IAM 授權，請在範本的 Globals區段EnableIamAuthorizer中指定 AWS_IAM和 true的。

如果您已在 API 上指定全域授權方，並想要將特定函數設為公有，請將 Authorizer設定為 以覆寫 NONE。

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

範例

Function-Auth

在函數層級指定授權

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，您還必須在範本的 Globals區段EnableIamAuthorizer中指定 true 的。如需詳細資訊，請參閱[AWS SAM 範本的全域區段](#)。

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 [CloudFormation 文件](#)

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

```
AwsIotSqlVersion: String  
Sql: String
```

Properties

AwsIotSqlVersion

評估規則時所用的 SQL 規則引擎版本。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::IoT::TopicRule TopicRulePayload` 資源的 [Sql](#) 屬性。

範例

IOT 規則

IOT 規則範例

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[搭配使用 AWS Lambda 與 Amazon Kinesis](#)。

AWS SAM 會在設定此事件類型時產生 [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  
KmsKeyArn: String  
MaximumBatchingWindowInSeconds: Integer  
MaximumRecordAgeInSeconds: Integer  
MaximumRetryAttempts: Integer  
MetricsConfig: MetricsConfig  
ParallelizationFactor: Integer  
StartingPosition: String  
StartingPositionTimestamp: Double  
Stream: String  
TumblingWindowInSeconds: Integer
```

Properties

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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [BatchSize](#) 屬性。

下限：1

上限：10000

BisectBatchOnFunctionError

如果函數傳回錯誤，請將批次分割為兩個，然後重試。

類型：布林值

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [BisectBatchOnFunctionError](#) 屬性。

DestinationConfig

捨棄記錄的 Amazon Simple Queue Service (Amazon SQS) 佇列或 Amazon Simple Notification Service (Amazon SNS) 主題目的地。

類型：[DestinationConfig](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [Enabled](#) 屬性。

FilterCriteria

定義判斷 Lambda 是否應處理事件之條件的物件。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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [FilterCriteria](#) 屬性。

FunctionResponseTypes

目前套用至事件來源映射的回應類型清單。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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [FunctionResponseTypes](#) 屬性。

KmsKeyArn

金鑰的 Amazon Resource Name (ARN)，用於加密與此事件相關的資訊。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [KmsKeyArn](#) 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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [MaximumBatchingWindowInSeconds](#) 屬性。

MaximumRecordAgeInSeconds

Lambda 傳送給函數進行處理的記錄最長存留期。

類型：整數

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [MaximumRetryAttempts](#) 屬性。

MetricsConfig

選擇加入組態，以取得擷取每個處理階段之事件來源映射的增強指標。如需範例，請參閱 [MetricsConfig 事件](#)。

類型：[MetricsConfig](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [MetricsConfig](#) 屬性。

ParallelizationFactor

要同時處理每個碎片的批次數量。

類型：整數

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [ParallelizationFactor](#) 屬性。

StartingPosition

要從中開始讀取的串流位置。

- `AT_TIMESTAMP` – 指定從中開始讀取記錄的時間。

- LATEST – 唯讀新記錄。
- TRIM_HORIZON – 處理所有可用的記錄。

有效值：AT_TIMESTAMP | LATEST | TRIM_HORIZON

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 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 StartingPosition 何時指定為 AT_TIMESTAMP。

類型：Double

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Lambda::EventSourceMapping 資源的 [StartingPositionTimestamp](#) 屬性。

Stream

資料串流或串流消費者的 Amazon Resource Name (ARN)。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Lambda::EventSourceMapping 資源的 [EventSourceArn](#) 屬性。

TumblingWindowInSeconds

處理時段的持續時間，以秒為單位。有效範圍為 1 到 900 (15 分鐘)。

如需詳細資訊，請參閱《AWS Lambda 開發人員指南》中的 [輪轉視窗](#)。

類型：整數

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [TumblingWindowInSeconds](#) 屬性。

範例

MetricsConfig 事件

以下是資源的範例，該資源使用 `MetricsConfig` 屬性來擷取其事件來源映射的每個處理階段。

```
Resources:
  FilteredEventsFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: s3://sam-demo-bucket/metricsConfig.zip
      Handler: index.handler
      Runtime: nodejs16.x
      Events:
        KinesisStream:
          Type: Kinesis
          Properties:
            Stream: !GetAtt KinesisStream.Arn
            StartingPosition: LATEST
            MetricsConfig:
              Metrics:
                - EventCount
```

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[搭配使用 Lambda 與 Amazon MQ](#)。

AWS Serverless Application Model (AWS SAM) 會在設定此事件類型時產生 [AWS::Lambda::EventSourceMapping](#) 資源。

Note

若要在虛擬私有雲端 (VPC) 中擁有連線至公有網路中 Lambda 事件函數的 Amazon MQ 佇列，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[執行角色許可](#)。

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
KmsKeyArn: String
MaximumBatchingWindowInSeconds: Integer
Queues: List
```

`SecretsManagerKmsKeyId`: *String*
`SourceAccessConfigurations`: *List*

Properties

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

CloudFormation 相容性：此屬性會直接傳遞至 `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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [EventSourceArn](#) 屬性。

DynamicPolicyName

根據預設，AWS Identity and Access Management (IAM) 政策名稱 `SamAutoGeneratedAMQPolicy` 用於回溯相容性。指定 `true` 為您的 IAM 政策使用自動產生的名稱。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` 以避免重複的 IAM 政策名稱。

類型：布林值

必要：否

預設：false

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

Enabled

如果為 true，則事件來源映射為作用中。若要暫停輪詢和調用，請將 設定為 false。

類型：布林值

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Lambda::EventSourceMapping 資源的 [Enabled](#) 屬性。

FilterCriteria

定義判斷 Lambda 是否應處理事件之條件的物件。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

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Lambda::EventSourceMapping 資源的 [FilterCriteria](#) 屬性。

KmsKeyArn

金鑰的 Amazon Resource Name (ARN)，用於加密與此事件相關的資訊。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Lambda::EventSourceMapping 資源的 [KmsKeyArn](#) 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

CloudFormation 相容性：此屬性會直接傳遞至 `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：清單

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [Queues](#) 屬性。

SecretsManagerKmsKeyId

客戶受管金鑰的 AWS Key Management Service (AWS KMS) 金鑰 ID AWS Secrets Manager。當您使用來自 Secrets Manager 的客戶受管金鑰搭配不包含 `kms:Decrypt` 許可的 Lambda 執行角色時，此為必要項目。

此屬性的值是 UUID。例如：1abc23d4-567f-8ab9-cde0-1fab234c5d67。

類型：字串

必要：有條件

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

SourceAccessConfigurations

身分驗證通訊協定或虛擬主機的陣列。使用 [SourceAccessConfigurations](#) 資料類型指定此項目。

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** – 存放代理程式登入資料的 Secrets Manager 秘密。對於此類型，登入資料必須採用下列格式：`{"username": "your-username", "password": "your-password"}`。僅允許一個 類型的物件 `BASIC_AUTH`。
- **VIRTUAL_HOST** – RabbitMQ 代理程式中的虛擬主機名稱。Lambda 將使用此 Rabbit MQ 的主機做為事件來源。僅允許一個 類型的物件 `VIRTUAL_HOST`。

類型：清單

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `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MQ的事件來源類型範例。

YAML

```
Events:
  MQEvent:
    Type: MQ
    Properties:
      Broker: arn:aws:mq:us-east-2:123456789012:broker:MyBroker:b-1234a5b6-78cd-901e-2fgh-3i45j6k17819
      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 開發人員指南》中的[搭配使用 AWS Lambda 與 Amazon MSK](#)。

AWS Serverless Application Model (AWS SAM) 會在設定此事件類型時產生 [AWS::Lambda::EventSourceMapping](#) 資源。

若要使用結構描述登錄檔，您需要定義函數的特定 IAM 角色許可。如需所需組態的範例，請參閱[使用 IAM 角色完成設定](#)。

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  
BisectBatchOnFunctionError: Boolean  
ConsumerGroupId: String  
DestinationConfig: DestinationConfig  
Enabled: Boolean  
FilterCriteria: FilterCriteria  
FunctionResponseTypes: List  
KmsKeyArn: String  
MaximumBatchingWindowInSeconds: Integer  
MaximumRecordAgeInSeconds: Integer  
MaximumRetryAttempts: Integer  
LoggingConfig: LoggingConfig  
MetricsConfig: MetricsConfig  
ProvisionedPollerConfig: ProvisionedPollerConfig  
SchemaRegistryConfig: SchemaRegistryConfig  
SourceAccessConfigurations: SourceAccessConfigurations  
StartingPosition: String  
StartingPositionTimestamp: Double  
Stream: String  
Topics: List
```

Properties

BatchSize

Lambda 從串流或佇列中提取並傳送至函數的每個批次中的記錄數目上限。Lambda 會將批次中的所有記錄以單一呼叫形式傳送至函數，直到達到同步調用的承載限制 (6 MB)。

預設值：100

有效範圍：最小值為 1。最大值為 10,000。

類型：整數

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [BatchSize](#) 屬性。

BisectBatchOnFunctionError

如果函數傳回錯誤，請將批次分割為兩個，然後重試。

類型：布林值

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [BisectBatchOnFunctionError](#) 屬性。

ConsumerGroupId

設定如何從 Kafka 主題讀取事件的字串。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `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。

使用此屬性可指定來自 Amazon MSK 事件來源的失敗調用目的地。

類型：[DestinationConfig](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [Enabled](#) 屬性。

FilterCriteria

定義判斷 Lambda 是否應處理事件之條件的物件。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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [FilterCriteria](#) 屬性。

FunctionResponseTypes

目前套用至事件來源映射的回應類型清單。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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [FunctionResponseTypes](#) 屬性。

KmsKeyArn

金鑰的 Amazon Resource Name (ARN)，用於加密與此事件相關的資訊。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [KmsKeyArn](#) 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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [MaximumBatchingWindowInSeconds](#) 屬性。

MaximumRecordAgeInSeconds

Lambda 傳送給函數進行處理的記錄最長存留期。

類型：整數

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [MaximumRetryAttempts](#) 屬性。

LoggingConfig

組態物件，指定事件來源映射的記錄組態。

類型：[LoggingConfig](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [LoggingConfig](#) 屬性。

MetricsConfig

組態物件，指定事件來源映射的指標組態。

類型：[MetricsConfig](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [MetricsConfig](#) 屬性。

ProvisionedPollerConfig

用於增加用於計算事件來源映射之輪詢器數量的組態。此組態允許最少 1 個輪詢器，最多 2000 個輪詢器。如需範例，請參閱 [ProvisionedPollerConfig 範例](#)。

類型：[ProvisionedPollerConfig](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [ProvisionedPollerConfig](#) 屬性。

SchemaRegistryConfig

搭配 Kafka 事件來源使用結構描述登錄檔的組態。

Note

此功能 `ProvisionedPollerConfig` 需要設定。

類型：`SchemaRegistryConfig`

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [AmazonManagedKafkaEventSourceConfig](#) 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](#) 清單

必要：否

CloudFormation 相容性：此屬性是 `AWS::Lambda::EventSourceMapping` 資源的 [AmazonManagedKafkaEventSourceConfig](#) 屬性的一部分。

StartingPosition

要從中開始讀取的串流位置。

- `AT_TIMESTAMP` – 指定從中開始讀取記錄的時間。
- `LATEST` – 唯讀新記錄。

- TRIM_HORIZON – 處理所有可用的記錄。

有效值：AT_TIMESTAMP | LATEST | TRIM_HORIZON

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 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 StartingPosition 何時指定為 AT_TIMESTAMP。

類型：Double

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Lambda::EventSourceMapping 資源的 [StartingPositionTimestamp](#) 屬性。

Stream

資料串流或串流取用者的 Amazon Resource Name (ARN)。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Lambda::EventSourceMapping 資源的 [EventSourceArn](#) 屬性。

Topics

Kafka 主題名稱。

類型：清單

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Lambda::EventSourceMapping 資源的 [Topics](#) 屬性。

範例

使用 IAM 角色完成設定

下列範例顯示完整的設定，包括使用結構描述登錄所需的 IAM 角色組態：

```
Parameters:
  PreCreatedSubnetOne:
    Type: String
  PreCreatedSubnetTwo:
    Type: String
  MskClusterName4:
    Type: String

Resources:
  MyLambdaExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Action: [sts:AssumeRole]
            Effect: Allow
            Principal:
              Service: [lambda.amazonaws.com]
      Policies:
        - PolicyName: KafkaClusterPermissions
          PolicyDocument:
            Statement:
              - Action: [kafka:DescribeClusterV2, kafka:GetBootstrapBrokers]
                Effect: Allow
                Resource: 'arn:aws:kafka:us-east-1:123456789012:cluster/*'
        - PolicyName: KafkaAuthPolicy
          PolicyDocument:
            Statement:
              - Action: [secretsmanager:GetSecretValue, kms:Decrypt]
                Effect: "Allow"
                Resource: ['arn:aws:secretsmanager:us-west-2:123456789012:secret:kafkaSecret-*****',
                          'arn:aws:kms:us-west-2:123456789012:key/keyId']
        - PolicyName: ENIPolicy
          PolicyDocument:
            Statement:
              - Action: [ec2:CreateNetworkInterface,
```



```

        ec2:DescribeNetworkInterfaces, ec2:DescribeVpcs,
ec2:DeleteNetworkInterface,
        ec2:DescribeSubnets, ec2:DescribeSecurityGroups]
    Effect: Allow
    Resource: '*'
  - PolicyName: SchemaRegistryPolicy
    PolicyDocument:
      Statement:
        - Action: [glue:GetRegistry]
          Effect: Allow
          Resource: 'arn:aws:glue:{region}:{account-id}:registry/{registry-name}'
  - PolicyName: SchemaVersionsPolicy
    PolicyDocument:
      Statement:
        - Action: [glue:GetSchemaVersions]
          Effect: Allow
          Resource: '*'
    ManagedPolicyArns:
      - !Sub arn:${AWS::Partition}:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole
    Tags:
      - {Value: SAM, Key: lambda:createdBy}

MyMskCluster:
  Type: AWS::MSK::Cluster
  Properties:
    BrokerNodeGroupInfo:
      ClientSubnets:
        - Ref: PreCreatedSubnetOne
        - Ref: PreCreatedSubnetTwo
      InstanceType: kafka.t3.small
      StorageInfo:
        EBSStorageInfo:
          VolumeSize: 1
    ClusterName:
      Ref: MskClusterName4
    KafkaVersion: 3.8.x
    NumberOfBrokerNodes: 2

MyMskStreamProcessor:
  Type: AWS::Serverless::Function
  Properties:
    Runtime: nodejs18.x
    Handler: index.handler

```

```

    CodeUri: ${codeuri}
    Role:
      Fn::GetAtt: [MyLambdaExecutionRole, Arn]
    Events:
      MyMskEvent:
        Type: MSK
        Properties:
          StartingPosition: LATEST
          Stream:
            Ref: MyMskCluster
          SourceAccessConfigurations:
            - Type: SASL_SCRAM_512_AUTH
              URI: !Sub arn:${AWS::Partition}:secretsmanager:us-
west-2:123456789012:secret:my-path/my-secret-name-1a2b3c
          Topics:
            - SchemaRegistryTestTopic
          ProvisionedPollerConfig:
            MinimumPollers: 1
          SchemaRegistryConfig:
            AccessConfigs:
              - Type: BASIC_AUTH
                URI: !Sub arn:${AWS::Partition}:secretsmanager:us-
west-2:123456789012:secret:my-path/my-secret-name-1a2b3c
            SchemaValidationConfigs:
              - Attribute: KEY
            EventRecordFormat: JSON
            SchemaRegistryURI: !Sub arn:${AWS::Partition}:glue:us-
west-2:123456789012:registry/myregistry

```

ProvisionedPollerConfig 範例

```

ProvisionedPollerConfig:
  MinimumPollers: 1
  MaximumPollers: 200

```

現有叢集的 Amazon MSK 範例

以下是已存在於 之 Amazon MSK 叢集MSK的事件來源類型範例 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
```

相同範本中宣告之叢集的 Amazon MSK 範例

以下是在相同範本檔案中宣告之 Amazon MSK 叢集MSK的事件來源類型範例。

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
```

具有結構描述登錄檔的 MSK 事件來源

以下是使用結構描述登錄檔設定MSK的事件來源類型範例。

```
Events:
  MSKEvent:
    Type: MSK
    Properties:
      StartingPosition: LATEST
      Stream:
        Ref: MyMskCluster
      Topics:
        - SchemaRegistryTestTopic
      ProvisionedPollerConfig:
        MinimumPollers: 1
      SchemaRegistryConfig:
        SchemaRegistryURI: !Sub arn:${AWS::Partition}:glue:us-
west-2:123456789012:registry/myregistry
```

```

EventRecordFormat: JSON
SchemaValidationConfigs:
  - Attribute: KEY
  - Attribute: VALUE

```

具有 Confluent 結構描述登錄檔的 MSK 事件來源

以下是使用 Confluent 結構描述登錄檔設定MSK的事件來源類型範例。

```

Events:
  MSKEvent:
    Type: MSK
    Properties:
      StartingPosition: LATEST
      Stream:
        Ref: MyMskCluster
      Topics:
        - SchemaRegistryTestTopic
    ProvisionedPollerConfig:
      MinimumPollers: 1
    SchemaRegistryConfig:
      SchemaRegistryURI: https://my-schema-registry.confluent.cloud
      AccessConfigs:
        - Type: BASIC_AUTH
          URI: !Sub arn:${AWS::Partition}:secretsmanager:us-
west-2:123456789012:secret:my-secret
      EventRecordFormat: JSON
      SchemaValidationConfigs:
        - Attribute: KEY
        - Attribute: VALUE

```

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](#)

Properties

Bucket

S3 儲存貯體名稱。此儲存貯體必須存在於相同的範本中。

類型：字串

必要：是

CloudFormation 相容性：此屬性類似於 AWS::S3::Bucket 資源的 [BucketName](#) 屬性。這是 SAM 中的必要欄位。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 事件函數的 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：字串 | 清單

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 AWS::S3::Bucket LambdaConfiguration 資料類型的 [Event](#) 屬性。

Filter

篩選規則，決定哪些 Amazon S3 物件叫用 Lambda 函數。如需 Amazon S3 金鑰名稱篩選的相關資訊，請參閱 [《Amazon Simple Storage Service 使用者指南》中的設定 Amazon S3 事件通知](#)。

類型：[NotificationFilter](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::S3::Bucket LambdaConfiguration 資料類型的 [Filter](#) 屬性。

範例

S3-Event

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，會將無伺服器函數設定為排程觸發的 Amazon EventBridge 規則的目標。如需詳細資訊，請參閱《Amazon EventBridge 使用者指南》中的[什麼是 Amazon EventBridge ?](#)。

AWS Serverless Application Model (AWS SAM) 會在設定此事件類型時產生 [AWS::Events::Rule](#) 資源。

Note

EventBridge 現在提供新的排程功能：[Amazon EventBridge。Scheduler](#)Amazon EventBridge Scheduler 是一種無伺服器排程器，可讓您從一個集中的受管服務建立、執行和管理任務。EventBridge Scheduler 可高度自訂，並提供比 EventBridge 排程規則更佳的可擴展性，具有更廣泛的目標 API 操作集和 AWS 服務。

我們建議您使用 EventBridge 排程Scheduler叫用目標。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
```

Properties

DeadLetterConfig

設定 EventBridge 在目標呼叫失敗後傳送事件的 Amazon Simple Queue Service (Amazon SQS) 佇列。例如，將事件傳送至不存在的 Lambda 事件函數時，或 EventBridge 沒有足夠的許可來叫用 Lambda 函數時，叫用可能會失敗。如需詳細資訊，請參閱《Amazon EventBridge 使用者指南》中的[事件重試政策和使用無效字母佇列](#)。

Note

[AWS::Serverless::Function](#) 資源類型具有類似的資料類型 DeadLetterQueue，可處理成功調用目標 Lambda 函數後發生的失敗。這些失敗類型的範例包括 Lambda 限流，或 Lambda 目標函數傳回的錯誤。如需 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

CloudFormation 相容性：此屬性類似於 `AWS::Events::RuleTarget` 資料類型的 [DeadLetterConfig](#) 屬性。此屬性的 AWS SAM 版本包含其他子屬性，以防您想要為您 AWS SAM 建立無效字母佇列。

Description

規則的說明。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 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 屬性，但不能同時指定兩者。

類型：布林值

必要：否

CloudFormation 相容性：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

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Events::Rule Target 資源的 [Input](#) 屬性。

Name

規則的名稱。如果您未指定名稱，CloudFormation 會產生唯一的實體 ID，並將該 ID 用於規則名稱。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 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 EventBridge 使用者指南》中的[事件重試政策和使用無效字母佇列](#)。

類型：[RetryPolicy](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Events::Rule Target` 資料類型的 [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

CloudFormation 相容性：此屬性會直接傳遞至 `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` 屬性，但不能同時指定兩者。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Events::Rule` 資源的 [State](#) 屬性。

範例

CloudWatch 排程事件

CloudWatch 排程事件範例

YAML

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

DeadLetterConfig

用來指定 Amazon Simple Queue Service (Amazon SQS) 佇列的物件，其中 EventBridge 會在失敗的目標調用後傳送事件。例如，將事件傳送至不存在的 Lambda 事件函數時，呼叫可能會失敗，或沒有足夠的許可來叫用 Lambda 函數。如需詳細資訊，請參閱《Amazon EventBridge 使用者指南》中的[事件重試政策和使用無效字母佇列](#)。

Note

[AWS::Serverless::Function](#) 資源類型具有類似的資料類型，DeadLetterQueue 可處理成功調用目標 Lambda 函數後發生的失敗。此類故障的範例包括 Lambda 限流，或 Lambda 目標函數傳回的錯誤。如需函數 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
```

Properties

Arn

指定為無效字母佇列目標之 Amazon SQS 佇列的 Amazon Resource Name (ARN)。

 Note

指定 Type 屬性或 Arn 屬性，但不能同時指定兩者。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Events::Rule DeadLetterConfig` 資料類型的 [Arn](#) 屬性。

QueueLogicalId

Type 指定 時 AWS SAM 建立的無效字母佇列自訂名稱。

 Note

如果未設定 Type 屬性，則會忽略此屬性。

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

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，並連接必要的 [資源型政策](#)，以授予將事件傳送至佇列的規則資源許可。

 Note

指定 Type 屬性或 Arn 屬性，但不能同時指定兩者。

有效值：SQS

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

範例

DeadLetterConfig

DeadLetterConfig

YAML

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

ScheduleV2

描述ScheduleV2事件來源類型的物件，會將無伺服器函數設定為排程觸發的 Amazon EventBridge 排程器事件的目標。如需詳細資訊，請參閱[什麼是 Amazon EventBridge 排程器？](#) 《EventBridge 排程器使用者指南》中的 。

AWS Serverless Application Model (AWS SAM) 會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
```

Properties

DeadLetterConfig

設定 EventBridge 在目標呼叫失敗後傳送事件的 Amazon Simple Queue Service (Amazon SQS) 佇列。例如，將事件傳送至不存在的 Lambda 事件函數時，或 EventBridge 沒有足夠的許可來叫用 Lambda 函數時，叫用可能會失敗。如需詳細資訊，請參閱《[EventBridge 排程器使用者指南](#)》中的[為 EventBridge 排程器設定無效字母佇列](#)。EventBridge

Note

[AWS::Serverless::Function](#) 資源類型具有類似的資料類型 DeadLetterQueue，可處理成功調用目標 Lambda 函數後發生的失敗。這些失敗類型的範例包括 Lambda 限流，或 Lambda 目標函數傳回的錯誤。如需函數 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

CloudFormation 相容性：此屬性類似於 `AWS::Scheduler::ScheduleTarget` 資料類型的 [DeadLetterConfig](#) 屬性。此屬性的 AWS SAM 版本包含其他子屬性，以防您想要為您 AWS SAM 建立無效字母佇列。

Description

排程的描述。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Scheduler::Schedule` 資源的 [EndDate](#) 屬性。

FlexibleTimeWindow

允許在其中叫用排程的時段組態。

類型：[FlexibleTimeWindow](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Scheduler::Schedule` 資源的 [FlexibleTimeWindow](#) 屬性。

GroupName

要與此排程建立關聯的排程群組名稱。如果未定義，則會使用預設群組。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Scheduler::Schedule Target` 資源的 [Input](#) 屬性。

KmsKeyArn

KMS 金鑰的 ARN，將用於加密客戶資料。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Scheduler::Schedule` 資源的 [KmsKeyArn](#) 屬性。

Name

排程的名稱。如果您未指定名稱，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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Scheduler::Schedule` 資源的 [Name](#) 屬性。

OmitName

根據預設，`AWS SAM` 會產生並使用 *<Function-logical-ID><event-source-name>* 格式的排程名稱。將此屬性設定為 `true`，讓 CloudFormation 產生唯一的實體 ID，並將其用於排程名稱。

類型：布林值

必要：否

預設：false

CloudFormation 相容性：此屬性對 是唯一的 `AWS SAM`，並且沒有同等 CloudFormation 屬性。

PermissionsBoundary

用來設定角色許可邊界的政策 ARN。

Note

如果 `PermissionsBoundary` 已定義，`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

CloudFormation 相容性：此屬性會直接傳遞至 `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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Scheduler::Schedule Target` 資料類型的 [RetryPolicy](#) 屬性。

RoleArn

調用排程時 `EventBridge` 排程器用於目標的 IAM 角色 ARN。

類型：[RoleArn](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Scheduler::Schedule Target` 資料類型的 [RoleArn](#) 屬性。

ScheduleExpression

排程表達式，可決定排程器排程事件執行的時間和頻率。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Scheduler::Schedule` 資源的 [ScheduleExpressionTimezone](#) 屬性。

StartDate

排程開始調用目標的日期，以 UTC 為單位。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

CloudFormation 相容性：此屬性會直接傳遞至 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

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Scheduler::Schedule 資源的 [State](#) 屬性。

範例

定義 ScheduleV2 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
```

Note

ScheduleV2 產生的實體 ID 不包含堆疊名稱。

SelfManagedKafka

描述SelfManagedKafka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[使用 AWS Lambda 搭配自我管理的 Apache Kafka](#)。

AWS Serverless Application Model (AWS SAM) 會在設定此事件類型時產生 [AWS::Lambda::EventSourceMapping](#) 資源。

若要使用結構描述登錄檔，您需要定義函數的特定 IAM 角色許可。如需所需組態的範例，請參閱[使用 IAM 角色完成設定](#)。

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
BisectBatchOnFunctionError: Boolean
ConsumerGroupId: String
DestinationConfig: DestinationConfig
Enabled: Boolean
FilterCriteria: FilterCriteria
KafkaBootstrapServers: List
FunctionResponseType: List
KmsKeyArn: String
LoggingConfig: LoggingConfig
```

```
MaximumRecordAgeInSeconds: Integer  
MaximumRetryAttempts: Integer  
MetricsConfig: MetricsConfig  
ProvisionedPollerConfig: ProvisionedPollerConfig  
SchemaRegistryConfig: SchemaRegistryConfig  
SourceAccessConfigurations: SourceAccessConfigurations  
StartingPosition: String  
StartingPositionTimestamp: Double  
Topics: List
```

Properties

BatchSize

Lambda 從串流提取並傳送至函數的每個批次中的記錄數目上限。

類型：整數

必要：否

預設值：100

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [BatchSize](#) 屬性。

下限：1

上限：10000

BisectBatchOnFunctionError

如果函數傳回錯誤，請將批次分割為兩個，然後重試。

類型：布林值

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [BisectBatchOnFunctionError](#) 屬性。

ConsumerGroupId

設定如何從 Kafka 主題讀取事件的字串。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [SelfManagedKafkaEventSourceConfig](#) 屬性。

DestinationConfig

組態物件，指定在 Lambda 處理過後事件的目標。

使用此屬性可指定從自我管理 Kafka 事件來源呼叫失敗的目的地。

類型：[DestinationConfig](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [Enabled](#) 屬性。

FilterCriteria

定義判斷 Lambda 是否應處理事件之條件的物件。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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [FilterCriteria](#) 屬性。

KafkaBootstrapServers

Kafka 代理程式的引導伺服器清單。包含連接埠，例如 `broker.example.com:xxxx`

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

FunctionResponseTypes

目前套用至事件來源映射的回應類型清單。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

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Lambda::EventSourceMapping 資源的 [FunctionResponseTypes](#) 屬性。

KmsKeyArn

金鑰的 Amazon Resource Name (ARN)，用於加密與此事件相關的資訊。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Lambda::EventSourceMapping 資源的 [KmsKeyArn](#) 屬性。

LoggingConfig

事件來源的記錄組態。

類型：[LoggingConfig](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Lambda::EventSourceMapping 資源的 [LoggingConfig](#) 屬性。

MaximumRecordAgeInSeconds

Lambda 傳送給函數進行處理的記錄最長存留期。

類型：整數

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [MaximumRecordAgeInSeconds](#) 屬性。

MetricsConfig

事件來源的指標組態。

類型：[MetricsConfig](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [MetricsConfig](#) 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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [MaximumRetryAttempts](#) 屬性。

ProvisionedPollerConfig

用於增加用於計算事件來源映射之輪詢器數量的組態。此組態允許最少 1 個輪詢器，最多 2000 個輪詢器。如需範例，請參閱 [ProvisionedPollerConfig 範例](#)

類型：[ProvisionedPollerConfig](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [ProvisionedPollerConfig](#) 屬性。

SchemaRegistryConfig

搭配自我管理 Kafka 事件來源使用結構描述登錄檔的組態。

Note

此功能 `ProvisionedPollerConfig` 需要設定。

類型：[SchemaRegistryConfig](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [SelfManagedKafkaEventSourceConfig](#) 屬性。

SourceAccessConfigurations

驗證通訊協定、VPC 元件和 OAuth 2.0 中繼資料的陣列，可保護和定義您的事件來源。每個項目都是具有Type欄位的物件，對於大多數類型而言，則為URI欄位。如需類型的完整清單和每個類型URI所需的值，請參閱AWS CloudFormation 《使用者指南》中的 [SourceAccessConfiguration](#)。

AWS SAM 在將此屬性傳遞至 Lambda 之前，會套用下列驗證：

- 您必須指定一個以上的身分驗證類型。如果您指定一個以上的身分驗證類型，會 AWS SAM 傳回錯誤。
- 您必須指定 VPC_SUBNET和 VPC_SECURITY_GROUP together。如果您指定一個沒有另一個的錯誤，則 AWS SAM 會傳回錯誤。
- 您必須為除 IAM_AUTH和 以外的每種類型指定URI一個值IAM_OAUTHBEARER_AUTH，這不需要 URI。只會 AWS SAM 檢查該值是否存在，並在不解譯它的情況下通過它。當您部署時，Lambda 會驗證 值和類型組合。

AWS SAM 會為您指定的每個秘密 ARN 新增secretsmanager:GetSecretValue許可，包括SERVER_ROOT_CA_CERTIFICATE秘密，以及 Lambda 在您指定 VPC 項目時所需的 Amazon EC2 網路介面許可。對於 IAM_AUTH和 IAM_OAUTHBEARER_AUTH，不會 AWS SAM 新增叢集或字符許可。自行將這些許可授予函數的執行角色。當這些都不適用時，例如IAM_AUTH沒有 VPC 項目時，不會將內嵌政策 AWS SAM 新增至執行角色。如需所需許可，請參閱《AWS Lambda 開發人員指南》中的在 [Lambda 中設定叢集身分驗證方法](#)。

有效值：BASIC_AUTH | CLIENT_CERTIFICATE_TLS_AUTH | IAM_AUTH | IAM_OAUTHBEARER_AUTH | OAUTHBEARER_AUDIENCE | OAUTHBEARER_AUTH | OAUTHBEARER_IDENTITY_POOL | OAUTHBEARER_LOGICAL_CLUSTER | OAUTHBEARER_SCOPE | SASL_SCRAM_256_AUTH | SASL_SCRAM_512_AUTH | SERVER_ROOT_CA_CERTIFICATE | VPC_SECURITY_GROUP | VPC_SUBNET

類型：[SourceAccessConfiguration](#) 清單

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [SourceAccessConfigurations](#) 屬性。

StartingPosition

要從中開始讀取的串流位置。

- `AT_TIMESTAMP` – 指定從中開始讀取記錄的時間。
- `LATEST` – 唯讀新記錄。
- `TRIM_HORIZON` – 處理所有可用的記錄。

有效值：`AT_TIMESTAMP` | `LATEST` | `TRIM_HORIZON`

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `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` `StartingPosition` 何時指定為 `AT_TIMESTAMP`。

類型：Double

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [StartingPositionTimestamp](#) 屬性。

Topics

Kafka 主題名稱。

類型：清單

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [Topics](#) 屬性。

範例

使用 IAM 角色完成設定

下列範例顯示完整的設定，包括使用結構描述登錄所需的 IAM 角色組態：

```
Parameters:
  PreCreatedSubnetOne:
    Type: String
  PreCreatedSubnetTwo:
    Type: String

Resources:
  MyLambdaExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Action: [sts:AssumeRole]
            Effect: Allow
            Principal:
              Service: [lambda.amazonaws.com]
      Policies:
        - PolicyName: KafkaAuthPolicy
          PolicyDocument:
            Statement:
              - Action: [secretsmanager:GetSecretValue, kms:Decrypt]
                Effect: "Allow"
                Resource: ['arn:aws:secretsmanager:us-west-2:123456789012:secret:kafkaSecret-*****',
                          'arn:aws:kms:us-west-2:123456789012:key/keyId']
        - PolicyName: ENIPolicy
          PolicyDocument:
            Statement:
              - Action: [ec2:CreateNetworkInterface,
                        ec2:DescribeNetworkInterfaces, ec2:DescribeVpcs,
                        ec2>DeleteNetworkInterface,
                        ec2:DescribeSubnets, ec2:DescribeSecurityGroups]
                Effect: Allow
                Resource: '*'
        - PolicyName: SchemaRegistryPolicy
          PolicyDocument:
            Statement:
```

```

    - Action: [glue:GetRegistry]
      Effect: Allow
      Resource: 'arn:aws:glue:{region}:{account-id}:registry/{registry-name}'
  - PolicyName: SchemaVersionsPolicy
    PolicyDocument:
      Statement:
        - Action: [glue:GetSchemaVersions]
          Effect: Allow
          Resource: '*'
    ManagedPolicyArns:
      - !Sub arn:${AWS::Partition}:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole
    Tags:
      - {Value: SAM, Key: lambda:createdBy}

MyKafkaProcessor:
  Type: AWS::Serverless::Function
  Properties:
    Runtime: nodejs18.x
    Handler: index.handler
    CodeUri: ${codeuri}
    Role:
      Fn::GetAtt: [MyLambdaExecutionRole, Arn]
  Events:
    SelfManagedKafkaEvent:
      Type: SelfManagedKafka
      Properties:
        KafkaBootstrapServers:
          - my-kafka-broker-1:9092
          - my-kafka-broker-2:9092
        Topics:
          - SchemaRegistryTestTopic
        StartingPosition: LATEST
        SourceAccessConfigurations:
          - Type: VPC_SUBNET
            URI: subnet:subnet-12345678
          - Type: VPC_SECURITY_GROUP
            URI: security_group:sg-12345678
          - Type: BASIC_AUTH
            URI: !Sub arn:${AWS::Partition}:secretsmanager:us-
west-2:123456789012:secret:my-path/my-secret-name-1a2b3c
        ProvisionedPollerConfig:
          MinimumPollers: 1
        SchemaRegistryConfig:

```

```

    AccessConfigs:
      - Type: BASIC_AUTH
        URI: !Sub arn:${AWS::Partition}:secretsmanager:us-
west-2:123456789012:secret:my-path/my-secret-name-1a2b3c
    SchemaValidationConfigs:
      - Attribute: KEY
        EventRecordFormat: JSON
        SchemaRegistryURI: !Sub arn:${AWS::Partition}:glue:us-
west-2:123456789012:registry/myregistry

```

ProvisionedPollerConfig 範例

```

ProvisionedPollerConfig:
  MinimumPollers: 1
  MaximumPollers: 200

```

自我管理的 Kafka 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

```

使用 OAuth 2.0 身分驗證的自我管理 Kafka 事件來源

下列範例使用 SASL/OAUTHBEARER 進行驗證。OAUTHBEARER_AUTH 項目指向存放 OAuth 2.0 登入資料的秘密，而每個中繼資料項目都有 Lambda 傳遞給您的身分提供者或代理程式的常值。

```

Events:

```

```

SelfManagedKafkaEvent:
  Type: SelfManagedKafka
  Properties:
    KafkaBootstrapServers:
      - abc.xyz.com:9092
    Topics:
      - MyKafkaTopic
    StartingPosition: LATEST
    ConsumerGroupId: consumergroup1
    SourceAccessConfigurations:
      - Type: OAUTHBEARER_AUTH
        URI: arn:aws:secretsmanager:us-west-2:123456789012:secret:my-path/my-oauth-secret-1a2b3c
      - Type: OAUTHBEARER_SCOPE
        URI: read write
      - Type: OAUTHBEARER_AUDIENCE
        URI: https://api.example.com
      - Type: SERVER_ROOT_CA_CERTIFICATE
        URI: arn:aws:secretsmanager:us-west-2:123456789012:secret:my-path/my-server-root-ca-cert-7p8q9r
      - Type: VPC_SUBNET
        URI: subnet:subnet-12345678
      - Type: VPC_SECURITY_GROUP
        URI: security_group:sg-12345678

```

使用 IAM 身分驗證的自我管理 Kafka 事件來源

下列範例使用 IAM 進行身分驗證。IAM_AUTH 項目不需要URI任何欄位，因為 Lambda 會使用函數執行角色的登入資料來簽署每個連線。將 Lambda 所需的叢集許可授予您的執行角色，因為 AWS SAM 不會新增它們。

```

Events:
  SelfManagedKafkaEvent:
    Type: SelfManagedKafka
    Properties:
      KafkaBootstrapServers:
        - abc.xyz.com:9092
      Topics:
        - MyKafkaTopic
      StartingPosition: LATEST
      ConsumerGroupId: consumergroup1
      SourceAccessConfigurations:
        - Type: IAM_AUTH

```

- Type: VPC_SUBNET
URI: subnet:subnet-12345678
- Type: VPC_SECURITY_GROUP
URI: security_group:sg-12345678

使用 IAM 和 OAuth 2.0 身分驗證的自我管理 Kafka 事件來源

下列範例使用 驗證 IAM_OAUTHBEARER_AUTH。此項目不需要URI任何欄位，但您必須將其與項目配對，而該OAUTHBEARER_AUDIENCE項目URI是代理程式預期的對象。Lambda 請求執行角色的短期 Web 身分字串，需要 sts:GetWebIdentityToken許可，且您的代理程式必須信任 AWS 發行者並接受對象值。

```
Events:
  SelfManagedKafkaEvent:
    Type: SelfManagedKafka
    Properties:
      KafkaBootstrapServers:
        - abc.xyz.com:9092
      Topics:
        - MyKafkaTopic
      StartingPosition: LATEST
      ConsumerGroupId: consumergroup1
      SourceAccessConfigurations:
        - Type: IAM_OAUTHBEARER_AUTH
        - Type: OAUTHBEARER_AUDIENCE
          URI: kafka-cluster
        - Type: VPC_SUBNET
          URI: subnet:subnet-12345678
        - Type: VPC_SECURITY_GROUP
          URI: security_group:sg-12345678
```

具有 AWS Glue 結構描述登錄檔的自我管理 Kafka 事件來源

以下是使用 AWS Glue 結構描述登錄檔設定SelfManagedKafka的事件來源類型範例。

```
Events:
  SelfManagedKafkaEvent:
    Type: SelfManagedKafka
    Properties:
      KafkaBootstrapServers:
        - abc.xyz.com:9092
      Topics:
```

```

    - SchemaRegistryTestTopic
StartingPosition: LATEST
ProvisionedPollerConfig:
  MinimumPollers: 1
SchemaRegistryConfig:
  SchemaRegistryURI: !Sub arn:${AWS::Partition}:glue:us-
west-2:123456789012:registry/myregistry
  EventRecordFormat: JSON
  SchemaValidationConfigs:
    - Attribute: KEY
    - Attribute: VALUE
SourceAccessConfigurations:
  - Type: VPC_SUBNET
    URI: subnet:subnet-12345678
  - Type: VPC_SECURITY_GROUP
    URI: security_group:sg-12345678

```

具有 Confluent Schema Registry 的自我管理 Kafka 事件來源

以下是使用 Confluent Schema Registry 設定SelfManagedKafka的事件來源類型範例。

```

Events:
  SelfManagedKafkaEvent:
    Type: SelfManagedKafka
    Properties:
      KafkaBootstrapServers:
        - abc.xyz.com:9092
      Topics:
        - SchemaRegistryTestTopic
      StartingPosition: LATEST
      ProvisionedPollerConfig:
        MinimumPollers: 1
      SchemaRegistryConfig:
        SchemaRegistryURI: https://my-schema-registry.confluent.cloud
        AccessConfigs:
          - Type: BASIC_AUTH
            URI: !Sub arn:${AWS::Partition}:secretsmanager:us-
west-2:123456789012:secret:my-secret
        EventRecordFormat: JSON
        SchemaValidationConfigs:
          - Attribute: KEY
          - Attribute: VALUE
      SourceAccessConfigurations:
        - Type: VPC_SUBNET

```

```
URI: subnet:subnet-12345678
- Type: VPC_SECURITY_GROUP
  URI: security_group:sg-12345678
- Type: BASIC_AUTH
  URI: !Sub arn:${AWS::Partition}:secretsmanager:us-
west-2:123456789012:secret:kafka-secret
```

SNS

描述 SNS 事件來源類型的物件。

設定此事件類型時，SAM 會產生 [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
```

Properties

FilterPolicy

指派給訂閱的篩選條件政策 JSON。如需詳細資訊，請參閱《Amazon Simple Notification Service API 參考》中的 [GetSubscriptionAttributes](#)。

類型：[SnsFilterPolicy](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::SNS::Subscription` 資源的 [FilterPolicy](#) 屬性。

FilterPolicyScope

此屬性可讓您使用下列其中一個字串值類型來選擇篩選範圍：

- `MessageAttributes` – 篩選條件會套用至訊息屬性。

- `MessageBody` – 篩選條件會套用至訊息內文。

類型：字串

必要：否

預設：MessageAttributes

CloudFormation 相容性：此屬性會直接傳遞至 `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。

如需再驅動政策和無效字母佇列的詳細資訊，請參閱 [《Amazon Simple Queue Service 開發人員指南》](#) 中的 [Amazon SQS 無效字母佇列](#)。

類型：Json

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `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，CloudFormation 會使用發起人的區域做為預設值。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::SNS::Subscription` 資源的 [Region](#) 屬性。

SqsSubscription

將此屬性設定為 `true`，或指定 `SqsSubscriptionObject` 在 SQS 佇列中啟用批次 SNS 主題通知。將此屬性設定為 `true` 建立新的 SQS 佇列，而指定 `SqsSubscriptionObject` 會使用現有的 SQS 佇列。

類型：布林值 | [SqsSubscriptionObject](#)

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

Topic

要訂閱的主題 ARN。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `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
```

Properties

BatchSize

單一批次中針對 SQS 佇列擷取的項目數量上限。

類型：字串

必要：否

預設值：10

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

Enabled

停用 SQS 事件來源映射以暫停輪詢和調用。

類型：布林值

必要：否

預設：True

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

QueueArn

指定現有的 SQS 佇列 arn。

類型：字串

必要：是

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

QueuePolicyLogicalId

為 [AWS::SQS::QueuePolicy](#) 資源提供自訂 logicalId 名稱。

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

QueueUrl

指定與 QueueArn 屬性相關聯的佇列 URL。

類型：字串

必要：是

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

範例

SNS 事件的現有 SQS

將現有的 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 開發人員指南》中的[搭配使用 AWS Lambda 與 Amazon SQS](#)。

設定此事件類型時，SAM 會產生 [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
KmsKeyArn: String
MaximumBatchingWindowInSeconds: Integer
MetricsConfig: MetricsConfig
ProvisionedPollerConfig: ProvisionedPollerConfig
Queue: String
ScalingConfig: ScalingConfig
```

Properties

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

CloudFormation 相容性：此屬性會直接傳遞至 [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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [Enabled](#) 屬性。

FilterCriteria

定義判斷 Lambda 是否應處理事件之條件的物件。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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [FilterCriteria](#) 屬性。

FunctionResponseTypes

目前套用至事件來源映射的回應類型清單。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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [FunctionResponseTypes](#) 屬性。

KmsKeyArn

金鑰的 Amazon Resource Name (ARN)，用於加密與此事件相關的資訊。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [KmsKeyArn](#) 屬性。

MaximumBatchingWindowInSeconds

叫用函數之前收集記錄的時間上限，以秒為單位。

類型：整數

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [MaximumBatchingWindowInSeconds](#) 屬性。

MetricsConfig

選擇加入組態，以取得擷取每個處理階段之事件來源映射的增強指標。如需範例，請參閱 [MetricsConfig 事件](#)。

類型：[MetricsConfig](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [MetricsConfig](#) 屬性。

ProvisionedPollerConfig

用於增加用於計算事件來源映射之輪詢器數量的組態。此組態允許最少 2 個輪詢器，最多 2000 個輪詢器。如需範例，請參閱 [ProvisionedPollerConfig 範例](#)。

類型：[ProvisionedPollerConfig](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [ProvisionedPollerConfig](#) 屬性。

Queue

佇列的 ARN。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [EventSourceArn](#) 屬性。

ScalingConfig

SQS 輪詢器的擴展組態，以控制調用速率並設定並行調用上限。

Type (類型) : [ScalingConfig](#)

必要 : 否

CloudFormation 相容性 : 此屬性會直接傳遞至 `AWS::Lambda::EventSourceMapping` 資源的 [ScalingConfig](#) 屬性。

範例

MetricsConfig 事件

以下是 資源的範例，該資源使用 `MetricsConfig` 屬性來擷取其事件來源映射的每個處理階段。

```
Resources:
  FilteredEventsFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: s3://sam-demo-bucket/metricsConfig.zip
      Handler: index.handler
      Runtime: nodejs16.x
      Events:
        KinesisStream:
          Type: Kinesis
          Properties:
            Stream: !GetAtt KinesisStream.Arn
            StartingPosition: LATEST
            MetricsConfig:
              Metrics:
                - EventCount
```

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"]}'
```

為您的 SQS 佇列設定部分批次報告

```
Events:
  SQSEvent:
    Type: SQS
    Properties:
      Enabled: true
      FunctionResponseTypes:
        - ReportBatchItemFailures
      Queue: !GetAtt MySqsQueue.Arn
      BatchSize: 10
```

具有已設定擴展之 SQS 事件的 Lambda 事件函數

```
MyFunction:
  Type: AWS::Serverless::Function
  Properties:
    ...
  Events:
    MySQSEvent:
      Type: SQS
      Properties:
        ...
        ScalingConfig:
          MaximumConcurrency: 10
```

ProvisionedPollerConfig 範例

```
MyFunction:
  Type: AWS::Serverless::Function
  Properties:
    Handler: index.handler
    Runtime: nodejs18.x
    Timeout: 30
  Events:
    SQSEvent:
      Type: SQS
      Properties:
        Queue: !GetAtt MyQueue.Arn
        BatchSize: 10
        Enabled: True
        ProvisionedPollerConfig:
```


MaximumPollers: 300

MinimumPollers: 10

FunctionCode

Lambda 事件函數的[部署套件](#)。

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
Key: String
StorageMode: String
Version: String
```

Properties

Bucket

與函數位於相同 AWS 區域中的 Amazon S3 儲存貯體。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Lambda::Function Code 資料類型的 [S3Bucket](#) 屬性。

Key

部署套件的 Amazon S3 金鑰。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Lambda::Function Code 資料類型的 [S3Key](#) 屬性。

StorageMode

控制 Lambda 如何存放部署套件。

- **COPY** – Lambda 會將部署套件的副本上傳至 Lambda 受管儲存。您可以在 Lambda 建立函數後刪除 Amazon S3 物件。
- **REFERENCE** – Lambda 會參考來自 Amazon S3 儲存貯體的部署套件，而不會儲存複本。物件必須保留在原處，Lambda 必須在函數的生命週期內保留其存取權。

若要使用 **REFERENCE**，請在 Amazon S3 儲存貯體上啟用版本控制，並授予 Lambda 服務主體存取物件的權限。如需詳細資訊，請參閱 [《開發人員指南》中的自我管理 Amazon S3 程式碼儲存](#)。

AWS Lambda

類型：字串

有效值：COPY | REFERENCE

必要：否

預設：COPY

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::Function` 資料類型的 [S3ObjectStorageMode](#) 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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::Function` 資料類型的 [S3ObjectVersion](#) 屬性。

範例

FunctionCode

CodeUri：函數程式碼範例

YAML

```
CodeUri:
  Bucket: sam-s3-demo-bucket-name
  Key: mykey-name
```

```
Version: 121212
```

自我管理的 Amazon S3 程式碼儲存

以下CodeUri參考來自您自己的 Amazon S3 儲存貯體的部署套件，而不是將其複製到 Lambda 受管儲存體。

YAML

```
CodeUri:
  Bucket: amzn-s3-demo-bucket-name
  Key: mykey-name
  Version: 121212
  StorageMode: REFERENCE
```

FunctionUrlConfig

使用指定的組態參數建立 AWS Lambda 函數 URL。Lambda 函數 URL 是 HTTPS 端點，可用來叫用函數。

根據預設，您建立的函數 URL 會使用 Lambda 事件函數的\$LATEST版本。如果您AutoPublishAlias為 Lambda 函數指定，端點會連線至指定的函數別名。

如需詳細資訊，請參閱《AWS Lambda 開發人員指南》中的[Lambda 函數 URLs](#)。

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
```

Properties

AuthType

函數 URL 的授權類型。若要使用 AWS Identity and Access Management (IAM) 來授權請求，請將設定為 AWS_IAM。針對開放存取，請將設定為 NONE。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::Url` 資源的 [Cors](#) 屬性。

InvokeMode

將叫用函數 URL 的模式。若要讓您的函數在呼叫完成後傳回回應，請將設定為 `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 相容性：此屬性會直接傳遞至 `AWS::Lambda::Url` 資源的 [InvokeMode](#) 屬性。

範例

函數 URL

下列範例會使用函數 URL 建立 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
```

CapacityProviderConfig

設定將附加發佈版本函數的容量提供者。這可讓函數在由 Lambda 管理的客戶擁有 EC2 執行個體上執行。

Note

此組態會決定函數的運算類型，且需要在第一個函數部署期間指定。在建立函數資源之後，就無法新增或移除它。

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
ExecutionEnvironmentMemoryGiBPerVCpu: Float
PerExecutionEnvironmentMaxConcurrency: Integer
```

Properties

Arn

用於此函數之容量提供者的 ARN。

類型：字串

必要：是

CloudFormation 相容性：此屬性對 SAM 是唯一的。

ExecutionEnvironmentMemoryGiBPerVCpu

每個執行環境的記憶體（以 GiB 為單位）與 vCPU 的比率。

Note

每個 CPU 的記憶體比率不得超過函數的總記憶體 2048MB。支援的memory-to-CPU 比率為每個 CPU 2GB, 4GB 或 8GB。

類型：Float

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::Function` 資源的 [ExecutionEnvironmentMemoryGiBPerVCpu](#) 屬性。

PerExecutionEnvironmentMaxConcurrency

每個執行環境（沙盒）的並行執行數目上限。

類型：整數

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::Function` 資源的 [PerExecutionEnvironmentMaxConcurrency](#) 屬性。

範例

容量提供者組態

下列範例顯示參考容量提供者資源的容量提供者組態。

```
CapacityProviderConfig:
  Arn: !GetAtt MyCapacityProvider.Arn
  ExecutionEnvironmentMemoryGiBPerVCpu: 4.0
```

```
PerExecutionEnvironmentMaxConcurrency: 100
```

FunctionScalingConfig

設定 Lambda 事件函數版本的擴展行為，控制可建立的執行環境（沙盒）數量。此組態同時適用於 \$LATEST.PUBLISHED 和數值函數版本。

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

```
MinExecutionEnvironments: Integer  
MaxExecutionEnvironments: Integer
```

Properties

MinExecutionEnvironments

要為函數版本維護的執行環境數目下限。

類型：整數

必要：否

預設：3

下限：0

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::Function` 資源的 [MinExecutionEnvironments](#) 屬性。

MaxExecutionEnvironments

可針對函數版本建立的執行環境數目上限。

類型：整數

必要：否

預設：3

下限：0

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::Function` 資源的 [MaxExecutionEnvironments](#) 屬性。

範例

函數擴展組態

下列範例顯示具有最小和最大執行環境的函數擴展組態。

```
FunctionScalingConfig:
  MinExecutionEnvironments: 5
  MaxExecutionEnvironments: 100
```

AWS::Serverless::GraphQLApi

使用 AWS Serverless Application Model (AWS SAM) `AWS::Serverless::GraphQLApi` 資源類型為您的無伺服器應用程式建立和設定 AWS AppSync GraphQL API。

若要進一步了解 AWS AppSync，請參閱[什麼是 AWS AppSync？](#) 《AWS AppSync 開發人員指南》中的。

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
```

Properties

ApiKeys

建立唯一金鑰，可用於執行需要 API 金鑰 GraphQL 的操作。

類型：[ApiKeys](#)

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

Auth

設定 GraphQL API 的身分驗證。

類型：[Auth](#)

必要：是

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

Cache

CreateApiCache 操作的輸入。

類型：[AWS::AppSync::ApiCache](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 [AWS::AppSync::ApiCache](#) 資源。

DataSources

在 中建立 函數的資料來源 AWS AppSync 以連線至 。 AWS SAM 支援 Amazon DynamoDB 和 AWS Lambda 資料來源。

類型：[DataSource](#)

必要：是

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

DomainName

您 GraphQL API 的自訂網域名稱。

類型：[AWS::AppSync::DomainName](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 [AWS::AppSync::DomainName](#) resource. AWS SAM automatically 產生 [AWS::AppSync::DomainNameApiAssociation](#) 資源。

Functions

在 GraphQL APIs中設定函數以執行特定操作。

類型：[Function](#)

必要：是

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

Logging

設定 GraphQL API 的 Amazon CloudWatch 記錄。

如果您未指定此屬性，AWS SAM 將產生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

CloudFormation 相容性：此屬性會直接傳遞至 [AWS::AppSync::GraphQLApi](#) 資源的 [LogConfig](#) 屬性。

LogicalId

API 的唯一名稱GraphQL。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 AWS::AppSync::GraphQLApi 資源的 [Name](#) 屬性。

Name

API 的名稱GraphQL。指定此屬性以覆寫LogicalId值。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::AppSync::GraphQLApi 資源的 [Name](#) 屬性。

Resolvers

為您的 GraphQL API 欄位設定解析程式。AWS SAM 支援[JavaScript管道解析程式](#)。

類型：[解析程式](#)

必要：是

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

SchemaInline

格式結構GraphQL描述的文字表示SDL。

類型：字串

必要：有條件限制。您必須指定 SchemaInline或 SchemaUri。

CloudFormation 相容性：此屬性會直接傳遞至 AWS::AppSync::GraphQLSchema 資源的 [Definition](#) 屬性。

SchemaUri

結構描述的 Amazon Simple Storage Service (Amazon S3) 儲存貯體 URI 或本機資料夾的路徑。

如果您指定本機資料夾的路徑，CloudFormation 需要先將檔案上傳至 Amazon S3，才能部署。您可以使用 AWS SAM CLI 來促進此程序。如需詳細資訊，請參閱[如何在部署時 AWS SAM 上傳本機檔案](#)。

類型：字串

必要：有條件限制。您必須指定 `SchemaInline` 或 `SchemaUri`。

CloudFormation 相容性：此屬性會直接傳遞至 `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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::AppSync::GraphQLApi` 資源的 [Tag](#) 屬性。

XrayEnabled

指出是否要為此資源使用 [AWS X-Ray 追蹤](#)。

類型：布林值

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::AppSync::GraphQLApi` 資源的 [XrayEnabled](#) 屬性。

傳回值

如需傳回值的清單，請參閱[CloudFormation 《使用者指南AWS::Serverless::GraphQLApi》](#)中的。

範例

GraphQL API 搭配 DynamoDB 資料來源

在此範例中，我們會建立使用 DynamoDB 資料表做為資料來源的 GraphQL API。

`schema.graphql`

```
schema {  
  query: Query  
  mutation: Mutation  
}  
  
type Query {  
  getPost(id: String!): Post  
}  
  
type Mutation {  
  addPost(author: String!, title: String!, content: String!): Post!  
}  
  
type Post {  
  id: String!  
  author: String  
  title: String  
  content: String  
  ups: Int!  
  downs: Int!  
  version: Int!  
}
```

template.yaml

```
AWS::Serverless::GraphQLApi  
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;
  }
```

以下是我們的解析程式碼：

getPost.js

```
export function request(ctx) {
  return {};
}

export function response(ctx) {
  return ctx.prev.result;
}
```

GraphQL 將 Lambda 事件函數做為資料來源的 API

在此範例中，我們會建立使用 Lambda 事件函數做為資料來源的 GraphQL API。

template.yaml

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyLambdaFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.handler
      Runtime: nodejs20.x
      CodeUri: ./lambda

  MyGraphQLAPI:
    Type: AWS::Serverless::GraphQLApi
    Properties:
      Name: MyApi
      SchemaUri: ./gql/schema.gql
      Auth:
        Type: API_KEY
```



```

ApiKeys:
  MyApiKey:
    Description: my api key
DataSources:
  Lambda:
    MyLambdaDataSource:
      FunctionArn: !GetAtt MyLambdaFunction.Arn
Functions:
  lambdaInvoker:
    Runtime:
      Name: APPSYNC_JS
      Version: 1.0.0
    DataSource: MyLambdaDataSource
    CodeUri: ./gql/invoke.js
Resolvers:
  Mutation:
    addPost:
      Runtime:
        Name: APPSYNC_JS
        Version: 1.0.0
      Pipeline:
        - lambdaInvoker
  Query:
    getPost:
      Runtime:
        Name: APPSYNC_JS
        Version: 1.0.0
      Pipeline:
        - lambdaInvoker

```

```

Outputs:
  MyGraphQLAPI:
    Description: AppSync API
    Value: !GetAtt MyGraphQLAPI.GraphQLUrl
  MyGraphQLAPIMyApiKey:
    Description: API Key for authentication
    Value: !GetAtt MyGraphQLAPIMyApiKey.ApiKey

```

schema.graphql

```

schema {
  query: Query
  mutation: Mutation
}

```

```
}  
type Query {  
  getPost(id: ID!): Post  
}  
type Mutation {  
  addPost(id: ID!, author: String!, title: String, content: String): Post!  
}  
type Post {  
  id: ID!  
  author: String!  
  title: String  
  content: String  
  ups: Int  
  downs: Int  
}
```

以下是我們的 函數：

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

建立唯一金鑰，可用於執行需要 API 金鑰GraphQL的操作。

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
```

Properties

ApiKeyId

API 金鑰的唯一名稱。指定 以覆寫 LogicalId值。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::AppSync::ApiKey` 資源的 [ApiKeyId](#) 屬性。

Description

描述您的 API 金鑰。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `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

CloudFormation 相容性：此屬性會直接傳遞至 `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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::AppSync::ApiKey` 資源的 [ApiKeyId](#) 屬性。

Auth

設定 GraphQL API 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

```
Additional:
- AuthProvider
LambdaAuthorizer: LambdaAuthorizerConfig
OpenIDConnect: OpenIDConnectConfig
Type: String
UserPool: UserPoolConfig
```

Properties

Additional

API 的其他授權類型清單 GraphQL。

類型：[AuthProvider](#) 的清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有同等 CloudFormation 屬性。

LambdaAuthorizer

為您的 Lambda 事件函數授權方指定選用的授權組態。當 Type 指定為 時，您可以設定此選用屬性 AWS_LAMBDA。

類型：[LambdaAuthorizerConfig](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::AppSync::GraphQLApi 資源的 [LambdaAuthorizerConfig](#) 屬性。

OpenIDConnect

為您的 OpenID Connect 合規服務指定選用的授權組態。當 Type 指定為 時，您可以設定此選用屬性 OPENID_CONNECT。

類型：[OpenIDConnectConfig](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::AppSync::GraphQLApi 資源的 [OpenIDConnectConfig](#) 屬性。

Type

應用程式和 API AWS AppSync GraphQL 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[授權和身分驗證](#)。

當您指定 Lambda 授權方 (AWS_LAMBDA) 時，會 AWS SAM 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

CloudFormation 相容性：此屬性會直接傳遞至 `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 Type 指定為 時，您可以設定此選用屬性 `AMAZON_COGNITO_USER_POOLS`。

類型：[UserPoolConfig](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `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，我們從將 Lambda 授權方設定為 GraphQL API 的預設授權類型開始。

```
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，我們將以下內容新增至 AWS SAM 範本，為 GraphQL API 設定其他授權類型：

```
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 授權類型的選用授權組態。

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
```

Properties

LambdaAuthorizer

為您的 AWS Lambda 事件函數授權方指定選用的授權組態。當 Type 指定為 時，您可以設定此選用屬性AWS_LAMBDA。

類型：[LambdaAuthorizerConfig](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至AWS::AppSync::GraphQLApi [AdditionalAuthenticationProvider](#)物件的 [LambdaAuthorizerConfig](#) 屬性。

OpenIDConnect

為您的OpenID Connect合規服務指定選用的授權組態。當 Type 指定為 時，您可以設定此選用屬性OPENID_CONNECT。

類型：[OpenIDConnectConfig](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至AWS::AppSync::GraphQLApi [AdditionalAuthenticationProvider](#)物件的 [OpenIDConnectConfig](#) 屬性。

Type

應用程式和 API AWS AppSync GraphQL 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[授權和身分驗證](#)。

當您指定 Lambda 授權方 (AWS_LAMBDA) 時，會 AWS SAM 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

CloudFormation 相容性：此屬性會直接傳遞至AWS::AppSync::GraphQLApi [AdditionalAuthenticationProvider](#)物件的 [AuthenticationType](#) 屬性。

UserPool

指定使用 Amazon Cognito 使用者集區的選用授權組態。當 Type 指定為 時，您可以設定此選用屬性AMAZON_COGNITO_USER_POOLS。

類型：[UserPoolConfig](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至AWS::AppSync::GraphQLApi [AdditionalAuthenticationProvider](#)物件的 [UserPoolConfig](#) 屬性。

資料來源

設定 GraphQL API 解析程式可連線的資料來源。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

若要進一步了解資料來源，請參閱《AWS AppSync 開發人員指南》中的[連接資料來源](#)。

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](#)

Properties

DynamoDb

將 DynamoDB 資料表設定為 GraphQL API 解析程式的資料來源。

類型: [DynamoDb](#)

必要: 否

CloudFormation 相容性: 此屬性對 是唯一的 AWS SAM , 並且沒有 CloudFormation 同等的。

Lambda

將 Lambda 事件函數設定為 GraphQL API 解析程式的資料來源。

類型: [Lambda](#)

必要: 否

CloudFormation 相容性: 此屬性對 是唯一的 AWS SAM , 並且沒有 CloudFormation 同等的。

DynamoDb

將 Amazon DynamoDB 資料表設定為 GraphQL API 解析程式的資料來源。

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*

Properties

DeltaSync

描述差異同步組態。

類型：[DeltaSyncConfig](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::AppSync::DataSource` DynamoDBConfig 物件的 [DeltaSyncConfig](#) 屬性。

Description

資料來源的描述。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::AppSync::DataSource` 資源的 [Name](#) 屬性。

Name

資料來源的名稱。指定此屬性以覆寫LogicalId值。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::AppSync::DataSource` 資源的 [Name](#) 屬性。

Permissions

使用 佈建資料來源的許可[AWS SAM 連接器](#)。您可以在清單中提供下列任何值：

- Read – 允許解析程式讀取資料來源。
- Write – 允許解析程式寫入資料來源。

AWS SAM 使用在部署時轉換 `AWS::Serverless::Connector` 的資源來佈建您的許可。若要了解產生的資源，請參閱 [CloudFormation 當您指定 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。如果兩者皆未指定，AWS SAM 將產生預設值 `Read` 和 `Write`。若要撤銷對資料來源的存取，請從 AWS SAM 範本中移除 `DynamoDB` 物件。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等的。它類似於 `AWS::Serverless::Connector` 資源的 [Permissions](#) 屬性。

Region

DynamoDB 資料表 AWS 區域 的。如果您未指定，AWS SAM 請使用 [AWS::Region](#)。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::AppSync::DataSource` `DynamoDBConfig` 物件的 [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

必要：否。如果未指定，則 AWS SAM 套用的預設值 `Permissions`。

CloudFormation 相容性：此屬性會直接傳遞至 `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。如果您未指定 `ServiceRoleArn`，`TableArn` 則需要。

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等的。

TableName

資料表名稱。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::AppSync::DataSource` `DynamoDBConfig` 物件的 [TableName](#) 屬性。

UseCallerCredentials

設定為 `true` 以搭配此資料來源使用 IAM。

類型：布林值

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::AppSync::DataSource` `DynamoDBConfig` 物件的 [UseCallerCredentials](#) 屬性。

Versioned

設定為 `true` 以使用此資料來源來使用 [衝突偵測、衝突解決和同步](#)。

類型：布林值

必要：否

CloudFormation 相容性：此屬性會直接傳遞至AWS::AppSync::DataSource
DynamoDBConfig物件的 [Versioned](#) 屬性。

Lambda

將 AWS Lambda Event Function 設定為 GraphQL API 解析程式的資料來源。

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
```

Properties

Description

資料來源的描述。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::AppSync::DataSource 資源的
[Description](#) 屬性。

FunctionArn

Lambda 函數的 ARN。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至AWS::AppSync::DataSource
LambdaConfig物件的 [LambdaFunctionArn](#) 屬性。

LogicalId

資料來源的唯一名稱。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 AWS::AppSync::DataSource 資源的 [Name](#) 屬性。

Name

資料來源的名稱。指定此屬性以覆寫LogicalId值。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 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，請從 AWS SAM 範本中移除 Lambda 物件。

類型：字串

必要：否。如果未指定，AWS SAM 將使用 佈建Write許可[AWS SAM 連接器](#)。

CloudFormation 相容性：此屬性會直接傳遞至 AWS::AppSync::DataSource 資源的 [ServiceRoleArn](#) 屬性。

函式

在 GraphQL APIs 中設定函數以執行特定操作。

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
```

Properties

CodeUri

函數程式碼的 Amazon Simple Storage Service (Amazon S3) URI 或本機資料夾的路徑。

如果您指定本機資料夾的路徑，CloudFormation 需要先將檔案上傳至 Amazon S3，才能部署。您可以使用 AWS SAM CLI 來促進此程序。如需詳細資訊，請參閱[如何在部署時 AWS SAM 上傳本機檔案](#)。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::AppSync::FunctionConfiguration` 資源的 [CodeS3Location](#) 屬性。

DataSource

此函數將連接的資料來源名稱。

- 若要參考 `AWS::Serverless::GraphQLApi` 資源中的資料來源，請指定其邏輯 ID。

- 若要參考AWS::Serverless::GraphQLApi資源外部的資料來源，請使用 Fn::GetAtt 內部函數提供其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](#)。

如果指定的變化，[NONE | None | none] AWS SAM 將產生 AWS::AppSync::DataSource [Type](#) 物件None的值。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 AWS::AppSync::FunctionConfiguration 資源的 [DataSourceName](#) 屬性。

Description

函數的描述。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::AppSync::FunctionConfiguration 資源的 [Description](#) 屬性。

Id

位於AWS::Serverless::GraphQLApi資源外部之函數的函數 ID。

- 若要在相同 AWS SAM 範本中參考函數，請使用 Fn::GetAtt 內部函數。例如 Id: !GetAtt createPostItemFunc.FunctionId。
- 若要參考來自不同堆疊的函數，請使用 [Fn::ImportValue](#)。

使用時Id，不允許所有其他屬性。AWS SAM 會自動傳遞參考函數的函數 ID。

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

InlineCode

包含請求和回應函數的函數程式碼。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::AppSync::FunctionConfiguration` 資源的 [Code](#) 屬性。

LogicalId

函數的唯一名稱。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::AppSync::FunctionConfiguration` 資源的 [Name](#) 屬性。

MaxBatchSize

將傳送至BatchInvoke操作中單一 AWS Lambda 事件函數的解析程式請求輸入數目上限。

類型：整數

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::AppSync::FunctionConfiguration` 資源的 [MaxBatchSize](#) 屬性。

Name

函數的名稱。指定 以覆寫 LogicalId值。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::AppSync::FunctionConfiguration` 資源的 [Name](#) 屬性。

Runtime

描述 AWS AppSync 管道解析程式或 AWS AppSync 函數使用的執行時間。指定要使用的執行階段名稱和版本。

類型：[執行時間](#)

必要：是

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。它類似於 `AWS::AppSync::FunctionConfiguration` 資源的 [Runtime](#) 屬性。

Sync

描述函數的 Sync 組態。

指定叫用函數時要使用的衝突偵測策略和解決策略。

類型：[SyncConfig](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::AppSync::FunctionConfiguration` 資源的 [SyncConfig](#) 屬性。

執行時期

管道解析程式或函數的執行時間。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
```

Properties

Name

要使用的執行時間名稱。目前，唯一允許的值為 `APPSYNC_JS`。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至AWS::AppSync::FunctionConfiguration AppSyncRuntime物件的 [Name](#) 屬性。

Version

要使用的執行時間版本。目前唯一允許的版本為 1.0.0。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至AWS::AppSync::FunctionConfiguration AppSyncRuntime物件的 [RuntimeVersion](#) 屬性。

解析程式

針對 GraphQL API 的欄位設定解析程式。AWS Serverless Application Model (AWS SAM) 支援 [JavaScript 管道解析程式](#)。

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
```

Properties

Caching

已啟用快取之解析程式的快取組態。

類型：[CachingConfig](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::AppSync::Resolver` 資源的 [CachingConfig](#) 屬性。

CodeUri

解析程式函數程式碼的 Amazon Simple Storage Service (Amazon S3) URI 或本機資料夾的路徑。

如果您指定本機資料夾的路徑，CloudFormation 需要先將檔案上傳至 Amazon S3，才能部署。您可以使用 AWS SAM CLI 來促進此程序。如需詳細資訊，請參閱[如何在部署時 AWS SAM 上傳本機檔案](#)。

如果未提供 `CodeUri` 或 `InlineCode`，AWS SAM 將產生 `InlineCode`，將請求重新導向至第一個管道函數，並接收來自最後一個管道函數的回應。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::AppSync::Resolver` 資源的 [CodeS3Location](#) 屬性。

FieldName

解析程式的名稱。指定此屬性以覆寫 `LogicalId` 值。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::AppSync::Resolver` 資源的 [FieldName](#) 屬性。

InlineCode

包含請求和回應函數的解析程式程式碼。

如果未提供 `CodeUri` 或 `InlineCode`，AWS SAM 將產生 `InlineCode`，將請求重新導向至第一個管道函數，並接收來自最後一個管道函數的回應。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `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，您的解析程式名稱應該與其所使用的欄位名稱相符。針對 使用相同的欄位名稱LogicalId。

類型：字串

必要：是

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

MaxBatchSize

將傳送至BatchInvoke操作中單一 AWS Lambda 事件函數的解析程式請求輸入數目上限。

類型：整數

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::AppSync::Resolver` 資源的 [MaxBatchSize](#) 屬性。

OperationType

與您的解析程式相關聯的GraphQL操作類型。例如，Query、Mutation 或 Subscription。您可以在單一 LogicalId中透過 巢狀化多個解析程式OperationType。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::AppSync::Resolver` 資源的 [TypeName](#) 屬性。

Pipeline

與解析程式連結的函數。依清單中的邏輯 ID 指定函數。

類型：清單

必要：是

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。它類似於 AWS::AppSync::Resolver 資源的 [PipelineConfig](#) 屬性。

Runtime

管道解析程式或函數的執行時間。指定要使用的名稱和版本。

類型：[執行時間](#)

必要：是

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。它類似於 AWS::AppSync::Resolver 資源的 [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

CloudFormation 相容性：此屬性會直接傳遞至 AWS::AppSync::Resolver 資源的 [SyncConfig](#) 屬性。

範例

使用 AWS SAM 產生的解析程式函數程式碼，並將欄位儲存為變數

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
  }
```

以下是我們 AWS SAM 範本的程式碼片段：

```
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，我們不會指定 `CodeUri` 或 `InlineCode`。在部署時，AWS SAM 會自動為解析程式產生下列內嵌程式碼：

```
export function request(ctx) {
  return {};
}

export function response(ctx) {
```



```
    return ctx.prev.result;
  }
```

此預設解析程式碼會將請求重新導向至第一個管道函數，並接收來自最後一個管道函數的回應。

在第一個管道函數中，我們可以使用提供的args欄位來剖析請求物件並建立變數。然後，我們可以在函數中使用這些變數。以下是我們的 preprocessPostItem 函數範例：

```
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;
}
```

執行時期

管道解析程式或函數的執行時間。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
```

Properties

Name

要使用的執行時間名稱。目前，唯一允許的值為 APPSYNC_JS。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至AWS::AppSync::Resolver AppSyncRuntime物件的 [Name](#) 屬性。

Version

要使用的執行時間版本。目前唯一允許的版本為 1.0.0。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至AWS::AppSync::Resolver AppSyncRuntime物件的 [RuntimeVersion](#) 屬性。

AWS::Serverless::HttpApi

建立 Amazon API Gateway HTTP API，這可讓您建立延遲較低且成本低於 REST API 的 RESTful APIs。APIs 如需詳細資訊，請參閱[APIs開發人員指南](#)中的使用 HTTP API。

我們建議您使用 CloudFormation 勾點或 IAM 政策來驗證 API Gateway 資源是否已連接授權方，以控制對它們的存取。

如需使用 CloudFormation 勾點的詳細資訊，請參閱 CloudFormation CLI 使用者指南中的[註冊勾點](#)和 [apigw-enforce-authorizer](#) GitHub 儲存庫。

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，會將您的 AWS SAM 資源 AWS SAM 轉換為 CloudFormation 資源。如需詳細資訊，請參閱[為產生 CloudFormation 資源 AWS SAM](#)。

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
```

Properties

AccessLogSettings

階段中存取記錄的設定。

類型：[AccessLogSettings](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGatewayV2::Stage` 資源的 [AccessLogSettings](#) 屬性。

Auth

設定授權以控制對 API Gateway HTTP API 的存取。

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

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

CorsConfiguration

管理所有 API Gateway HTTP APIs 的跨來源資源共用 (CORS)。指定要允許做為字串的網域，或指定HttpApiCorsConfiguration物件。請注意，CORS 需要 AWS SAM 修改您的 OpenAPI 定義，因此 CORS 只有在指定 DefinitionBody 屬性時才有效。

如需詳細資訊，請參閱 [《API Gateway 開發人員指南》中的設定 HTTP API 的 CORS](#)。

Note

如果 CorsConfiguration 在 OpenAPI 定義和屬性層級同時設定，則 AWS SAM 會合併兩個組態來源，屬性優先。如果此屬性設定為 true，則允許所有原始伺服器。

類型：字串 | [HttpApiCorsConfiguration](#)

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

DefaultRouteSettings

此 HTTP API 的預設路由設定。除非由特定路由的 RouteSettings 屬性覆寫，否則這些設定適用於所有路由。

類型：[RouteSettings](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::ApiGatewayV2::Stage 資源的 [RouteSettings](#) 屬性。

DefinitionBody

描述 HTTP API 的 OpenAPI 定義。如果您未指定 DefinitionUri或 DefinitionBody，會根據範本組態DefinitionBody為您 AWS SAM 產生。

類型：JSON

必要：否

CloudFormation 相容性：此屬性類似於 `AWS::ApiGatewayV2::Api` 資源的 [Body](#) 屬性。如果提供特定屬性，AWS SAM 則可以在內容傳遞至 `DefinitionBody` 之前，將內容插入或修改 CloudFormation。對應 `AWS::Serverless::Function` 資源 `EventSource` 的屬性包括 `Auth` 和 `HttpApi` 類型的。

DefinitionUri

定義 HTTP API 之 OpenAPI 定義的 Amazon Simple Storage Service (Amazon S3) URI、本機檔案路徑或位置物件。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 `DefinitionBody` 已指定，會根據範本組態 `DefinitionBody` 為您 AWS SAM 產生。

如果您提供本機檔案路徑，範本必須經過包含 `sam deploy` 或 `sam package` 命令的工作流程，才能正確轉換定義。

您使用參考的外部 OpenAPI 定義檔案中不支援內部函數 `DefinitionUri`。若要將 OpenAPI 定義匯入範本，請使用 `DefinitionBody` 屬性搭配 [包含轉換](#)。

類型：字串 | [HttpApiDefinition](#)

必要：否

CloudFormation 相容性：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 資源的描述。

當您指定 `Description`，AWS SAM 將透過設定 `description` 欄位來修改 HTTP API 資源的 OpenAPI 定義。下列案例將導致錯誤：

- `DefinitionBody` 屬性是在開啟 API 定義中以 `description` 欄位集指定 – 這會導致 AWS SAM 無法解析 `description` 的欄位衝突。
- 屬性 `DefinitionUri` 已指定 – AWS SAM 不會修改從 Amazon S3 擷取的 Open API 定義。

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

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，您必須在 OpenAPI 定義 `x-amazon-apigateway-endpoint-configurationdisableExecuteApiEndpoint` 中指定 `DefinitionBody` 屬性，而不是 `DefinitionUri` 屬性或使用 定義。

類型：布林值

必要：否

CloudFormation 相容性：此屬性類似於 `AWS::ApiGatewayV2::Api` 資源的 [DisableExecuteApiEndpoint](#) 屬性。它會直接傳遞至 [x-amazon-apigateway-endpoint-configuration](#) 延伸的 `disableExecuteApiEndpoint` 屬性，該延伸會新增至 `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

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

FailOnWarnings

指定在遇到警告時是否要轉返 HTTP API 建立 (true) 與否 (false)。預設值為 false。

類型：布林值

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGatewayV2::Api` 資源的 [FailOnWarnings](#) 屬性。

Name

HTTP API 資源的名稱。

當您指定 `title`，AWS SAM 將透過設定 `title` 欄位來修改 HTTP API 資源的 OpenAPI 定義。下列案例將導致錯誤：

- `DefinitionBody` 屬性是在開啟 API 定義中以 `title` 欄位集指定 – 這會導致 AWS SAM 無法解析 `title` 的欄位衝突。
- 屬性 `DefinitionUri` 已指定 – AWS SAM 不會修改從 Amazon S3 擷取的 Open API 定義。

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

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。指定 `True` 在產生的資源中傳播標籤。

類型：布林值

必要：否

預設：False

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

RouteSettings

此 HTTP API 每個路由的路由設定。如需詳細資訊，請參閱 [APIs》中的使用 HTTP API 的路由](#)。

類型：[RouteSettings](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `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

預設：\$default

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGatewayV2::Stage` 資源的 [StageName](#) 屬性。

StageVariables

定義階段變數的映射。變數名稱可以包含英數字元和底線字元。這些值必須符合 `【A-Za-z0-9-._~:/?#&=,】+。`

類型：[Json](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGatewayV2::Stage` 資源的 [StageVariables](#) 屬性。

Tags

映射（字串到字串），指定要新增至此 API Gateway 階段的標籤。金鑰長度可以是 1 到 128 個 Unicode 字元，且不能包含字首 `aws:`。您可以使用以下任何字元：Unicode 字母、數字、空格、`_`、`.`、`/`、`=`、`+` 和 `-` 的組合。值長度可以是 1 到 256 個 Unicode 字元。

類型：映射

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

其他備註：Tags 屬性需要 AWS SAM 修改您的 OpenAPI 定義，因此只有在指定 `DefinitionBody` 屬性時才會新增標籤 - 如果指定 `DefinitionUri` 屬性，則不會新增標籤。AWS SAM 自動新增 `httpapi:createdBy:SAM` 標籤。標籤也會新增至 `AWS::ApiGatewayV2::Stage` 資源和資源 `AWS::ApiGatewayV2::DomainName` (如果 `DomainName` 已指定)。

傳回值

Ref

當您將此資源的邏輯 ID 傳遞至內部 Ref 函數時，便會 Ref 傳回基礎 `AWS::ApiGatewayV2::Api` 資源的 API ID，例如 `a1bcdef2gh`。

如需使用 Ref 函數的詳細資訊，請參閱 AWS CloudFormation 《使用者指南[Ref](#)》中的。

範例

簡易 HttpApi

下列範例顯示設定由 Lambda 事件函數支援的 HTTP API 端點所需的最低需求。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
```

具有 Auth 的 HttpApi

下列範例示範如何在 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"
```

具有 OpenAPI 定義的 HttpApi

下列範例示範如何將 OpenAPI 定義新增至範本。

請注意，會為參考此 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 事件參考的任何遺失路徑。

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
```

具有組態設定的 HttpApi

下列範例示範如何將 HTTP API 和階段組態新增至範本。

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 APIs 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

```

Authorizers: OAuth2Authorizer | LambdaAuthorizer
DefaultAuthorizer: String
EnableIamAuthorizer: Boolean

```

Properties

Authorizers

用於控制 API Gateway API 存取的授權方。

類型：[OAuth2Authorizer](#) | [LambdaAuthorizer](#)

必要：否

預設：無

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

其他備註：將授權方 AWS SAM 新增至 OpenAPI 定義。

DefaultAuthorizer

指定用於授權 API Gateway API 呼叫的預設授權方。如果 `EnableIamAuthorizer` 設定為 `true`，您可以將 `AWS_IAM` 指定為預設授權方。否則，請指定您在 `Authorizers` 中定義的授權方。

類型：字串

必要：否

預設：無

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

EnableIamAuthorizer

指定是否使用 API 路由的 IAM 授權。

類型：布林值

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

範例

OAuth 2.0 授權方

OAuth 2.0 授權方範例

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 授權方

IAM 授權方範例

YAML

```
Auth:
  EnableIamAuthorizer: true
  DefaultAuthorizer: AWS_IAM
```

LambdaAuthorizer

設定 Lambda 授權方，以使用 AWS Lambda 事件函數控制對 Amazon API Gateway HTTP API 的存取。

如需詳細資訊和範例，請參閱[APIs》中的使用 HTTP API 的授權 AWS Lambda 方](#)。

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
```

Properties

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。

這會在 OpenAPI 定義的 `authorizerPayloadFormatVersion` 區段 `x-amazon-apigateway-authorizer` 中傳遞至的 `securitySchemes` 區段。

有效值：1.0 或 2.0

類型：字串

必要：是

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

`EnableFunctionDefaultPermissions`

根據預設，不會授予 HTTP API 資源调用 Lambda 授權方的許可。將此屬性指定為 `true`，以在 HTTP API 資源和 Lambda 授權方之間自動建立許可。

類型：布林值

必要：否

預設值：`false`

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有同等 CloudFormation 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 授權方必須傳回 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。

這會在 OpenAPI 定義的 `enableSimpleResponses` 區段 `x-amazon-apigateway-authorizer` 中傳遞至的 `securitySchemes` 區段。

類型：布林值

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

`FunctionArn`

提供 API 授權的 Lambda 函數的 Amazon Resource Name (ARN)。

這會在 OpenAPI 定義的 `authorizerUri` 區段 `x-amazon-apigateway-authorizer` 中傳遞至的 `securitySchemes` 區段。

類型：字串

必要：是

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

FunctionInvokeRole

IAM 角色的 ARN，具有 API Gateway 叫用授權方函數所需的登入資料。如果函數的資源型政策未授予 API Gateway `lambda:InvokeFunction` 許可，請指定此參數。

這會在 OpenAPI 定義的 `authorizerCredentials` 區段 `x-amazon-apigateway-authorizer` 中傳遞至的 `securitySchemes` 區段。

如需詳細資訊，請參閱《API Gateway 開發人員指南》中的 [建立 Lambda 授權方](#)。

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

Identity

在 授權方的傳入請求 `IdentitySource` 中指定。

這會在 OpenAPI 定義的 `identitySource` 區段 `x-amazon-apigateway-authorizer` 中傳遞至的 `securitySchemes` 區段。

類型：[LambdaAuthorizationIdentity](#)

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

範例

LambdaAuthorizer

LambdaAuthorizer 範例

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` 可用於在 Lambda 授權方的傳入請求中指定 `IdentitySource`。如需身分來源的詳細資訊，請參閱 API Gateway 開發人員指南中的 [身分來源](#)。

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
QueryString: List
ReauthorizeEvery: Integer
StageVariables: List

```

Properties

Context

將指定的內容字串轉換為格式為 `的映射表達式清單` `$context.contextString`。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

Headers

將標頭轉換為格式為 的映射表達式清單`$request.header.name`。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

QueryString

將指定的查詢字串轉換為格式為 的映射表達式清單`$request.querystring.queryString`。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

ReauthorizeEvery

存活期 (TTL) 期間 (秒)，指定 API Gateway 快取授權方結果的時間。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

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

StageVariables

將指定的階段變數轉換為格式為 的映射表達式清單`$stageVariables.stageVariable`。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

範例

LambdaRequestIdentity

Lambda 請求身分範例

YAML

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

OAuth2Authorizer

OAuth 2.0 授權方的定義，也稱為 JSON Web Token (JWT) 授權方。

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
```

Properties

AuthorizationScopes

此授權方的授權範圍清單。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

IdentitySource

此授權方的身分來源表達式。

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

JwtConfiguration

此授權方的 JWT 組態。

這會在 OpenAPI 定義的 `jwtConfiguration` 區段 `x-amazon-apigateway-authorizer` 中傳遞至的 `securitySchemes` 區段。

Note

屬性 `issuer` 和 `audience` 大小寫不區分，可用於小寫，如 OpenAPI 或大寫 `IssuerAudience` 和 [AWS::ApiGatewayV2::Authorizer](#)。

類型：映射

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

範例

OAuth 2.0 授權方

OAuth 2.0 授權方範例

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 APIs 的跨來源資源共用 (CORS)。指定要允許做為字串的網域，或指定具有其他 Cors 組態的字典。注意：Cors 需要 SAM 修改您的 OpenAPI 定義，因此它僅適用於 DefinitionBody 屬性中定義的內嵌 OpenApi。

如需 CORS 的詳細資訊，請參閱 API Gateway 開發人員指南中的[設定 HTTP API 的 CORS](#)。

注意：如果在 OpenAPI 和屬性層級同時設定 HttpApiCorsConfiguration，會將它們與優先的屬性 AWS SAM 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
```

Properties

AllowCredentials

指定 CORS 請求中是否包含憑證。

類型：布林值

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

AllowHeaders

代表允許標頭的集合。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

AllowMethods

代表允許的 HTTP 方法集合。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

AllowOrigins

代表允許的來源集合。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

ExposeHeaders

代表公開標頭的集合。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

MaxAge

瀏覽器應快取預檢請求結果的秒數。

類型：整數

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

範例

HttpApiCorsConfiguration

HTTP API Cors 組態範例。

YAML

```
CorsConfiguration:
  AllowOrigins:
    - "https://example.com"
  AllowHeaders:
    - x-apigateway-header
  AllowMethods:
    - GET
  MaxAge: 600
  AllowCredentials: true
```

HttpApiDefinition

定義 API 的 OpenAPI 文件。

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
Key: String
Version: String
```

Properties

Bucket

存放 OpenAPI 檔案的 Amazon S3 儲存貯體名稱。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGatewayV2::Api BodyS3Location` 資料類型的 [Bucket](#) 屬性。

Key

OpenAPI 檔案的 Amazon S3 金鑰。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGatewayV2::Api BodyS3Location` 資料類型的 [Key](#) 屬性。

Version

對於版本控制的物件，則為 OpenAPI 檔案的版本。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGatewayV2::Api BodyS3Location` 資料類型的 [Version](#) 屬性。

範例

定義 Uri 範例

API 定義範例

YAML

```
DefinitionUri:
  Bucket: sam-s3-demo-bucket-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
```

Properties

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：/

CloudFormation 相容性：此屬性類似於 `AWS::ApiGatewayV2::ApiMapping` 資源的 [ApiMappingKey](#) 屬性。會 AWS SAM 建立多個 `AWS::ApiGatewayV2::ApiMapping` 資源，每個在此屬性中指定的值各一個。

CertificateArn

此網域名稱端點受 AWS 管憑證的 Amazon Resource Name (ARN)。AWS Certificate Manager 是唯一支援的來源。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `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 會在設定此屬性時產生 `AWS::ApiGatewayV2::DomainName` 資源。如需此案例的資訊，請參閱 [已指定 DomainName 屬性](#)。如需所產生 CloudFormation 資源的資訊，請參閱 [為產生 CloudFormation 資源 AWS SAM](#)。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGatewayV2::DomainName` 資源的 [DomainName](#) 屬性。

EndpointConfiguration

定義要映射到自訂網域的 API Gateway 端點類型。此屬性的值會決定 `CertificateArn` 屬性的映射方式 CloudFormation。

HTTP APIs 的唯一有效值是 REGIONAL。

類型：字串

必要：否

預設：REGIONAL

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

MutualTlsAuthentication

自訂網域名稱的交互傳輸層安全 (TLS) 身分驗證組態。

類型：[MutualTlsAuthentication](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `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。只有在您設定交互 TLS 並指定的 ACM 匯入或私有 CA 憑證 ARN 時才需要 `CertificateArn`。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGatewayV2::DomainName` `DomainNameConfiguration` 資料類型的 [OwnershipVerificationCertificateArn](#) 屬性。

Route53

定義 Amazon Route 53 組態。

類型：[Route53Configuration](#)

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

SecurityPolicy

此網域名稱之安全政策的 TLS 版本。

HTTP APIs 的唯一有效值是 TLS_1_2。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGatewayV2::DomainName` `DomainNameConfiguration` 資料類型的 [SecurityPolicy](#) 屬性。

範例

DomainName

DomainName 範例

YAML

```
Domain:
  DomainName: www.example.com
  CertificateArn: arn-example
  EndpointConfiguration: REGIONAL
  Route53:
```

```
HostedZoneId: Z1PA6795UKMFR9
BasePath:
- foo
- bar
```

Route53Configuration

設定 API 的 Route53 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
```

Properties

DistributionDomainName

設定 API 自訂網域名稱的自訂分佈。

類型：字串

必要：否

預設：使用 API Gateway 分佈。

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Route53::RecordSetGroup` `AliasTarget` 資源的 [DNSName](#) 屬性。

其他備註：[CloudFront 分佈](#)的網域名稱。

EvaluateTargetHealth

當 `EvaluateTargetHealth` 為 `true` 時，別名記錄會繼承參考 AWS 資源的運作狀態，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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Route53::RecordSetGroup` `AliasTarget` 資源的 [EvaluateTargetHealth](#) 屬性。

其他備註：當別名目標為 CloudFront 分佈時，您無法將 `EvaluateTargetHealth` 設定為 `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

CloudFormation 相容性：此屬性會直接傳遞至 `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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Route53::RecordSetGroup` `RecordSet` 資源的 [HostedZoneName](#) 屬性。

IPv6

設定此屬性時，會 AWS SAM 建立 `AWS::Route53::RecordSet` 資源，並將提供的 `HostedZone` AAAA 的 [Type](#) 設定為。

類型：布林值

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

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。資源通常是 AWS 資源，例如 EC2 執行個體或 ELB 負載平衡器，並且根據記錄類型由 IP 地址或 DNS 網域名稱參考。

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

CloudFormation 相容性：此屬性會直接傳遞至

AWS::Route53::RecordSetGroupRecordSet資料類型的 [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

CloudFormation 相容性：此屬性會直接傳遞至

AWS::Route53::RecordSetGroupRecordSet資料類型的 [SetIdentifier](#) 屬性。

範例

Route 53 組態範例

此範例說明如何設定 Route 53。

YAML

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

AWS::Serverless::WebSocketApi

建立 Amazon API Gateway WebSocket API，可讓您建立雙向互動式通訊應用程式。WebSocket APIs 可讓伺服器傳送訊息給用戶端，而不需要用戶端加以請求。如需詳細資訊，請參閱[APIs》中的使用 WebSocket API](#)。

我們建議您使用 CloudFormation 勾點或 IAM 政策來驗證 API Gateway 資源是否已連接授權方，以控制對它們的存取。

如需使用 CloudFormation 勾點的詳細資訊，請參閱 CloudFormation CLI 使用者指南中的[註冊勾點](#)和 [apigw-enforce-authorizer](#) GitHub 儲存庫。

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，會將您的 AWS SAM 資源 AWS SAM 轉換為 CloudFormation 資源。如需詳細資訊，請參閱[為產生 CloudFormation 資源 AWS SAM](#)。

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::WebSocketApi
Properties:
  ApiKeySelectionExpression: String
  AccessLogSettings: AccessLogSettings
  Auth: WebSocketApiAuth
  DefaultRouteSettings: RouteSettings
  Description: String
  DisableExecuteApiEndpoint: Boolean
  DisableSchemaValidation: Boolean
  Domain: WebSocketApiDomainConfiguration
  IpAddressType: String
  Name: String
  PropagateTags: Boolean
  Routes: RouteConfiguration
  RouteSelectionExpression: String
  RouteSettings: RouteSettings
  StageName: String
  StageVariables: Json
  Tags: Map
```

Properties

ApiKeySelectionExpression

API 金鑰選擇表達式。如需詳細資訊，請參閱《[API Gateway 開發人員指南](#)》中的 [API 金鑰選擇表達式](#)。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::ApiGatewayV2::Api 資源的 [ApiKeySelectionExpression](#) 屬性。

AccessLogSettings

階段中存取記錄的設定。

類型：[AccessLogSettings](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGatewayV2::Stage` 資源的 [AccessLogSettings](#) 屬性。

Auth

設定控制 WebSocket API 存取的授權。授權會套用至 `$connect` 路由。

如需詳細資訊，請參閱 [《API Gateway 開發人員指南》](#) 中的控制對 WebSocket API 的存取 APIs。

類型：[WebSocketApiAuth](#)

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

DefaultRouteSettings

此 WebSocket API 的預設路由設定。除非由特定路由的 `RouteSettings` 屬性覆寫，否則這些設定適用於所有路由。

類型：[RouteSettings](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGatewayV2::Stage` 資源的 [DefaultRouteSettings](#) 屬性。

Description

WebSocket API 的說明。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGatewayV2::Api` 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。如要要求用戶端使用自訂網域名稱來叫用 API，請停用預設端點。

類型：布林值

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGatewayV2::Api` 資源的 [DisableExecuteApiEndpoint](#) 屬性。

DisableSchemaValidation

避免在建立部署時驗證模型。

類型：布林值

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGatewayV2::Api` 資源的 [DisableSchemaValidation](#) 屬性。

Domain

設定此 WebSocket API 的自訂網域。

Note

WebSocket APIs 不支援交互 TLS 身分驗證 (MTLS)。如果您指定 `MutualTlsAuthentication` 或 `OwnershipVerificationCertificateArn`，AWS SAM 將傳回錯誤。

類型：[WebSocketApiDomainConfiguration](#)

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

IpAddressType

API 的 IP 地址類型。有效值僅適用於 `ipv4` IPv4，`dualstack` 適用於 IPv4 和 IPv6。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGatewayV2::Api` 資源的 [IpAddressType](#) 屬性。

Name

WebSocket API 的名稱。如果您未指定名稱，會為您 AWS SAM 產生名稱。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGatewayV2::Api` 資源的 [Name](#) 屬性。

PropagateTags

如果為 `true`，則會將 `Tags` 屬性 AWS SAM 新增至 AWS SAM 產生的 `AWS::ApiGatewayV2::DomainName` `AWS::ApiGatewayV2::Stage` 和資源。

類型：布林值

必要：否

CloudFormation 相容性：此屬性對是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

Routes

此 WebSocket API 的路由組態。路由定義訊息如何路由到 Lambda 事件函數。每個路由都包含路由金鑰和 Lambda 函數 ARN。

WebSocket APIs 支援三個預先定義的路由：`$connect`、`$disconnect` 和 `$default`。您也可以定義自訂路由。

類型：[RouteConfiguration](#)

必要：是

CloudFormation 相容性：此屬性對是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

RouteSelectionExpression

WebSocket API 的路由選擇表達式。如需詳細資訊，請參閱《API Gateway 開發人員指南》中的[路由選擇表達式](#)。

常見值為 `$request.body.action`，它會根據訊息內文中的 `action` 欄位來路由訊息。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGatewayV2::Api` 資源的 [RouteSelectionExpression](#) 屬性。

RouteSettings

此 WebSocket API 的路由設定。這些設定會覆寫特定路由 `DefaultRouteSettings` 的。

類型：[RouteSettings](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGatewayV2::Stage` 資源的 [RouteSettings](#) 屬性。

StageName

API 階段的名稱。如果您未指定名稱，AWS SAM 會使用 `default` 做為階段名稱。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGatewayV2::Stage` 資源的 [StageName](#) 屬性。

StageVariables

定義階段變數的映射。變數名稱可以包含英數字元和底線字元，且值必須符合 `[A-Za-z0-9-._~:/?#&=, ]+`。

類型：[Json](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGatewayV2::Stage` 資源的 [StageVariables](#) 屬性。

Tags

映射（字串到字串），指定要新增至此 WebSocket API 的標籤。如需標籤有效金鑰和值的詳細資訊，請參閱 CloudFormation 《使用者指南》中的 [資源標籤](#)。

類型：映射

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

傳回值

Ref

當您將此資源的邏輯 ID 傳遞至內部 Ref 函數時， 便會Ref傳回基礎AWS::ApiGatewayV2::Api資源的 API ID，例如 a1bcdef2gh。

如需使用 Ref 函數的詳細資訊，請參閱AWS CloudFormation 《使用者指南[Ref](#)》中的 。

範例

簡易 WebSocket API

下列範例會建立具有三個路由的 WebSocket API。

```
Resources:
  MyWebSocketApi:
    Type: AWS::Serverless::WebSocketApi
    Properties:
      RouteSelectionExpression: $request.body.action
      Routes:
        $connect:
          FunctionArn: !GetAtt ConnectFunction.Arn
        $disconnect:
          FunctionArn: !GetAtt DisconnectFunction.Arn
        sendMessage:
          FunctionArn: !GetAtt SendMessageFunction.Arn

  ConnectFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.connect
      Runtime: nodejs20.x
      CodeUri: ./src

  DisconnectFunction:
    Type: AWS::Serverless::Function
    Properties:
```

```
Handler: index.disconnect
Runtime: nodejs20.x
CodeUri: ./src
```

SendMessageFunction:

```
Type: AWS::Serverless::Function
Properties:
  Handler: index.sendMessage
  Runtime: nodejs20.x
  CodeUri: ./src
```

具有 Lambda 授權方的 WebSocket API

下列範例會使用 Lambda 授權方建立 WebSocket API。

Resources:**MyWebSocketApi:**

```
Type: AWS::Serverless::WebSocketApi
Properties:
  RouteSelectionExpression: $request.body.action
  Auth:
    AuthType: CUSTOM
    AuthArn: !GetAtt AuthorizerFunction.Arn
    IdentitySource:
      - route.request.header.Authorization
  Routes:
    $connect:
      FunctionArn: !GetAtt ConnectFunction.Arn
    sendMessage:
      FunctionArn: !GetAtt SendMessageFunction.Arn
```

AuthorizerFunction:

```
Type: AWS::Serverless::Function
Properties:
  Handler: index.authorize
  Runtime: nodejs20.x
  CodeUri: ./src
```

ConnectFunction:

```
Type: AWS::Serverless::Function
Properties:
  Handler: index.connect
  Runtime: nodejs20.x
  CodeUri: ./src
```

```
SendMessageFunction:
  Type: AWS::Serverless::Function
  Properties:
    Handler: index.sendMessage
    Runtime: nodejs20.x
    CodeUri: ./src
```

WebSocketApiAuth

設定授權以控制對 Amazon API Gateway WebSocket API 的存取。

如需設定 WebSocket APIs存取權的詳細資訊，請參閱《API Gateway 開發人員指南》中的[控制 WebSocket APIs存取權](#)。

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

```
AuthArn: String
AuthType: String
IdentitySource: List
InvokeRole: String
Name: String
```

Properties

AuthArn

用於授權的 Lambda 事件函數 ARN。當 AuthType 為 時為必要CUSTOM。

類型：字串

必要：有條件

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

AuthType

授權的類型。有效值為 NONE、AWS_IAM 或 CUSTOM。

- NONE - 無授權
- AWS_IAM - IAM 授權
- CUSTOM - Lambda 授權方

類型：字串

必要：是

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

IdentitySource

請求授權的身分來源。例如 `route.request.header.Authorization`。

類型：清單

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGatewayV2::Authorizer` 資源的 [IdentitySource](#) 屬性。

InvokeRole

API Gateway 在叫用授權方函數時擔任的 IAM 角色 ARN。如果未指定，AWS SAM 會自動建立資源型許可，允許 API Gateway 叫用授權方函數。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGatewayV2::Authorizer` 資源的 [AuthorizerCredentialsArn](#) 屬性。

Name

授權方的名稱。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGatewayV2::Authorizer` 資源的 [Name](#) 屬性。

範例

Lambda 授權方

下列範例會設定 WebSocket API 的 Lambda 授權方。

```
Auth:
  AuthType: CUSTOM
  AuthArn: !GetAtt AuthorizerFunction.Arn
  IdentitySource:
    - route.request.header.Authorization
```

IAM 授權

下列範例會設定 WebSocket API 的 IAM 授權。

```
Auth:
  AuthType: AWS_IAM
```

WebSocketApiDomainConfiguration

設定 WebSocket 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
Route53: Route53Configuration
SecurityPolicy: String
```

Properties

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：/

CloudFormation 相容性：此屬性類似於 `AWS::ApiGatewayV2::ApiMapping` 資源的 [ApiMappingKey](#) 屬性。會 AWS SAM 建立多個 `AWS::ApiGatewayV2::ApiMapping` 資源，每個在此屬性中指定的值各一個。

CertificateArn

此網域名稱端點受 AWS 管憑證的 Amazon Resource Name (ARN)。AWS Certificate Manager 是唯一支援的來源。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGatewayV2::DomainName` `DomainNameConfiguration` 資源的 [CertificateArn](#) 屬性。

DomainName

API Gateway WebSocket API 的自訂網域名稱。不支援大寫字母。

AWS SAM 會在設定此屬性時產生 `AWS::ApiGatewayV2::DomainName` 資源。如需所產生 CloudFormation 資源的資訊，請參閱 [為產生 CloudFormation 資源 AWS SAM](#)。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGatewayV2::DomainName` 資源的 [DomainName](#) 屬性。

EndpointConfiguration

定義要映射到自訂網域的 API Gateway 端點類型。此屬性的值會決定 `CertificateArn` 屬性的映射方式 CloudFormation。

WebSocket APIs 的唯一有效值是 REGIONAL。

類型：字串

必要：否

預設：REGIONAL

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

Route53

定義 Route 53 組態。

類型：[Route53Configuration](#)

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

SecurityPolicy

此網域名稱之安全政策的 TLS 版本。

WebSocket APIs 的唯一有效值是 TLS_1_2。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGatewayV2::DomainNameDomainNameConfiguration` 資料類型的 [SecurityPolicy](#) 屬性。

範例

DomainName

DomainName 範例

YAML

```
Domain:
  DomainName: ws.example.com
  CertificateArn: arn:aws:acm:us-east-1:123456789012:certificate/example
  EndpointConfiguration: REGIONAL
  Route53:
    HostedZoneId: Z1PA6795UKMFR9
  BasePath:
    - v1
```

- v2

Route53Configuration

設定 API 的 Route53 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

```
EvaluateTargetHealth: Boolean  
HostedZoneId: String  
HostedZoneName: String  
IPv6: Boolean  
Region: String  
SetIdentifier: String
```

Properties

EvaluateTargetHealth

當 EvaluateTargetHealth 為 true 時，別名記錄會繼承參考 AWS 資源的運作狀態，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

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Route53::RecordSetGroup AliasTarget 資源的 [EvaluateTargetHealth](#) 屬性。

其他備註：當別名目標為 CloudFront 分佈時，您無法將 EvaluateTargetHealth 設定為 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

CloudFormation 相容性：此屬性會直接傳遞至 `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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Route53::RecordSetGroup` `RecordSet` 資源的 [HostedZoneName](#) 屬性。

IPv6

設定此屬性時，會 AWS SAM 建立 `AWS::Route53::RecordSet` 資源，並將提供的 `HostedZone` `AAAA` 的 [Type](#) 設定為。

類型：布林值

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

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。資源通常是 AWS 資源，例如 EC2 執行個體或 ELB 負載平衡器，並且根據記錄類型由 IP 地址或 DNS 網域名稱參考。

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

CloudFormation 相容性：此屬性會直接傳遞至

AWS::Route53::RecordSetGroupRecordSet 資料類型的 [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

CloudFormation 相容性：此屬性會直接傳遞至

AWS::Route53::RecordSetGroupRecordSet 資料類型的 [SetIdentifier](#) 屬性。

範例

Route 53 組態範例

此範例說明如何設定 Route 53。

YAML

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

RouteConfiguration

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  
FunctionArn: String  
IntegrationTimeout: Integer  
ModelSelectionExpression: String  
OperationName: String  
RequestModels: Json  
RequestParameters: Json  
RouteResponseSelectionExpression: String
```

Properties

ApiKeyRequired

指定此路由是否需要 API 金鑰。

類型：布林值

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGatewayV2::Route` 資源的 [ApiKeyRequired](#) 屬性。

FunctionArn

處理此路由的 Lambda 事件函數 ARN。會 AWS SAM 建立整合和 API Gateway 叫用函數的必要許可。

類型：字串

必要：是

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

IntegrationTimeout

整合的逾時，以毫秒為單位。最大值為 29,000 毫秒 (29 秒)。

類型：整數

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGatewayV2::Integration` 資源的 [TimeoutInMillis](#) 屬性。

ModelSelectionExpression

路由的模型選擇表達式。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGatewayV2::Route` 資源的 [ModelSelectionExpression](#) 屬性。

OperationName

路由的操作名稱。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGatewayV2::Route` 資源的 [OperationName](#) 屬性。

RequestModels

路由的請求模型。

類型：Json

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGatewayV2::Route` 資源的 [RequestModels](#) 屬性。

RequestParameters

路由的請求參數。

類型：Json

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGatewayV2::Route` 資源的 [RequestParameters](#) 屬性。

RouteResponseSelectionExpression

路由的路由回應選擇表達式。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::ApiGatewayV2::Route` 資源的 [RouteResponseSelectionExpression](#) 屬性。

範例

簡易路由

下列範例會設定簡單的路由。

```
Routes:
  $connect:
    FunctionArn: !GetAtt ConnectFunction.Arn
  sendMessage:
    FunctionArn: !GetAtt SendMessageFunction.Arn
    OperationName: SendMessage
```

AWS::Serverless::LayerVersion

建立 Lambda LayerVersion，其中包含 Lambda 事件函數所需的程式庫或執行時間程式碼。

[AWS::Serverless::LayerVersion](#) 資源也支援 Metadata 資源屬性，因此您可以指示 AWS SAM 建置應用程式中包含的層。如需建置層的詳細資訊，請參閱 [在中建置 Lambda 層 AWS SAM](#)。

重要注意事項：由於 [UpdateReplacePolicy](#) 資源屬性在 中發行 CloudFormation，[AWS::Lambda::LayerVersion](#)（建議）提供與 相同的優點[AWS::Serverless::LayerVersion](#)。

轉換 Serverless LayerVersion 時，SAM 也會轉換資源的邏輯 ID，以便在更新資源時 CloudFormation 不會自動刪除舊的 LayerVersions。

Note

當您部署到 時 AWS CloudFormation，會將您的 AWS SAM 資源 AWS SAM 轉換為 CloudFormation 資源。如需詳細資訊，請參閱 [為 產生 CloudFormation 資源 AWS SAM](#)。

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
  PublishLambdaVersion: Boolean
  RetentionPolicy: String
```

Properties

CompatibleArchitectures

指定 layer 版本的支援指令集架構。

如需此屬性的詳細資訊，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：清單

必要：否

預設：x86_64

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Lambda::LayerVersion 資源的 [CompatibleArchitectures](#) 屬性。

CompatibleRuntimes

與此 LayerVersion 相容的執行時間清單。

類型：清單

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::LayerVersion` 資源的 [CompatibleRuntimes](#) 屬性。

ContentUri

Amazon S3 Uri、本機資料夾的路徑，或 layer 程式碼的 LayerContent 物件。

如果提供 Amazon S3 Uri 或 LayerContent 物件，則參考的 Amazon S3 物件必須是包含 [Lambda](#) 層內容的有效 ZIP 封存。

如果提供本機資料夾的路徑，則要正確轉換內容，範本必須經過包含的工作流程，[sam build](#)後面接著 [sam deploy](#)或 [sam package](#)。根據預設，相對路徑會根據 AWS SAM 範本的位置進行解析。

類型：字串 | [LayerContent](#)

必要：是

CloudFormation 相容性：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

此層的描述。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `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：資源邏輯 ID

CloudFormation 相容性：此屬性類似於 `AWS::Lambda::LayerVersion` 資源的 [LayerName](#) 屬性。如果您未指定名稱，資源的邏輯 ID 將用作名稱。

LicenseInfo

此 `LayerVersion` 的授權相關資訊。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::LayerVersion` 資源的 [LicenseInfo](#) 屬性。

PublishLambdaVersion

選擇加入屬性，可在參考 `LayerVersion` 資源發生變更時建立新的 Lambda 版本。在連線的 Lambda 函數 `AutoPublishAliasAllProperties` 中使用 `AutoPublishAlias` 和 啟用時，將會針對對 `LayerVersion` 資源所做的每個變更建立新的 Lambda 版本。

類型：布林值

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

RetentionPolicy

此屬性會指定刪除資源時 `LayerVersion`，是否保留或刪除舊版本的。如果您在更新或取代資源 `LayerVersion` 時需要保留舊版，您必須啟用 `UpdateReplacePolicy` 屬性。如需執行此操作的資訊，請參閱 AWS CloudFormation 《使用者指南》中的 [UpdateReplacePolicy 屬性](#)。

有效值：Retain 或 Delete

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

其他備註：當您指定 `Retain`，會將 [支援的資源屬性 AWS SAM](#) 的 `DeletionPolicy`：`Retain` AWS SAM 新增至轉換 `AWS::Lambda::LayerVersion` 的資源。

傳回值

Ref

當將此資源的邏輯 ID 提供給 `Ref` 內部 函數時，它會傳回基礎 `Lambda LayerVersion` 的資源 ARN。

如需使用 `Ref` 函數的詳細資訊，請參閱 AWS CloudFormation 《使用者指南[Ref](#)》中的 。

範例

LayerVersionExample

LayerVersion 的範例

YAML

```
Properties:
  LayerName: MyLayer
  Description: Layer description
  ContentUri: 's3://sam-s3-demo-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
StorageMode: String
```

Version: *String*

Properties

Bucket

層封存的 Amazon S3 儲存貯體。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::LayerVersion Content` 資料類型的 [S3Bucket](#) 屬性。

Key

層封存的 Amazon S3 金鑰。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::LayerVersion Content` 資料類型的 [S3Key](#) 屬性。

StorageMode

控制 Lambda 如何存放 layer 封存。

- COPY – Lambda 會將層封存的副本上傳至 Lambda 受管儲存。您可以在 Lambda 建立 layer 版本後刪除 Amazon S3 物件。
- REFERENCE – Lambda 參考來自 Amazon S3 儲存貯體的層封存，不會儲存複本。物件必須保留在原處，Lambda 必須在 layer 版本的生命週期內保留其存取權。

若要使用 REFERENCE，請在 Amazon S3 儲存貯體上啟用版本控制，並授予 Lambda 服務主體存取物件的權限。如需詳細資訊，請參閱 [《開發人員指南》中的自我管理 Amazon S3 程式碼儲存](#)。

AWS Lambda

類型：字串

有效值：COPY | REFERENCE

必要：否

預設：COPY

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::LayerVersion Content` 資料類型的 [S3ObjectStorageMode](#) 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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::LayerVersion Content` 資料類型的 [S3ObjectVersion](#) 屬性。

範例

LayerContent

Layer 內容範例

YAML

```
LayerContent:
  Bucket: amzn-s3-demo-bucket-name
  Key: mykey-name
  Version: 121212
```

自我管理的 Amazon S3 程式碼儲存

以下 `LayerContent` 參考來自您自己的 Amazon S3 儲存貯體的層封存，而不是將其複製到 Lambda 受管儲存。

YAML

```
LayerContent:
  Bucket: amzn-s3-demo-bucket-name
  Key: mykey-name
  Version: 121212
  StorageMode: REFERENCE
```

AWS::Serverless::MicrovmImage

從存放在 Amazon S3 中的程式碼成品建立 AWS Lambda MicroVM 映像。MicroVM 映像是從 Dockerfile 和應用程式程式碼建置的 Firecracker 快照，可為 MicroVM 執行個體啟用一秒的啟動。

Note

當您部署到時 AWS CloudFormation，會將您的 AWS SAM 資源 AWS SAM CloudFormation 轉換為資源。如需詳細資訊，請參閱 [為產生 CloudFormation 資源 AWS SAM](#)。

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::MicrovmImage
Properties:
  AdditionalOsCapabilities: List
  BaseImageArn: String
  BaseImageVersion: String
  BuildRoleArn: String
  CodeUri: String
  CpuConfigurations: List
  Description: String
  EgressNetworkConnectors: List
  EnvironmentVariables: Map
  Hooks: Hooks
  Logging: Logging
  Name: String
  PropagateTags: Boolean
  Resources: List
  Tags: Map
```

Properties

AdditionalOsCapabilities

要授予 MicroVM 執行期環境的其他作業系統功能。如果您未指定值，服務不會授予其他作業系統功能。

有效值：ALL

類型：清單

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::MicrovmImage` 資源的 [AdditionalOsCapabilities](#) 屬性。

BaseImageArn

要從中建置之基礎 MicroVM 映像的 ARN。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::MicrovmImage` 資源的 [BaseImageArn](#) 屬性。

BaseImageVersion

要使用的基本 MicroVM 映像版本。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::MicrovmImage` 資源的 [BaseImageVersion](#) 屬性。

BuildRoleArn

MicroVM 組建服務從 Amazon S3 下載程式碼成品並寫入組建日誌時所擔任的 IAM 角色 ARN。

類型：字串

必要：否

CloudFormation 相容性：此屬性類似於 `AWS::Lambda::MicrovmImage` 資源的 [BuildRoleArn](#) 屬性。這是在 CloudFormation 的必要項目，但在 AWS SAM 中不是。如果您未指定角色，會 AWS SAM 建立邏輯 ID 為 `<resource-logical-id>BuildRole` 的角色。如果您的程式碼成品使用中的客戶受管金鑰加密 AWS KMS，您必須提供自己的角色適當的 `kms:Decrypt` 許可。

CodeUri

包含 Dockerfile 和應用程式程式碼之 zip 成品的 Amazon S3 URI。

類型：字串

必要：是

CloudFormation 相容性：此屬性會轉換為 `AWS::Lambda::MicrovmImage` 資源的 [CodeArtifact](#) 屬性。會將 URI AWS SAM 包裝到 `CodeArtifact.Uri` 結構中。

CpuConfigurations

指定 MicroVM 支援 CPU 架構的 [CpuConfiguration](#) 物件清單。如果您未指定值，服務會使用預設 CPU 架構。

類型：清單

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::MicrovmImage` 資源的 [CpuConfigurations](#) 屬性。

Description

MicroVM 映像的描述。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::MicrovmImage` 資源的 [Description](#) 屬性。

EgressNetworkConnectors

MicroVM 在執行時間可用的輸出網路連接器 ARNs 清單。

類型：清單

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::MicrovmImage` 資源的 [EgressNetworkConnectors](#) 屬性。

EnvironmentVariables

在 MicroVM 執行時間環境中設定的環境變數。

類型：映射

必要：否

CloudFormation 相容性：此屬性類似於 `AWS::Lambda::MicrovmImage` 資源的 [EnvironmentVariables](#) 屬性。在 AWS SAM，指定做為映射（例如，`KEY: value`）。AWS SAM 會轉換為 CloudFormation 陣列格式 `[{Key: KEY, Value: value}]`。

Hooks

MicroVMs 和 MicroVM 映像的生命週期掛鉤組態。

類型：[勾點](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::MicrovmImage` 資源的 [Hooks](#) 屬性。

Logging

MicroVM 記錄輸出的組態。僅指定一個：`CloudWatch` 啟用 CloudWatch 記錄，或 `Disabled` 關閉記錄。

類型：[記錄](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::MicrovmImage` 資源的 [Logging](#) 屬性。

Name

MicroVM 映像的名稱。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::MicrovmImage` 資源的 [Name](#) 屬性。

PropagateTags

指定是否要將標籤從 Tags 屬性傳遞至產生的資源，例如自動產生的建置角色。

類型：布林值

必要：否

預設：False

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等的。

Resources

指定 MicroVM 資源需求的[資源](#)物件清單。

類型：清單

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Lambda::MicrovmImage 資源的 [Resources](#) 屬性。

Tags

索引鍵/值對的映射，指定新增至此 MicroVM 映像的標籤。

類型：映射

必要：否

CloudFormation 相容性：此屬性類似於 AWS::Lambda::MicrovmImage 資源的 [Tags](#) 屬性。中的 Tags 屬性 AWS SAM 包含鍵值對（而 CloudFormation 此屬性包含 Tag 物件清單）。此外，AWS SAM 自動將 lambda:createdBy: SAM 標籤新增至此 MicroVM 映像，以及為其產生的預設角色。

傳回值

Ref

當將此資源的邏輯 ID 提供給 Ref 內部 函數時，它會傳回 MicroVM 映像的 ARN。

如需使用 Ref 函數的詳細資訊，請參閱 AWS CloudFormation 《使用者指南[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。

如需使用 的詳細資訊Fn::GetAtt，請參閱AWS CloudFormation 《使用者指南[Fn::GetAtt](#)》中的。

CreatedAt

建立 MicroVM 映像版本的時間戳記。

ImageArn

MicroVM 映像的 ARN。

LatestActiveImageVersion

MicroVM 映像的最新作用中版本。

LatestFailedImageVersion

MicroVM 映像的最新失敗版本，如果有的話。

State

MicroVM 映像的目前狀態。

UpdatedAt

MicroVM 映像版本上次更新的時間戳記。

範例

最小 MicroVM 映像

下列範例只會建立具有所需屬性的 MicroVM 映像。AWS SAM 會自動產生建置角色。

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

Resources:
  MyMicroVM:
    Type: AWS::Serverless::MicrovmImage
    Properties:
      Name: my-app
```

```
CodeUri: s3://my-bucket/app.zip
BaseImageArn: arn:aws:lambda:us-east-2:aws:microvm-image:al2023-1
BaseImageVersion: "0"
```

完整 MicroVM 映像組態

下列範例會建立已設定所有可用屬性的 MicroVM 映像。

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

Resources:
  MyMicroVM:
    Type: AWS::Serverless::MicrovmImage
    Properties:
      Name: my-app
      CodeUri: s3://my-bucket/app.zip
      BaseImageArn: arn:aws:lambda:us-east-2:aws:microvm-image:al2023-1
      BaseImageVersion: "0"
      Description: "Production MicroVM application"
      CpuConfigurations:
        - Architecture: ARM_64
      Resources:
        - MinimumMemoryInMiB: 1024
      AdditionalOsCapabilities:
        - ALL
      EgressNetworkConnectors:
        - arn:aws:lambda:us-east-2:aws:network-connector:aws-network-connector:INTERNET_EGRESS
      Hooks:
        Port: 9000
        MicrovmHooks:
          Run: ENABLED
          RunTimeoutInSeconds: 30
          Suspend: ENABLED
          Resume: ENABLED
        MicrovmImageHooks:
          Ready: ENABLED
          ReadyTimeoutInSeconds: 60
      EnvironmentVariables:
        LOG_LEVEL: info
        APP_ENV: production
      Logging:
        CloudWatch:
```

```
LogGroup: /aws/lambda-microvms/my-app
Tags:
  Environment: Production
  Team: Platform
```

AWS::Serverless::NetworkConnector

建立 AWS Lambda Network Connector，定義 Lambda 運算資源的網路連線能力。網路連接器指定運算資源如何連線至網路，例如使用服務在指定的子網路和安全群組中佈建的彈性網路界面 (ENIs)，透過客戶 VPC 路由輸出流量。每個連接器都提供一種運算類型，可以是 MicroVMs 或 Web 函數。若要搭配 [AWS::Serverless::WebFunction](#) 使用連接器，ComputeTypes 請將設定為 `WebFunction` 並將連接器的 `LatestVersionArn` 傳遞至函數的 `EgressNetworkConnectorArn` 屬性。

Note

當您部署到時 AWS CloudFormation，會將您的 AWS SAM 資源 `AWS::Serverless::NetworkConnector` 轉換為 `aws-networkconnect` 資源。如需詳細資訊，請參閱 [為產生 CloudFormation 資源 AWS SAM](#)。

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::NetworkConnector
Properties:
  ComputeTypes: List
  Name: String
  OperatorRole: String
  PropagateTags: Boolean
  Tags: Map
  VpcConfig: VpcConfig
```

Properties

ComputeTypes

可以使用此網路連接器的 Lambda 運算資源類型。請僅指定下列其中一個值：

- `MicroVm`：MicroVM 運算資源會使用連接器。

- **WebFunction**：連接器由 [AWS::Serverless::WebFunction](#) 資源透過其 `EgressNetworkConnectorArn` 屬性使用。

類型：清單

必要：否

預設：[MicroVm]

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::NetworkConnector` 資源的 [Configuration.VpcEgressConfiguration.AssociatedComputeResourceTypes](#) 屬性。

Name

網路連接器的名稱。名稱在您的帳戶和區域中必須是唯一的。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::NetworkConnector` 資源的 [Name](#) 屬性。

OperatorRole

網路連接器服務用來建立和管理彈性網路介面 (ENIs) 的 IAM 角色 ARN。

類型：字串

必要：否

CloudFormation 相容性：此屬性類似於 `AWS::Lambda::NetworkConnector` 資源的 [OperatorRole](#) 屬性。這是在 CloudFormation 的必要項目，但在 AWS SAM 中不是。如果您未指定角色，AWS SAM 會建立邏輯 ID 為 `<resource-logical-id>OperatorRole` 的角色。

PropagateTags

指定是否要將標籤從 `Tags` 屬性傳遞至產生的資源，例如自動產生的運算子角色。

類型：布林值

必要：否

預設：False

CloudFormation 相容性：此屬性對 AWS SAM 是唯一的，並且沒有 CloudFormation 同等屬性。

Tags

索引鍵/值對的映射，指定新增至此網路連接器的標籤。

類型：映射

必要：否

CloudFormation 相容性：此屬性類似於 `AWS::Lambda::NetworkConnector` 資源的 [Tags](#) 屬性。中的 `Tags` 屬性 AWS SAM 包含鍵值對（而 CloudFormation 此屬性包含 Tag 物件清單）。此外，AWS SAM 會自動將 `lambda:createdBy:SAM` 標籤新增至此網路連接器，以及為其產生的預設角色。

VpcConfig

網路連接器的 VPC 輸出組態。指定透過 VPC 路由傳出流量的子網路、安全群組和網路通訊協定。

類型：[VpcConfig](#)

必要：是

CloudFormation 相容性：此屬性會從 `ComputeTypes` 屬性轉換為 [Configuration.VpcEgressConfiguration](#) `AWS::Lambda::NetworkConnector` resource. AWS SAM sets `AssociatedComputeResourceTypes` 的屬性。[ComputeTypes](#)

傳回值

Ref

當將此資源的邏輯 ID 提供給 `Ref` 內部函數時，它會傳回網路連接器的 ARN。

如需使用 `Ref` 函數的詳細資訊，請參閱 AWS CloudFormation 《使用者指南[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。

如需使用的詳細資訊 `Fn::GetAtt`，請參閱 AWS CloudFormation 《使用者指南[Fn::GetAtt](#)》中的。

Arn

網路連接器的 Amazon Resource Name (ARN)。

LatestVersionArn

網路連接器最新版本的版本限定 ARN，例如 `arn:aws:lambda:us-east-1:123456789012:network-connector:my-connector:1`。將此值用於 [AWS::Serverless::WebFunction](#) 的 `EgressNetworkConnectorArn` 屬性。

State

網路連接器的目前狀態。有效值為 `PENDING`、`ACTIVE`、`INACTIVE`、`DELETING`、`FAILED` 和 `DELETE_FAILED`。

範例

具有自動產生角色的基本網路連接器

下列範例會建立 VPC 輸出網路連接器。AWS SAM 會自動產生運算子角色。

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

Resources:
  MyVpcConnector:
    Type: AWS::Serverless::NetworkConnector
    Properties:
      Name: my-vpc-connector
      VpcConfig:
        SubnetIds:
          - subnet-0abc123def4567890
          - subnet-0def456abc7890123
        SecurityGroupIds:
          - sg-0abc1234abcde0001
        NetworkProtocol: IPv4
      Tags:
        Environment: Production
```

Web 函數的網路連接器

下列範例會建立 Web Functions 的網路連接器，並將其用於 Web Function 的傳出流量。

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

```
Resources:
  MyWebFunctionConnector:
    Type: AWS::Serverless::NetworkConnector
    Properties:
      ComputeTypes:
        - WebFunction
      VpcConfig:
        SubnetIds:
          - subnet-0abc123def4567890
        SecurityGroupIds:
          - sg-0abc1234abcde0001
        NetworkProtocol: IPv4

  MyWebFunction:
    Type: AWS::Serverless::WebFunction
    Properties:
      Runtime: nodejs24.x
      AuthType: IamAuth
      CodeUri: s3://amzn-s3-demo-bucket/webfn.zip
      EgressNetworkConnectorArn: !GetAtt MyWebFunctionConnector.LatestVersionArn
```

VpcConfig

設定網路連接器的 VPC 輸出設定，包括彈性網路介面 (ENI) 佈建的子網路、安全群組和網路通訊協定。

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

```
NetworkProtocol: String
SecurityGroupIds: List
SubnetIds: List
```

Properties

NetworkProtocol

VPC 輸出連線的網路通訊協定。

有效值：IPv4、DualStack

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::NetworkConnector` 資源 `Configuration.VpcEgressConfiguration` 的 [NetworkProtocol](#) 屬性。

SecurityGroupIds

要與彈性網路介面 (ENIs) 建立關聯的安全群組 IDs 清單。

類型：清單

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::NetworkConnector` 資源 `Configuration.VpcEgressConfiguration` 的 [SecurityGroupIds](#) 屬性。

SubnetIds

子網路 IDs 清單，其中服務為 VPC 輸出連線佈建 ENIs。

類型：清單

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::NetworkConnector` 資源 `Configuration.VpcEgressConfiguration` 的 [SubnetIds](#) 屬性。

範例

VpcConfig 範例

```
VpcConfig:
  SubnetIds:
    - subnet-0abc123def4567890
    - subnet-0def456abc7890123
  SecurityGroupIds:
    - sg-0abc1234def567890
  NetworkProtocol: IPv4
```

AWS::Serverless::SimpleTable

使用單一屬性主索引鍵建立 DynamoDB 資料表。當只需要透過主索引鍵存取資料時，此功能非常有用。

如需更進階的功能，請使用中的 [AWS::DynamoDB::Table](#) 資源 CloudFormation。這些資源可用於 AWS SAM。它們是全方位的，並提供進一步的自訂，包括 [key schema](#)和 [resource policy](#) 自訂。

Note

當您部署到時 AWS CloudFormation，會將您的 AWS SAM 資源 AWS SAM 轉換為 CloudFormation 資源。如需詳細資訊，請參閱[為產生 CloudFormation 資源 AWS SAM](#)。

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: ProvisionedThroughputObject
  SSESpecification: SSESpecification
  TableName: String
  Tags: Map
```

Properties

PointInTimeRecoverySpecification

用於啟用時間點復原恢復的設定。

類型：[PointInTimeRecoverySpecification](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::DynamoDB::Table 資源的 [PointInTimeRecoverySpecification](#) 屬性。

PrimaryKey

要用作資料表主索引鍵的屬性名稱和類型。如果未提供，主索引鍵將是值String為的 id。

Note

建立此資源後，無法修改此屬性的值。

類型：[PrimaryKeyObject](#)

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

ProvisionedThroughput

讀取和寫入輸送量佈建資訊。

如果ProvisionedThroughput未指定，BillingMode則會指定為 PAY_PER_REQUEST。

類型：[ProvisionedThroughputObject](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 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

CloudFormation 相容性：此屬性會直接傳遞至 AWS::DynamoDB::Table 資源的 [SSESpecification](#) 屬性。

TableName

DynamoDB 資料表的名稱。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::DynamoDB::Table` 資源的 [TableName](#) 屬性。

Tags

映射（字串到字串），指定要新增至此 SimpleTable 的標籤。如需標籤有效金鑰和值的詳細資訊，請參閱AWS CloudFormation 《使用者指南》中的[資源標籤](#)。

類型：映射

必要：否

CloudFormation 相容性：此屬性類似於 `AWS::DynamoDB::Table` 資源的 [Tags](#) 屬性。SAM 中的標籤屬性包含 Key : Value 對；在 CloudFormation 中包含標籤物件清單。

傳回值

Ref

當將此資源的邏輯 ID 提供給 Ref 內部函數時，它會傳回基礎 DynamoDB 資料表的資源名稱。

如需使用 Ref 函數的詳細資訊，請參閱AWS CloudFormation 《使用者指南[Ref](#)》中的。

範例

SimpleTableExample

SimpleTable 的範例

YAML

```
Properties:
  TableName: my-table
  PrimaryKey:
    Name: MyPrimaryKey
    Type: String
  ProvisionedThroughput:
    ReadCapacityUnits: 5
    WriteCapacityUnits: 5
  Tags:
    Department: Engineering
```

AppType: Serverless

PrimaryKeyObject

描述主索引鍵屬性的物件。

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

```
Name: String  
Type: String
```

Properties

Name

主索引鍵的屬性名稱。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::DynamoDB::Table` `AttributeDefinition` 資料類型的 [AttributeName](#) 屬性。

其他備註：此屬性也會傳遞至 `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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::DynamoDB::Table` `AttributeDefinition` 資料類型的 [AttributeType](#) 屬性。

範例

PrimaryKey

主索引鍵範例。

YAML

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

ProvisionedThroughputObject

描述佈建輸送量屬性的物件。

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

```
ReadCapacityUnits: Integer
WriteCapacityUnits: Integer
```

Properties

ReadCapacityUnits

DynamoDB 傳回 `ThrottlingException` 之前，每秒消耗的強制一致性讀取數目上限。

類型：整數

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::DynamoDB::Table` `ProvisionedThroughput` 資料類型的 [ReadCapacityUnits](#) 屬性。

WriteCapacityUnits

DynamoDB 傳回 `ThrottlingException` 之前，每秒消耗的寫入數目上限。

類型：整數

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::DynamoDB::Table` `ProvisionedThroughput` 資料類型的 [WriteCapacityUnits](#) 屬性。

範例

ProvisionedThroughput

佈建輸送量範例。

YAML

```
Properties:
  ProvisionedThroughput:
    ReadCapacityUnits: 5
    WriteCapacityUnits: 5
```

AWS::Serverless::StateMachine

建立 AWS Step Functions 狀態機器，您可以使用它來協調 AWS Lambda 事件函數和其他 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，會將您的 AWS SAM 資源 AWS SAM 轉換為 CloudFormation 資源。如需詳細資訊，請參閱[為產生 CloudFormation 資源 AWS SAM](#)。

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
UseAliasAsEventTarget: Boolean
Definition: Map
DefinitionSubstitutions: Map
DefinitionUri: String | S3Location
DeploymentPreference: DeploymentPreference
Events: EventSource
Logging: LoggingConfiguration
Name: String
PermissionsBoundary: String
Policies: String | List | Map
PropagateTags: Boolean
RolePath: String
Role: String
Tags: Map
Tracing: TracingConfiguration
Type: String
```

Properties

AutoPublishAlias

狀態機器別名的名稱。若要進一步了解如何使用 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](#)。

使用 DeploymentPreference 來設定別名的部署偏好設定。如果您未指定 DeploymentPreference，AWS SAM 會設定流量一次轉移到較新的狀態機器版本。

AWS SAM Retain 根據預設，會將版本的 DeletionPolicy 和 UpdateReplacePolicy 設為。舊版不會自動刪除。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::StepFunctions::StateMachineAlias 資源的 [Name](#) 屬性。

UseAliasAsEventTarget

指出是否要將使用 AutoPublishAlias 屬性建立的別名傳遞至以 [Events](#) 定義的事件來源目標。

指定 True 以使用別名做為事件的目標。

類型：布林值

必要：否

預設：False

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

Definition

狀態機器定義是一個物件，其中物件的格式與您的 AWS SAM 範本檔案格式相符，例如 JSON 或 YAML。狀態機器定義遵循 [Amazon 狀態語言](#)。

如需內嵌狀態機器定義的範例，請參閱 [範例](#)。

您必須提供 Definition 或 DefinitionUri。

類型：映射

必要：有條件

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

DefinitionSubstitutions

string-to-string 映射，指定狀態機器定義中預留位置變數的映射。這可讓您將執行時間取得的值（例如，從內部函數）注入狀態機器定義。

類型：映射

必要：否

CloudFormation 相容性：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任何內部函數，會將項目 AWS SAM 新增至此屬性，以將它們插入狀態機器定義。

DefinitionUri

以 Amazon [States Language](#) 撰寫之狀態機器定義的 Amazon Simple Storage Service (Amazon S3) URI 或本機檔案路徑。

如果您提供本機檔案路徑，範本必須經過包含 sam deploy 或 sam package 命令的工作流程，才能正確轉換定義。若要這樣做，您必須使用 CLI 的 0.52.0 版或更新版本 AWS SAM。

您必須提供 Definition 或 DefinitionUri。

類型：字串 | [S3Location](#)

必要：有條件

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::StepFunctions::StateMachine` 資源的 [DefinitionS3Location](#) 屬性。

DeploymentPreference

啟用和設定逐步狀態機器部署的設定。若要進一步了解 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](#)。

在設定此屬性 `AutoPublishAlias` 之前，請指定。您的 `DeploymentPreference` 設定將套用至以指定的別名 `AutoPublishAlias`。

當您指定 `DeploymentPreference`，會自動 AWS SAM 產生 `StateMachineVersionArn` 子屬性值。

類型：[DeploymentPreference](#)

必要：否

CloudFormation compatibility：AWS SAM 產生 `StateMachineVersionArn` 屬性值並將其連接至 `AWS::StepFunctions::StateMachineAlias` 資源的 [DeploymentPreference](#) 屬性 `DeploymentPreference`，並將其傳遞 `DeploymentPreference` 至。

Events

指定觸發此狀態機器的事件。事件由類型和一組屬性組成，這些屬性取決於類型。

類型：[EventSource](#)

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

Logging

定義記錄哪些執行歷史記錄事件及其記錄位置。

類型：[LoggingConfiguration](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::StepFunctions::StateMachine` 資源的 [LoggingConfiguration](#) 屬性。

Name

狀態機器的名稱。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::StepFunctions::StateMachine` 資源的 [StateMachineName](#) 屬性。

PermissionsBoundary

用於此狀態機器執行角色的許可界限 ARN。此屬性只有在為您產生角色時才有效。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::IAM::Role` 資源的 [PermissionsBoundary](#) 屬性。

Policies

此狀態機器的許可政策。政策會附加至狀態機器的預設 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 政策範本](#)。
- [AWS 受管政策](#)或[客戶受管政策](#)ARN的。
- 下列[清單中](#)受 AWS 管政策的名稱。
- 在 中格式化YAML為映射的[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

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

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。指定 True 在產生的資源中傳播標籤。

類型：布林值

必要：否

預設：False

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

Role

做為此狀態機器執行角色的 IAM 角色 ARN。

類型：字串

必要：有條件

CloudFormation 相容性：此屬性會直接傳遞至 `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。使用 Role 屬性指定角色時，請勿使用。

類型：字串

必要：有條件

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::IAM::Role` 資源的 [Path](#) 屬性。

Tags

string-to-string 映射，指定新增至狀態機器和對應執行角色的標籤。如需標籤有效金鑰和值的相關資訊，請參閱 [AWS::StepFunctions::StateMachine](#) 資源的 [標籤](#) 屬性。

類型：映射

必要：否

CloudFormation 相容性：此屬性類似於 [Tags](#) `AWS::StepFunctions::StateMachine` resource. AWS SAM automatically 新增 `stateMachine:createdBy:SAM` 標籤至此資源，以及為其產生的預設角色。

Tracing

選取是否 AWS X-Ray 已啟用狀態機器。如需搭配 Step Functions 使用 X-Ray 的詳細資訊，請參閱《AWS Step Functions 開發人員指南》中的 [AWS X-Ray 和 Step Functions](#)。

類型：[TracingConfiguration](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::StepFunctions::StateMachine` 資源的 [TracingConfiguration](#) 屬性。

Type

狀態機器的類型。

有效值：STANDARD 或 EXPRESS

類型：字串

必要：否

預設：STANDARD

CloudFormation 相容性：此屬性會直接傳遞至 `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 內部函數時，Ref 會傳回基礎 `AWS::StepFunctions::StateMachine` 資源的 Amazon Resource Name (ARN)。

如需使用 Ref 函數的詳細資訊，請參閱 AWS CloudFormation 《使用者指南[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。

如需使用 的詳細資訊Fn::GetAtt，請參閱AWS CloudFormation 《使用者指南[Fn::GetAtt](#)》中的。

Name

傳回狀態機器的名稱，例如 HelloWorld-StateMachine。

範例

狀態機器定義檔案

以下是內嵌狀態機器定義的範例，允許 lambda 函數叫用狀態機器。請注意，此範例預期 Role 屬性會設定適當的政策以允許呼叫。my_state_machine.asl.json 檔案必須以 [Amazon States 語言](#) 撰寫。

在此範例中，DefinitionSubstitution項目允許狀態機器包含在 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
```

內嵌狀態機器定義

以下是內嵌狀態機器定義的範例。

在此範例中，AWS SAM 範本檔案是以 YAML 撰寫，因此狀態機器定義也以 YAML 表示。若要以 JSON 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取決於該類型的屬性。如需每個事件來源屬性的詳細資訊，請參閱與該類型對應的子主題。

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
```

Properties

Properties

描述此事件映射屬性的物件。屬性集必須符合定義的 Type。

類型：[排程](#) | [ScheduleV2](#) | [CloudWatchEvent](#) | [EventBridgeRule](#) | [Api](#)

必要：是

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

Type

事件類型。

有效值：Api、Schedule、ScheduleV2、CloudWatchEvent、EventBridgeRule

類型：字串

必要：是

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

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 操作。

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
```

Properties

Auth

此 API、路徑和方法的授權組態。

當未指定 `DefaultAuthorizer` 時，使用此屬性覆寫個別路徑的 API `DefaultAuthorizer` 設定，或覆寫預設設定 `ApiKeyRequired`。

類型：[ApiStateMachineAuth](#)

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

Method

叫用此函數的 HTTP 方法。

類型：字串

必要：是

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

Path

叫用此函數的 URI 路徑。值必須以 開頭/。

類型：字串

必要：是

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

RestApiId

RestApi 資源的識別符，其中必須包含具有指定路徑和方法的操作。一般而言，這會設定為參考此範本中定義的[AWS::Serverless::Api](#)資源。

如果您未定義此屬性，會使用產生的 `OpenApi` 文件 AWS SAM 建立預設[AWS::Serverless::Api](#)資源。該資源包含由相同範本中未指定之 `Api` 事件定義的所有路徑和方法的聯集 `RestApiId`。

此屬性無法參考在另一個範本中定義的 [AWS::Serverless::Api](#) 資源。

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

UnescapeMappingTemplate

在傳遞給狀態機器的輸入上，將 取代 \ 為 ' 來取消單引號。當您的輸入包含單引號時，請使用 。

Note

如果設定為 `False` 且您的輸入包含單引號，則會發生錯誤。

類型：布林值

必要：否

預設值：False

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有同等 CloudFormation 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

為特定 API、路徑和方法設定事件層級的授權。

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
```

Properties

ApiKeyRequired

此 API、路徑和方法需要 API 金鑰。

類型：布林值

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

AuthorizationScopes

要套用至此 API、路徑和方法的授權範圍。

如果您已指定 DefaultAuthorizer 屬性套用的任何範圍，您指定的範圍將會覆寫它。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

Authorizer

特定狀態機器Authorizer的。

如果您已指定 API 的全域授權方，並想要將此狀態機器設為公有，請將 Authorizer 設定為 以覆寫全域授權方NONE。

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

ResourcePolicy

設定此 API 和路徑的資源政策。

類型：[ResourcePolicyStatement](#)

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

範例

StateMachine-Auth

下列範例會在狀態機器層級指定授權。

YAML

```
Auth:
  ApiKeyRequired: true
  Authorizer: NONE
```

ResourcePolicyStatement

為 API 的所有方法和路徑設定資源政策。如需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 資源政策控制對 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
```

Properties

AwsAccountBlacklist

要封鎖 AWS 的帳戶。

類型：字串的清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

AwsAccountWhitelist

要允許 AWS 的帳戶。如需此屬性的範例使用方式，請參閱此頁面底部的範例區段。

類型：字串的清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

CustomStatements

要套用至此 API 的自訂資源政策陳述式清單。如需此屬性的範例使用方式，請參閱此頁面底部的範例區段。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

IntrinsicVpcBlacklist

要封鎖的虛擬私有雲端 (VPCs) 清單，其中每個 VPC 指定為參考，例如[動態參考](#)或[Ref 內部函數](#)。如需此屬性的範例使用方式，請參閱此頁面底部的範例區段。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

IntrinsicVpcWhitelist

要允許的 VPCs 清單，其中每個 VPC 指定為參考，例如 [動態參考](#) 或 Ref [內部函數](#)。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

IntrinsicVpceBlacklist

要封鎖的 VPC 端點清單，其中每個 VPC 端點指定為參考，例如 [動態參考](#) 或 Ref [內部函數](#)。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

IntrinsicVpceWhitelist

要允許的 VPC 端點清單，其中每個 VPC 端點指定為參考，例如 [動態參考](#) 或 Ref [內部函數](#)。如需此屬性的範例使用方式，請參閱此頁面底部的範例區段。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

IpRangeBlacklist

要封鎖的 IP 地址或地址範圍。如需此屬性的範例使用方式，請參閱此頁面底部的範例區段。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

IpRangeWhitelist

要允許的 IP 地址或地址範圍。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

SourceVpcBlacklist

要封鎖的來源 VPC 或 VPC 端點。來源 VPC 名稱必須以 開頭"vpc-"，來源 VPC 端點名稱必須以 開頭"vpce-"。如需此屬性的範例使用方式，請參閱此頁面底部的範例區段。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

SourceVpcWhitelist

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

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

範例

資源政策範例

下列範例會封鎖兩個 IP 地址和來源 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) 會在設定此事件類型時產生 [AWS::Events::Rule](#) 資源。

重要注意事項：[EventBridgeRule](#)是要使用的慣用事件來源類型，而不是 CloudWatchEvent。EventBridgeRule 和 CloudWatchEvent會使用相同的基礎服務、API CloudFormation 和資源。不過，AWS SAM 只會將新功能的支援新增至 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

```

Properties

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：預設事件匯流排

CloudFormation 相容性：此屬性會直接傳遞至 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

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Events::Rule Target 資源的 [Input](#) 屬性。

InputPath

當您不想將整個相符事件傳遞至目標時，請使用 InputPath 屬性來描述要傳遞的事件部分。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 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](#)》中的 [EventBridge 中的事件和事件模式](#)。 EventBridge

類型：[EventPattern](#)

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `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，這會將您的狀態機器設定為 Amazon EventBridge 規則的目標。如需詳細資訊，請參閱《Amazon EventBridge 使用者指南》中的[什麼是 Amazon EventBridge ?](#)。

AWS SAM 會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
EventBusName: String
Input: String
InputPath: String
InputTransformer: InputTransformer
Pattern: EventPattern
RetryPolicy: RetryPolicy
RuleName: String
```

```
State: String  
Target: Target
```

Properties

DeadLetterConfig

設定 EventBridge 在目標呼叫失敗後傳送事件的 Amazon Simple Queue Service (Amazon SQS) 佇列。例如，將事件傳送至不存在的 Lambda 事件函數時，或 EventBridge 沒有足夠的許可來叫用 Lambda 函數時，叫用可能會失敗。如需詳細資訊，請參閱《Amazon EventBridge 使用者指南》中的[事件重試政策和使用無效字母佇列](#)。

類型：[DeadLetterConfig](#)

必要：否

CloudFormation 相容性：此屬性類似於 `AWS::Events::RuleTarget` 資料類型的 [DeadLetterConfig](#) 屬性。此屬性的 AWS SAM 版本包含其他子屬性，以防您想要為您 AWS SAM 建立無效字母佇列。

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：預設事件匯流排

CloudFormation 相容性：此屬性會直接傳遞至 `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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Events::Rule Target` 資源的 [Input](#) 屬性。

InputPath

當您不想將整個相符事件傳遞至目標時，請使用 InputPath 屬性來描述要傳遞的事件部分。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 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](#)。 EventBridge

類型：[InputTransformer](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Events::RuleTarget 資料類型的 [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》中的 EventBridge 中的事件和事件模式](#)。 EventBridge

類型：[EventPattern](#)

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 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 EventBridge 使用者指南》中的事件重試政策和使用無效字母佇列](#)。

類型：[RetryPolicy](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Events::Rule` Target 資料類型的 [RetryPolicy](#) 屬性。

RuleName

規則的名稱。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Events::Rule` 資源的 [State](#) 屬性。

Target

EventBridge 在觸發規則時叫用 AWS 的資源。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

CloudFormation 相容性：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

用來指定 Amazon Simple Queue Service (Amazon SQS) 佇列的物件，其中 EventBridge 會在失敗的目標調用後傳送事件。例如，當將事件傳送至不存在的狀態機器，或沒有足夠的許可來叫用狀態機器時，叫用可能會失敗。如需詳細資訊，請參閱《Amazon EventBridge 使用者指南》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
```

Properties

Arn

指定為無效字母佇列目標之 Amazon SQS 佇列的 Amazon Resource Name (ARN)。

Note

指定 Type 屬性或 Arn 屬性，但不能同時指定兩者。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Events::Rule DeadLetterConfig` 資料類型的 [Arn](#) 屬性。

QueueLogicalId

Type 指定 時 AWS SAM 建立的無效字母佇列自訂名稱。

Note

如果未設定 Type 屬性，則會忽略此屬性。

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

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，並連接必要的[資源型政策](#)，以授予將事件傳送至佇列的規則資源許可。

Note

指定 Type 屬性或 Arn 屬性，但不能同時指定兩者。

有效值：SQS

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

範例

DeadLetterConfig

DeadLetterConfig

YAML

```
DeadLetterConfig:
```

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

Target

設定觸發規則時 EventBridge 叫用 AWS 的資源。

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
```

Properties

Id

目標的邏輯 ID。

的值Id可以包含英數字元、句點 (.)、連字號 (-) 和底線 (_)。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Events::Rule Target 資料類型的 [Id](#) 屬性。

範例

Target

YAML

```
EBRule:
  Type: EventBridgeRule
  Properties:
    Target:
      Id: MyTarget
```

Schedule

描述Schedule事件來源類型的物件，這會將您的狀態機器設定為依排程觸發之 EventBridge 規則的目標。如需詳細資訊，請參閱《Amazon EventBridge 使用者指南》中的[什麼是 Amazon EventBridge ?](#)。

AWS Serverless Application Model (AWS SAM) 會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
```

Properties

DeadLetterConfig

設定 EventBridge 在目標呼叫失敗後傳送事件的 Amazon Simple Queue Service (Amazon SQS) 佇列。例如，將事件傳送至不存在的 Lambda 事件函數時，或 EventBridge 沒有足夠的許可來叫用 Lambda 函數時，叫用可能會失敗。如需詳細資訊，請參閱《Amazon EventBridge 使用者指南》中的[事件重試政策和使用無效字母佇列](#)。

類型：[DeadLetterConfig](#)

必要：否

CloudFormation 相容性：此屬性類似於 `AWS::Events::RuleTarget` 資料類型的 [DeadLetterConfig](#) 屬性。此屬性的 AWS SAM 版本包含其他子屬性，以防您想要為您 AWS SAM 建立無效字母佇列。

Description

規則的說明。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `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` 屬性，但不能同時指定兩者。

類型：布林值

必要：否

CloudFormation 相容性：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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Events::Rule Target` 資源的 [Input](#) 屬性。

Name

規則的名稱。如果您未指定名稱，CloudFormation 會產生唯一的實體 ID，並將該 ID 用於規則名稱。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 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 EventBridge 使用者指南》中的 [事件重試政策和使用無效字母佇列](#)。

類型：[RetryPolicy](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Events::Rule Target 資料類型的 [RetryPolicy](#) 屬性。

RoleArn

調用排程時EventBridge 排程器用於目標的 IAM 角色 ARN。

類型：[RoleArn](#)

必要：否。如果未提供，則會建立新的角色並使用。

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Scheduler::Schedule Target 資料類型的 [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

CloudFormation 相容性：此屬性會直接傳遞至 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 屬性，但不能同時指定兩者。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Events::Rule 資源的 [State](#) 屬性。

Target

EventBridge 在觸發規則時叫用 AWS 的資源。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

CloudFormation 相容性：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 排程事件

CloudWatch 排程事件範例

YAML

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

DeadLetterConfig

用來指定 Amazon Simple Queue Service (Amazon SQS) 佇列的物件，其中 EventBridge 會在失敗的目標調用後傳送事件。例如，當將事件傳送至不存在的狀態機器，或沒有足夠的許可來叫用狀態機器

時，叫用可能會失敗。如需詳細資訊，請參閱《Amazon EventBridge 使用者指南》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
```

Properties

Arn

指定為無效字母佇列目標之 Amazon SQS 佇列的 Amazon Resource Name (ARN)。

Note

指定 Type 屬性或 Arn 屬性，但不能同時指定兩者。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Events::Rule DeadLetterConfig` 資料類型的 [Arn](#) 屬性。

QueueLogicalId

Type 指定 時 AWS SAM 建立的無效字母佇列自訂名稱。

Note

如果未設定 Type 屬性，則會忽略此屬性。

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等屬性。

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，並連接必要的[資源型政策](#)，以授予將事件傳送至佇列的規則資源許可。

Note

指定 Type 屬性或 Arn 屬性，但不能同時指定兩者。

有效值：SQS

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM ，並且沒有 CloudFormation 同等的。

範例

DeadLetterConfig

DeadLetterConfig

YAML

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

Target

設定觸發規則時 EventBridge 叫用 AWS 的資源。

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
```

Properties

Id

目標的邏輯 ID。

的值Id可以包含英數字元、句點 (.)、連字號 (-) 和底線 ()_。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Events::Rule Target 資料類型的 [Id](#) 屬性。

範例

Target

YAML

```
EBRule:
  Type: Schedule
  Properties:
    Target:
      Id: MyTarget
```

ScheduleV2

描述ScheduleV2事件來源類型的物件，會將您的狀態機器設定為排程觸發的 Amazon EventBridge 排程器事件的目標。如需詳細資訊，請參閱[什麼是 Amazon EventBridge 排程器？](#) 《EventBridge 排程器使用者指南》中的 。

AWS Serverless Application Model (AWS SAM) 會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
```

Properties

DeadLetterConfig

設定 EventBridge 在目標呼叫失敗後傳送事件的 Amazon Simple Queue Service (Amazon SQS) 佇列。例如，將事件傳送至不存在的 Lambda 事件函數時，或 EventBridge 沒有足夠的許可來叫用 Lambda 函數時，叫用可能會失敗。如需詳細資訊，請參閱《[EventBridge 排程器使用者指南](#)》中的[為 EventBridge 排程器設定無效字母佇列](#)。EventBridge

類型：[DeadLetterConfig](#)

必要：否

CloudFormation 相容性：此屬性類似於 `AWS::Scheduler::ScheduleTarget` 資料類型的 [DeadLetterConfig](#) 屬性。此屬性的 AWS SAM 版本包含其他子屬性，以防您想要為您 AWS SAM 建立無效字母佇列。

Description

排程的描述。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Scheduler::Schedule` 資源的 [EndDate](#) 屬性。

FlexibleTimeWindow

允許在其中叫用排程的時段組態。

類型：[FlexibleTimeWindow](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Scheduler::Schedule` 資源的 [FlexibleTimeWindow](#) 屬性。

GroupName

要與此排程建立關聯的排程群組名稱。如果未定義，則會使用預設群組。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Scheduler::Schedule` Target 資源的 [Input](#) 屬性。

KmsKeyArn

KMS 金鑰的 ARN，將用於加密客戶資料。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Scheduler::Schedule` 資源的 [KmsKeyArn](#) 屬性。

Name

排程的名稱。如果您未指定名稱，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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Scheduler::Schedule` 資源的 [Name](#) 屬性。

OmitName

根據預設，AWS SAM 會產生並使用 *<State-machine-logical-ID><event-source-name>* 格式的排程名稱。將此屬性設定為 `true`，讓 CloudFormation 產生唯一的實體 ID，並將其用於排程名稱。

類型：布林值

必要：否

預設：`false`

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有同等 CloudFormation 屬性。

PermissionsBoundary

用來設定角色許可邊界的政策 ARN。

 Note

如果 `PermissionsBoundary` 已定義，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

CloudFormation 相容性：此屬性會直接傳遞至 `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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Scheduler::Schedule Target` 資料類型的 [RetryPolicy](#) 屬性。

RoleArn

調用排程時 `EventBridge` 排程器用於目標的 IAM 角色 ARN。

類型：[RoleArn](#)

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Scheduler::Schedule Target` 資料類型的 [RoleArn](#) 屬性。

ScheduleExpression

排程表達式，可決定排程執行的時間和頻率。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Scheduler::Schedule` 資源的 [ScheduleExpressionTimezone](#) 屬性。

StartDate

排程開始調用目標的日期，以 UTC 為單位。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

CloudFormation 相容性：此屬性會直接傳遞至 `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

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Scheduler::Schedule` 資源的 [State](#) 屬性。

範例

定義 ScheduleV2 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-sam-s3-demo-bucket
    Key: my-state-machine.asl.json
    Version: 3
  Policies:
    - LambdaInvokePolicy:
        FunctionName: !Ref MyFunction
```

AWS::Serverless::WebFunction

建立 AWS Lambda Web 函數，透過受管端點提供 HTTP 請求。與為了回應呼叫和事件而執行的 Lambda 事件函數不同，Web 函數會執行您的 Web 伺服器並直接接收 HTTP 請求。從單一 AWS::Serverless::WebFunction AWS SAM 產生 Web 函數、包含程式碼和組態的不可變修訂，以及將流量路由至一或多個修訂的端點。

Note

當您部署到時 AWS CloudFormation，會將您的 AWS SAM 資源 AWS SAM CloudFormation 轉換為 資源。如需詳細資訊，請參閱 [為產生 CloudFormation 資源 AWS SAM](#)。

⚠ Important

您的函數程式碼必須執行 HTTP 伺服器，以監聽 `AWS_LAMBDA_HTTP_ENDPOINT` 環境變數中的主機和連接埠。執行時間不會設定 PORT。根據預設，執行時間 `index.js` 會在部署套件的根目錄執行。使用 [EntryPoint](#) 執行不同的檔案。

⚠ Important

存放部署套件的 Amazon S3 儲存貯體必須符合下列要求：

- 版本控制已啟用。
- 儲存貯體政策會授予 `s3:GetObject` 和 `lambda.amazonaws.com` `s3:GetObjectVersion` 許可。建議您使用 `aws:SourceArn` 符合函數修訂版的 `ArnLike` 條件來限定政策範圍，例如 `arn:aws:lambda:us-east-1:111122223333:web-function/my-function/revision/*`。
- 如果物件使用 SSE-KMS 加密，它會使用客戶受管金鑰，其金鑰政策允許 `lambda.amazonaws.com` 呼叫 `kms:Decrypt`。不支援 Amazon S3 (`aws/s3`) 的 AWS 受管金鑰，因為您無法變更其金鑰政策。

如需儲存貯體政策範例，請參閱《AWS Lambda 開發人員指南》中的 [Lambda 儲存貯體政策](#)。

`CodeUri` 必須在 Amazon S3 中參考物件。`sam build` 和 `sam package` 命令不支援此資源的本機路徑。

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::WebFunction
Properties:
  AuthType: String
  AutoDeploymentMode: String
  CodeUri: String | CodeUri
  EgressNetworkConnectorArn: String
```

```

EndpointDescription: String
EndpointName: String
EndpointType: String
EntryPoint: String
EnvironmentVariables: Map
ExecutionRoleArn: String
FunctionName: String
InlineCode: String
KmsKeyArn: String
LoggingConfig: Map
MaxConcurrencyPerEnvironment: Integer
ReplicaRegions: List | ReplicaRegions
RevisionDescription: String
RevisionWeights: RevisionWeights
Runtime: String
ScalingConfig: Map
Tags: Map
ThrottleConfig: Map
Timeout: Integer

```

Properties

AuthType

端點如何授權請求。

- **IamAuth** : Lambda 使用 Signature 第 4 版和 IAM 授權請求。呼叫者需要透過 函數上的資源型政策授予的 `lambda:InvokeWebFunctionEndpoint` 許可。
- **ApplicationManaged** : 端點為公有。您的函數必須授權每個請求本身。

有效值 : `IamAuth`、`ApplicationManaged`

類型 : 字串

必要 : 是

CloudFormation 相容性 : 此屬性會直接傳遞至 `AWS::Lambda::WebFunctionEndpoint` 資源的 [AuthType](#) 屬性。

AutoDeploymentMode

端點如何跨修訂路由流量。

- **LatestRevision** : 會將 100% 的流量傳送至此部署建立的修訂版。

- Disabled：根據 [RevisionWeights](#) 路由流量。

您可以使用哪些值，以及是否需要RevisionWeights，取決於 [EndpointType](#)：

- HomeRegion：使用任一值。使用時Disabled，RevisionWeights為必要項目。
- MultiRegion 或 PerRegion：使用 Disabled。如果您省略 RevisionWeights，端點會將 100% 的流量路由到相同堆疊中建立的修訂版，因此您可以在單一部署中建立端點。流量不會移至稍後部署建立的修訂版。

當 AWS SAM 轉換範本時，值必須解析。支援參數Ref的。不支援其他內部函數。

有效值：LatestRevision、Disabled

類型：字串

必要：否

預設：LatestRevision

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

CodeUri

部署套件的 Amazon S3 位置。使用 Bucket、Key和選用Version金鑰指定 s3://*bucket/key* URI 或 物件。如需儲存貯體需求，請參閱此頁面頂端的備註。

類型：字串 | [CodeUri](#)

必要：是

CloudFormation 相容性：此屬性類似於 AWS::Lambda::WebFunctionRevision 資源的 [BuildConfig.CodeConfig.S3Object](#) 屬性。會將 s3:// URI AWS SAM 轉換為 Bucket、Key和 VersionId。

EgressNetworkConnectorArn

網路連接器的版本限定 ARN，透過 VPC 路由函數的傳出流量。ARN 必須以版本編號結尾，例如 arn:aws:lambda:us-east-1:123456789012:network-connector:my-connector:1。若要在相同範本中使用最新版本的連接器，請指定 `!GetAtt MyConnector.LatestVersionArn`。連接器的 [ComputeTypes](#) 必須是 WebFunction。

如果您指定常值字串，會 AWS SAM 檢查是否為符合版本資格的網路連接器 ARN。內部函數由服務傳遞和驗證。變更此值會建立新的修訂。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::WebFunctionRevision` 資源的 [ServiceConfig.EgressNetworkConnectorArn](#) 屬性。

EndpointDescription

端點的描述。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::WebFunctionEndpoint` 資源的 [Description](#) 屬性。

EndpointName

端點的名稱。變更名稱會取代端點。變更 `EndpointType` 也會取代端點，因此 `EndpointName` 請同時變更。如果沒有新名稱，CloudFormation 則無法取代自訂命名的資源。

類型：字串

必要：否

預設：prod

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::WebFunctionEndpoint` 資源的 [EndpointName](#) 屬性。

EndpointType

端點提供流量的位置。

- `HomeRegion`：從堆疊的 區域提供服務。
- `MultiRegion`：會將函數複寫至 [ReplicaRegions](#) 中的區域，並將每個請求路由至最近的區域。
- `PerRegion`：在每個複本區域中公開個別的端點。

`MultiRegion` 和 `PerRegion` 需要 `AutoDeploymentMode: Disabled`。

有效值：`HomeRegion`、`MultiRegion`、`PerRegion`

類型：字串

必要：否

預設：HomeRegion

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::WebFunctionEndpoint` 資源的 [EndpointType](#) 屬性。

EntryPoint

執行時間執行的檔案，相對於部署套件的根目錄。會將它 AWS SAM 傳遞至預留 `AWS_LAMBDA_NODEJS_ENTRYPOINT` 環境變數中的執行時間。

當 `EnvironmentVariables` 是內部 函數時，您無法使用此屬性。

類型：字串

必要：否

預設：index.js

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

EnvironmentVariables

要設定為修訂版環境變數的鍵/值對。

- 金鑰必須以字母開頭，且只包含字母、數字和底線。
- 值必須為非空白且最多 4,096 個字元。
- AWS SAM 會將數字和布林值轉換為字串。

類型：映射

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::WebFunctionRevision` 資源的 [ServiceConfig.EnvironmentVariables](#) 屬性。

ExecutionRoleArn

函數執行時所擔任之 IAM 角色的 ARN。如果您未指定角色，會 AWS SAM 建立一個具有 `AWSLambdaBasicExecutionRole` 受管政策且邏輯 ID 為的角色 `<resource-logical-id>Role`。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::WebFunctionRevision` 資源的 [ServiceConfig.ExecutionRoleArn](#) 屬性。

FunctionName

函數的名稱。函數名稱在所有區域中的帳戶內必須是唯一的。如果您未指定名稱，會從邏輯 ID 和堆疊 ID AWS SAM 產生名稱，最多 64 個字元。變更名稱會取代函數、其修訂版及其端點。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::WebFunction`、`AWS::Lambda::WebFunctionRevision` 和 `AWS::Lambda::WebFunctionEndpoint` 資源的 [FunctionName](#) 屬性。

InlineCode

不支援 `AWS::Serverless::WebFunction`。使用 [CodeUri](#)。

類型：字串

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

KmsKeyArn

用於加密修訂環境變數之 AWS KMS 金鑰的 ARN。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::WebFunctionRevision` 資源的 [KmsKeyArn](#) 屬性。

LoggingConfig

修訂版的記錄組態，例如日誌群組和日誌層級。

類型：映射

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::WebFunctionRevision` 資源的 [ServiceConfig.TelemetryConfig.LoggingConfig](#) 屬性。

MaxConcurrencyPerEnvironment

每個執行環境處理的並行請求數目上限。

類型：整數

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::WebFunctionRevision` 資源的 [ServiceConfig.MaxConcurrencyPerEnvironment](#) 屬性。

ReplicaRegions

要複寫函數的區域。將此屬性與 `MultiRegion` 和 `PerRegion` 端點搭配使用。指定區域代碼和區域群組的清單，或具有 `IncludeRegions` 和選用的物件 `ExcludeRegions`。

類型：List | [ReplicaRegions](#)

必要：否

CloudFormation 相容性：此屬性類似於 `AWS::Lambda::WebFunctionEndpoint` resource. AWS SAM expands 區域群組的 [Regions](#) 屬性，並適用於 `ExcludeRegions` 產生區域清單。

RevisionDescription

此部署建立之修訂的描述。

類型：字串

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::WebFunctionRevision` 資源的 [Description](#) 屬性。

RevisionWeights

修訂 IDs 到流量權重的映射。當 `AutoDeploymentMode` 為 `時`，使用此屬性 `Disabled`。指定最多兩個修訂。每個權重是從 1 到 100 的整數，且權重最多必須加到 100。

類型：[RevisionWeights](#)

必要：有條件限制。當 `AutoDeploymentMode` 為 `Disabled` 且 `EndpointType` 為 `ANY` 時為必要 `HomeRegion`。

CloudFormation 相容性：此屬性類似於 `AWS::Lambda::WebFunctionEndpoint` resource。AWS SAM convert the map to a list of `RevisionId` 和 `Weight` 物件的 [RevisionWeights](#) 屬性。

Runtime

函數的執行時間，例如 `nodejs24.x`。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::WebFunctionRevision` 資源的 [BuildConfig.RuntimeConfig.Runtime](#) 屬性。

ScalingConfig

端點的擴展組態，例如執行環境的最大數量。

類型：映射

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::WebFunctionEndpoint` 資源的 [ScalingConfig](#) 屬性。

Tags

索引鍵/值對的映射，指定新增至函數和 AWS SAM 產生的執行角色的標籤。

類型：映射

必要：否

CloudFormation 相容性：此屬性類似於 `AWS::Lambda::WebFunction` 資源的 [Tags](#) 屬性。中的 `Tags` AWS SAM 屬性包含鍵值對（而 CloudFormation 此屬性包含 `Tag` 物件清單）。此外，AWS SAM 會自動將 `lambda:createdBy: SAM` 標籤新增至函數，以及為其產生的預設角色。

ThrottleConfig

端點的限流組態，例如請求率上限。

類型：映射

必要：否

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::WebFunctionEndpoint` 資源的 [ThrottleConfig](#) 屬性。

Timeout

函數可依請求執行的時間上限，以秒為單位。有效值為 3 到 900。

類型：整數

必要：否

預設：30

CloudFormation 相容性：此屬性會直接傳遞至 `AWS::Lambda::WebFunctionRevision` 資源的 [ServiceConfig.TimeoutSeconds](#) 屬性。

傳回值

Ref

當將此資源的邏輯 ID 提供給 `Ref` 內部函數時，它會傳回 Web 函數的 ARN。如果您需要函數名稱，請 `FunctionName` 改用的值。

如需使用 `Ref` 函數的詳細資訊，請參閱 AWS CloudFormation 《使用者指南[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。

如需使用的詳細資訊 `Fn::GetAtt`，請參閱 AWS CloudFormation 《使用者指南[Fn::GetAtt](#)》中的。

FunctionArn

Web 函數的 Amazon Resource Name (ARN)。

State

Web 函數的目前狀態。

StateReason

Web 函數目前狀態的原因。

CreatedAt

Web 函數建立的時間。

UpdatedAt

Web 函數上次更新的時間。

若要參考 AWS SAM 產生的修訂或端點，請使用其邏輯 IDs `<resource-logical-id>Revision` 和 `<resource-logical-id>Endpoint`。例如，會 `!GetAtt MyWebFunctionEndpoint.DomainName` 傳回邏輯 ID 為 `MyWebFunctionEndpoint` 之 Web 函數的端點網域名稱 `MyWebFunction`。如需詳細資訊，請參閱 [CloudFormation 指定 AWS::Serverless::WebFunction 時產生的資源](#)。

範例

基本 Web 函數

下列範例會在堆疊的 Region 中使用 IAM 授權端點建立 Web 函數。AWS SAM 會產生執行角色，而端點會將所有流量路由至此部署建立的修訂版。

```
AWS::Serverless::WebFunction
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31

Resources:
  MyWebFunction:
    Type: AWS::Serverless::WebFunction
    Properties:
      Runtime: nodejs24.x
      AuthType: IamAuth
      CodeUri:
        Bucket: amzn-s3-demo-bucket
        Key: webfn.zip
        Version: object-version-id
      Timeout: 30

Outputs:
  EndpointUrl:
    Value: !Sub https://${MyWebFunctionEndpoint.DomainName}
```

在兩個修訂之間分割流量

下列範例會停用自動部署，並將 90% 的流量傳送至一個現有修訂版本，並將 10% 傳送至另一個修訂版本。兩個修訂必須已經存在。

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

Resources:
  MyWebFunction:
    Type: AWS::Serverless::WebFunction
    Properties:
      Runtime: nodejs24.x
      AuthType: IamAuth
      CodeUri: s3://amzn-s3-demo-bucket/webfn-v2.zip
      AutoDeploymentMode: Disabled
      RevisionWeights:
        rev-abc123: 90
        rev-def456: 10
```

多區域 Web 函數

下列範例會將函數複寫到美國區域，但 us-west-1 和 會將每個請求路由到最近的區域。由於 RevisionWeights 已省略，端點會將所有流量路由到在此堆疊中建立的修訂。

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

Resources:
  MyWebFunction:
    Type: AWS::Serverless::WebFunction
    Properties:
      Runtime: nodejs24.x
      AuthType: IamAuth
      CodeUri:
        Bucket: amzn-s3-demo-bucket
        Key: webfn.zip
        Version: object-version-id
      EndpointName: prodmr
      EndpointType: MultiRegion
      AutoDeploymentMode: Disabled
      ReplicaRegions:
        IncludeRegions:
```

```
- US
ExcludeRegions:
- us-west-1
```

具有 VPC 輸出的 Web 函數

下列範例會為 Web 函數建立網路連接器，並透過連接器的子網路和安全群組路由函數的傳出流量。

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

Resources:
  MyConnector:
    Type: AWS::Serverless::NetworkConnector
    Properties:
      ComputeTypes:
        - WebFunction
      VpcConfig:
        SubnetIds:
          - subnet-0abc123def4567890
          - subnet-0def456abc7890123
        SecurityGroupIds:
          - sg-0abc1234abcde0001
        NetworkProtocol: IPv4

  MyWebFunction:
    Type: AWS::Serverless::WebFunction
    Properties:
      Runtime: nodejs24.x
      AuthType: IamAuth
      CodeUri:
        Bucket: amzn-s3-demo-bucket
        Key: webfn.zip
        Version: object-version-id
      EgressNetworkConnectorArn: !GetAtt MyConnector.LatestVersionArn
```

CodeUri

指定 Web 函數部署套件的 Amazon S3 位置。您可以使用 `s3://bucket/key` 選用的？
`versionId=version` 查詢參數，將位置指定為物件，或指定為表單的字串。

儲存貯體必須已啟用版本控制，以及授予 `lambda.amazonaws.com` `s3:GetObject` 和 `s3:GetObjectVersion` 許可的儲存貯體政策。建議您將政策的範圍限定為符合函數修訂

版aws:SourceArn的條件。如果物件使用 SSE-KMS 加密，請使用客戶受管金鑰，其金鑰政策允許 lambda.amazonaws.com 呼叫 kms:Decrypt。不支援 Amazon S3 (aws/s3) 的 AWS 受管金鑰。如需詳細資訊，請參閱《AWS Lambda 開發人員指南》中的[從 Amazon S3 部署 Web 函數程式碼](#)。

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
```

Properties

Bucket

包含部署套件的 Amazon S3 儲存貯體名稱。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Lambda::WebFunctionRevision 資源[BuildConfig.CodeConfig.S3Object](#)的 Bucket 屬性。

Key

部署套件的 Amazon S3 金鑰。

類型：字串

必要：是

CloudFormation 相容性：此屬性會直接傳遞至 AWS::Lambda::WebFunctionRevision 資源[BuildConfig.CodeConfig.S3Object](#)的 Key 屬性。

Version

部署套件物件的版本 ID。指定版本以將修訂固定到特定上傳。

類型：字串

必要：否

CloudFormation 相容性：此屬性類似於 `AWS::Lambda::WebFunctionRevision` resource。
AWS SAM passes 值為 [BuildConfig.CodeConfig.S3Object](#) 的 `VersionId` 屬性 `VersionId`。

範例

物件表單

```
CodeUri:
  Bucket: amzn-s3-demo-bucket
  Key: webfn.zip
  Version: object-version-id
```

URI 表單

```
CodeUri: s3://amzn-s3-demo-bucket/webfn.zip?versionId=object-version-id
```

ReplicaRegions

指定 `MultiRegion` 或 `PerRegion` Web 函數端點複寫函數的區域。您可以指定區域代碼和區域群組的清單，或包含區域並選擇性地排除其中部分區域的物件。

您可以在指定區域的任何位置使用下列區域群組：

- US: `us-east-1`, `us-east-2`, `us-west-1`, `us-west-2`
- EU: `eu-west-1`, `eu-west-2`, `eu-west-3`, `eu-central-1`, `eu-north-1`
- AP: `ap-south-1`, `ap-northeast-1`, `ap-northeast-2`, `ap-northeast-3`, `ap-southeast-1`, `ap-southeast-2`
- CA: `ca-central-1`
- SA: `sa-east-1`
- `SPARSE_GLOBAL`：每個洲一或兩個區域，涵蓋較少區域的全球範圍：`us-east-1`、`eu-west-1`、`eu-central-1`、`ap-northeast-1`、`ap-southeast-1`、`ap-southeast-2`、`sa-east-1`
- `ALL_STANDARD`：US、CA、和 SA 群組中的每個區域 EU AP (17 個區域)

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

```
ExcludeRegions: List  
IncludeRegions: List
```

Properties

ExcludeRegions

要從 移除的區域代碼和區域群組清單IncludeRegions。您也必須指定 IncludeRegions。項目不能是內部函數，而且當 IncludeRegions包含內部函數ExcludeRegions時，您無法使用。

類型：清單

必要：否

CloudFormation 相容性：此屬性對 是唯一的 AWS SAM，並且沒有 CloudFormation 同等屬性。

IncludeRegions

要複寫函數的區域代碼和區域群組清單。清單本身不能是內部 函數，但個別項目可以。

類型：清單

必要：否

CloudFormation 相容性：此屬性類似於 AWS::Lambda::WebFunctionEndpoint resource。
AWS SAM expands 區域群組的 [Regions](#) 屬性，並移除任何ExcludeRegions項目以產生區域清單。

範例

區域清單

下列範例會將函數複寫到兩個區域。

```
ReplicaRegions:  
  - us-east-1  
  - eu-west-1
```

具有排除的區域群組

下列範例會將 函數複寫至美國區域，但 除外us-west-1。

```
ReplicaRegions:
  IncludeRegions:
    - US
  ExcludeRegions:
    - us-west-1
```

RevisionWeights

指定 Web 函數端點如何分割跨修訂的流量。在 `AutoDeploymentMode` 為 `Disabled` 時使用此屬性，例如執行 Canary 或藍/綠部署。

`RevisionWeights` 是映射。每個金鑰都是修訂 ID，而每個值都是要路由至該修訂的流量百分比。適用的規定如下：

- 指定最多兩個修訂。
- 每個權重是從 1 到 100 的整數，且權重最多必須加到 100。
- 個別權重可以是內部函數，但映射本身不能是內部函數。

類型：映射

CloudFormation 相容性：此屬性類似於 `AWS::Lambda::WebFunctionEndpoint` 資源的 [RevisionWeights](#) 屬性。會將映射 AWS SAM 轉換為物件清單，每個物件都具有 `RevisionId` 和 `Weight`。

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

```
revision-id: Integer
revision-id: Integer
```

範例

Canary 分割

下列範例會將 90% 的流量傳送至 `rev-abc123`，並將 10% 流量傳送至 `rev-def456`。

```
AutoDeploymentMode: Disabled
```



```
RevisionWeights:
  rev-abc123: 90
  rev-def456: 10
```

AWS SAM 會將此轉換為下列端點屬性：

```
RevisionWeights:
  - RevisionId: rev-abc123
    Weight: 90
  - RevisionId: rev-def456
    Weight: 10
```

為產生 CloudFormation 資源 AWS SAM

本節提供處理 AWS SAM AWS 範本時所建立 CloudFormation 資源的詳細資訊。AWS SAM 產生的 CloudFormation 資源集會因您指定的案例而有所不同。案例是範本檔案中指定的資源和屬性的 AWS SAM 組合。您可以參考範本檔案中其他位置產生的 CloudFormation 資源，類似於您在範本檔案中明確宣告的資源參考方式。

例如，如果您在 AWS SAM 範本檔案中指定 `AWS::Serverless::Function` 資源，AWS SAM 一律會產生 `AWS::Lambda::Function` 基本資源。如果您也指定選用 `AutoPublishAlias` 屬性，則會 AWS SAM 另外產生 `AWS::Lambda::Version` `AWS::Lambda::Alias` 和資源。

本節列出案例及其產生的 CloudFormation 資源，並說明如何參考 AWS SAM 範本檔案中產生的 CloudFormation 資源。

參考產生的 CloudFormation 資源

您有兩個選項可參考 AWS SAM 範本檔案中產生的 CloudFormation 資源，`LogicalId` 或參考屬性。

參考 LogicalId 產生的 CloudFormation 資源

AWS SAM 產生每個 CloudFormation 的資源都有一個 [LogicalId](#)，這是範本檔案中唯一的英數字元 (A-Z、a-z、0-9) 識別符。AWS SAM 會使用範本檔案中 `LogicalIds` AWS SAM 的資源的來建構其產生的 CloudFormation 資源 `LogicalIds` 的。您可以使用產生的 CloudFormation 資源 `LogicalId` 的來存取範本檔案中該資源的屬性，就像 CloudFormation 您明確宣告的資源一樣。如需 CloudFormation 和 AWS SAM 範本 `LogicalIds` 中的詳細資訊，請參閱 AWS CloudFormation 《使用者指南》中的 [資源](#)。

Note

部分產生的資源LogicalIds的 包含唯一的雜湊值，以避免命名空間衝突。堆疊建立時，會衍生這些資源LogicalIds的。您只能在使用 AWS 管理主控台 AWS CLI或其中一個 AWS SDKs 建立堆疊之後擷取它們。我們不建議 參考這些資源，LogicalId因為雜湊值可能會變更。

透過可參考屬性參考產生的 CloudFormation 資源

對於某些產生的資源，AWS SAM 會提供資源的 AWS SAM 可參考屬性。您可以使用此屬性來參考 AWS SAM 範本檔案中產生的 CloudFormation 資源及其屬性。

Note

並非所有產生的 CloudFormation 資源都有可參考的屬性。對於這些資源，您必須使用 LogicalId。

產生的 CloudFormation 資源案例

下表摘要說明組成產生 AWS SAM 資源之案例 CloudFormation 的資源和屬性。案例欄中的主題提供有關為該案例產生之額外 CloudFormation 資源 AWS SAM 的詳細資訊。

AWS SAM 資源	基本 CloudFormation 資源	案例
<u>AWS::Serverless::Api</u>	<u>AWS::ApiGateway::RestApi</u>	<ul style="list-style-type: none"> <u>已指定 DomainName 屬性</u> <u>已指定 UsagePlan 屬性</u>
<u>AWS::Serverless::Application</u>	<u>AWS::CloudFormation::Stack</u>	<ul style="list-style-type: none"> 除了產生基本 CloudFormation 資源之外，此無伺服器資源沒有其他案例。

AWS SAM 資源	基本 CloudFormation 資源	案例
<u>AWS::Serverless::CapacityProvider</u>	<u>AWS::Lambda::CapacityProvider</u>	<ul style="list-style-type: none"> • <u>未指定 OperatorRole 屬性</u>
<u>AWS::Serverless::MicrovmImage</u>	<u>AWS::Lambda::MicrovmImage</u>	<ul style="list-style-type: none"> • <u>未指定 BuildRoleArn 屬性</u>
<u>AWS::Serverless::NetworkConnector</u>	<u>AWS::Lambda::NetworkConnector</u>	<ul style="list-style-type: none"> • <u>未指定 OperatorRole 屬性</u>
<u>AWS::Serverless::WebFunction</u>	<u>AWS::Lambda::WebFunction</u>	<ul style="list-style-type: none"> • <u>未指定 ExecutionRoleArn 屬性</u>

AWS SAM 資源	基本 CloudFormation 資源	案例
<u>AWS::Serverless::Function</u>	<u>AWS::Lambda::Function</u>	<ul style="list-style-type: none"> • <u>已指定 AutoPublishAlias 屬性</u> • <u>未指定角色屬性</u> • <u>已指定 DeploymentPreference 屬性</u> • <u>已指定 Api 事件來源</u> • <u>已指定 HttpApi 事件來源</u> • <u>已指定串流事件來源</u> • <u>已指定事件橋接（或事件匯流排）事件來源</u> • <u>已指定 IoTRule 事件來源</u> • <u>針對 Amazon SNS 事件指定 OnSuccess（或 OnFailure）屬性 Amazon SNS</u> • <u>針對 Amazon SQS 事件指定 OnSuccess（或 OnFailure）屬性 Amazon SQS</u>
<u>AWS::Serverless::HttpApi</u>	<u>AWS::ApiGatewayV2::Api</u>	<ul style="list-style-type: none"> • <u>已指定 StageName 屬性</u> • <u>未指定 StageName 屬性</u> • <u>已指定 DomainName 屬性</u>
<u>AWS::Serverless::LayerVersion</u>	<u>AWS::Lambda::LayerVersion</u>	<ul style="list-style-type: none"> • 除了產生基本 CloudFormation 資源之外，此無伺服器資源沒有其他案例。

AWS SAM 資源	基本 CloudFormation 資源	案例
<u>AWS::Serverless::SimpleTable</u>	<u>AWS::DynamoDB::Table</u>	<ul style="list-style-type: none"> 除了產生基本 CloudFormation 資源之外，此無伺服器資源沒有其他案例。
<u>AWS::Serverless::StateMachine</u>	<u>AWS::StepFunctions::StateMachine</u>	<ul style="list-style-type: none"> <u>未指定角色屬性</u> <u>已指定 API 事件來源</u> <u>已指定事件橋接（或事件匯流排）事件來源</u>

主題

- [CloudFormation 指定 AWS::Serverless::Api 時產生的資源](#)
- [CloudFormation 指定 AWS::Serverless::Application 時產生的資源](#)
- [CloudFormation 指定 AWS::Serverless::CapacityProvider 時產生的資源](#)
- [CloudFormation 當您指定 時產生的資源 AWS::Serverless::Connector](#)
- [CloudFormation 指定 AWS::Serverless::Function 時產生的資源](#)
- [CloudFormation 指定 AWS::Serverless::GraphQLApi 時產生的資源](#)
- [CloudFormation 指定 AWS::Serverless::HttpApi 時產生的資源](#)
- [CloudFormation 指定 AWS::Serverless::MicrovmImage 時產生的資源](#)
- [CloudFormation 指定 AWS::Serverless::NetworkConnector 時產生的資源](#)
- [CloudFormation 指定 AWS::Serverless::WebFunction 時產生的資源](#)
- [CloudFormation 指定 AWS::Serverless::WebSocketApi 時產生的資源](#)
- [CloudFormation 指定 AWS::Serverless::LayerVersion 時產生的資源](#)
- [CloudFormation 指定 AWS::Serverless::SimpleTable 時產生的資源](#)
- [CloudFormation 指定 AWS::Serverless::StateMachine 時產生的資源](#)

CloudFormation 指定 AWS::Serverless::Api 時產生的資源

指定 AWS::Serverless::Api 時，AWS Serverless Application Model (AWS SAM) 一律會產生 AWS::ApiGateway::RestApi 基本 CloudFormation 資源。此外，它也會一律產生 AWS::ApiGateway::Stage 和 AWS::ApiGateway::Deployment 資源。

AWS::ApiGateway::RestApi

LogicalId: <api-LogicalId>

可參考屬性：N/A（您必須使用 LogicalId 來參考此 CloudFormation 資源）

AWS::ApiGateway::Stage

LogicalId: <api-LogicalId><stage-name>Stage

<stage-name> 是 StageName 屬性設定為的字串。例如，如果您將 StageName 設定為 Gamma，則 LogicalId 為 *MyRestApiGammaStage*。

可參考屬性：*<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 *MyRestApiDeployment926eeb5ff1*。

可參考屬性：*<api-LogicalId>.Deployment*

除了這些 CloudFormation 資源之外，指定 AWS::Serverless::Api 時，還會為下列案例 AWS SAM 產生其他 CloudFormation 資源。

案例

- [已指定 DomainName 屬性](#)
- [已指定 UsagePlan 屬性](#)

已指定 DomainName 屬性

AWS::Serverless::Api 指定 DomainName 的 Domain 屬性時，AWS SAM 會產生 AWS::ApiGateway::DomainName CloudFormation 資源。

AWS::ApiGateway::DomainName

LogicalId: ApiGatewayDomainName<sha>

<sha> 是建立堆疊時產生的唯一雜湊值。例如：ApiGatewayDomainName926eeb5ff1。

可參考屬性：<api-LogicalId>.DomainName

已指定 UsagePlan 屬性

AWS::Serverless::Api 指定 UsagePlan 的 Auth 屬性時，AWS SAM 會產生下列 CloudFormation 資源：AWS::ApiGateway::UsagePlan、AWS::ApiGateway::UsagePlanKey 和 AWS::ApiGateway::ApiKey。

AWS::ApiGateway::UsagePlan

LogicalId: <api-LogicalId>UsagePlan

可參考屬性：<api-LogicalId>.UsagePlan

AWS::ApiGateway::UsagePlanKey

LogicalId: <api-LogicalId>UsagePlanKey

可參考屬性：<api-LogicalId>.UsagePlanKey

AWS::ApiGateway::ApiKey

LogicalId: <api-LogicalId>ApiKey

可參考屬性：<api-LogicalId>.ApiKey

CloudFormation 指定 AWS::Serverless::Application 時產生的資源

指定 AWS::Serverless::Application 時，AWS Serverless Application Model (AWS SAM) 會產生 AWS::CloudFormation::Stack 基本 CloudFormation 資源。

AWS::CloudFormation::Stack

LogicalId: <application-LogicalId>

可參考屬性：N/A（您必須使用 LogicalId 來參考此 CloudFormation 資源）

CloudFormation 指定 AWS::Serverless::CapacityProvider 時產生的資源

指定 AWS::Serverless::CapacityProvider 時，AWS Serverless Application Model (AWS SAM) 會產生 AWS::Lambda::CapacityProvider 基本 CloudFormation 資源。

AWS::Lambda::CapacityProvider

LogicalId: **<capacityprovider-LogicalId>**

可參考屬性：N/A（您必須使用 LogicalId 來參考此 CloudFormation 資源）

除了此 CloudFormation 資源之外，AWS::Serverless::CapacityProvider 指定時，AWS SAM 也會為下列案例產生 CloudFormation 資源：

案例

- [未指定 OperatorRole 屬性](#)

未指定 OperatorRole 屬性

AWS::Serverless::CapacityProvider 未指定的 OperatorRole 屬性時，AWS SAM 會產生已連接 AWSLambdaManagedEC2ResourceOperator 受管政策 AWS::IAM::Role CloudFormation 的資源。

AWS::IAM::Role

LogicalId: **<capacityprovider-LogicalId>OperatorRole**

可參考屬性：N/A（您必須使用 LogicalId 來參考此 CloudFormation 資源）

CloudFormation 當您指定時產生的資源 AWS::Serverless::Connector

Note

當您透過內嵌 Connectors 屬性定義連接器時，它會先轉換為 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，會視需要 AWS SAM 產生下列 AWS CloudFormation 資源。

AWS::IAM::ManagedPolicy

LogicalId: *<connector-LogicalId>*Policy

可參考屬性：N/A（若要參考此 CloudFormation 資源，您必須使用 LogicalId。）

AWS::SNS::TopicPolicy

LogicalId: *<connector-LogicalId>*TopicPolicy

可參考屬性：N/A（若要參考此 CloudFormation 資源，您必須使用 LogicalId。）

AWS::SQS::QueuePolicy

LogicalId: *<connector-LogicalId>*QueuePolicy

可參考屬性：N/A（若要參考此 CloudFormation 資源，您必須使用 LogicalId。）

AWS::Lambda::Permission

LogicalId: *<connector-LogicalId>**<permission>*LambdaPermission

<permission> 是由 Permissions 屬性指定的許可。例如 Write。

可參考屬性：N/A（若要參考此 CloudFormation 資源，您必須使用 LogicalId。）

CloudFormation 指定 AWS::Serverless::Function 時產生的資源

指定 AWS::Serverless::Function 時，AWS Serverless Application Model (AWS SAM) 一律會建立 AWS::Lambda::Function 基本 CloudFormation 資源。

AWS::Lambda::Function

LogicalId: *<function-LogicalId>*

可參考屬性：N/A（您必須使用 LogicalId 來參考此 CloudFormation 資源）

除了此 CloudFormation 資源之外，AWS::Serverless::Function 指定時，AWS SAM 也會為下列案例產生 CloudFormation 資源。

案例

- [核心函數屬性](#)

- [事件來源](#)
- [事件組態](#)

核心函數屬性

下列案例會根據核心函數屬性產生 CloudFormation 資源：

未指定角色屬性

AWS::Serverless::Function 未指定的 Role 屬性時，AWS SAM 會產生 AWS::IAM::Role CloudFormation 資源。

AWS::IAM::Role

LogicalId: <function-LogicalId>Role

可參考屬性：N/A（您必須使用 LogicalId 來參考此 CloudFormation 資源）

已指定 AutoPublishAlias 屬性

AWS::Serverless::Function 指定的 AutoPublishAlias 屬性時，AWS SAM 會產生下列 CloudFormation 資源：AWS::Lambda::Alias 和 AWS::Lambda::Version。

AWS::Lambda::Alias

LogicalId: <function-LogicalId>Alias<alias-name>

<alias-name> 是設定為 AutoPublishAlias 的字串。例如，如果您將 AutoPublishAlias 設定為 live，則 LogicalId 為：*MyFunction Aliaslive*。

可參考屬性：*<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 Version926eeb5ff1*。

可參考屬性：*<function-LogicalId>.Version*

如需 AutoPublishAlias 屬性的詳細資訊，請參閱 [AWS::Serverless::Function 的屬性一節](#)。

已指定 DeploymentPreference 屬性

AWS::Serverless::Function 指定的 DeploymentPreference 屬性時，AWS SAM 會產生下列 CloudFormation 資源：AWS::CodeDeploy::Application 和 AWS::CodeDeploy::DeploymentGroup。此外，如果未指定 DeploymentPreference 物件的 Role 屬性，AWS SAM 也會產生 AWS::IAM::Role CloudFormation 資源。

AWS::CodeDeploy::Application

LogicalId: ServerlessDeploymentApplication

可參考屬性：N/A（您必須使用 LogicalId 來參考此 CloudFormation 資源）

AWS::CodeDeploy::DeploymentGroup

LogicalId: *<function-LogicalId>*DeploymentGroup

可參考屬性：N/A（您必須使用 LogicalId 來參考此 CloudFormation 資源）

AWS::IAM::Role

LogicalId: CodeDeployServiceRole

可參考屬性：N/A（您必須使用 LogicalId 來參考此 CloudFormation 資源）

已指定 FunctionUrlConfig 屬性

指定 FunctionUrlConfig 屬性時，會根據 AWS SAM 產生不同的 CloudFormation 資源 AuthType。

指定 AuthType: NONE 時，AWS SAM 會產生下列 CloudFormation 資源：

AWS::Lambda::Permission（叫用存取）

LogicalId: *<function-LogicalId>*URLInvokeAllowPublicAccess

可參考屬性：N/A（您必須使用 LogicalId 來參考此 CloudFormation 資源）

AWS::Lambda::Permission（公開存取）

LogicalId: *<function-LogicalId>*UrlPublicPermissions

可參考屬性：N/A（您必須使用 LogicalId 來參考此 CloudFormation 資源）

AWS::Lambda::Url

LogicalId: *<function-LogicalId>Url*

可參考屬性：N/A（您必須使用 LogicalId 來參考此 CloudFormation 資源）

指定 AuthType：AWS_IAM 時，只會 AWS SAM 產生：

AWS::Lambda::Url

LogicalId: *<function-LogicalId>Url*

可參考屬性：N/A（您必須使用 LogicalId 來參考此 CloudFormation 資源）

如需 FunctionUrlConfig 屬性的詳細資訊，請參閱 [FunctionUrlConfig](#)。

事件來源

下列案例會根據事件來源產生 CloudFormation 資源：

已指定 Api 事件來源

當的 Event 屬性 `AWS::Serverless::Function` 設定為 `Api`，但未指定 `RestApiId` 屬性時，AWS SAM 會產生 `AWS::ApiGateway::RestApi` CloudFormation 資源。

AWS::ApiGateway::RestApi

LogicalId: `ServerlessRestApi`

可參考屬性：N/A（您必須使用 LogicalId 來參考此 CloudFormation 資源）

已指定 HttpApi 事件來源

當的 Event 屬性 `AWS::Serverless::Function` 設定為 `HttpApi`，但未指定 `ApiId` 屬性時，AWS SAM 會產生 `AWS::ApiGatewayV2::Api` CloudFormation 資源。

AWS::ApiGatewayV2::Api

LogicalId: `ServerlessHttpApi`

可參考屬性：N/A（您必須使用 LogicalId 來參考此 CloudFormation 資源）

已指定串流事件來源

當的 Event 屬性 `AWS::Serverless::Function` 設定為其中一個串流類型時，AWS SAM 會產生 `AWS::Lambda::EventSourceMapping` CloudFormation 資源。這適用於下列類型：DynamoDB、Kinesis、MSK、MQ 和 SQS。

AWS::Lambda::EventSourceMapping

LogicalId: <function-LogicalId><event-LogicalId>

可參考屬性：N/A（您必須使用 LogicalId 來參考此 CloudFormation 資源）

已指定事件橋接（或事件匯流排）事件來源

當的 Event 屬性 `AWS::Serverless::Function` 設定為其中一個事件橋接（或事件匯流排）類型時，AWS SAM 會產生 `AWS::Events::Rule` CloudFormation 資源。這適用於下列類型：EventBridgeRule、Schedule 和 CloudWatchEvents。

AWS::Events::Rule

LogicalId: <function-LogicalId><event-LogicalId>

可參考屬性：N/A（您必須使用 LogicalId 來參考此 CloudFormation 資源）

已指定 IoTRule 事件來源

當的 Event 屬性 `AWS::Serverless::Function` 設定為 IoTRule 時，AWS SAM 會產生 `AWS::IoT::TopicRule` CloudFormation 資源。

AWS::IoT::TopicRule

LogicalId: <function-LogicalId><event-LogicalId>

可參考屬性：N/A（您必須使用 LogicalId 來參考此 CloudFormation 資源）

事件組態

下列案例會根據事件組態產生 CloudFormation 資源：

針對 Amazon SNS 事件指定 OnSuccess (或 OnFailure) 屬性 Amazon SNS

當 `EventInvokeConfigAWS::Serverless::Function` 指定之 屬性的 `DestinationConfig` 屬性 OnSuccess (或 OnFailure) 屬性，且目的地類型為 `SNS` 但未指定目的地 ARN 時，AWS SAM 會產生下列 CloudFormation 資源：`AWS::Lambda::EventInvokeConfig` 和 `AWS::SNS::Topic`。

AWS::Lambda::EventInvokeConfig

LogicalId: *<function-LogicalId>*EventInvokeConfig

可參考屬性：N/A (您必須使用 *LogicalId* 來參考此 CloudFormation 資源)

AWS::SNS::Topic

LogicalId: *<function-LogicalId>*OnSuccessTopic (或 *<function-LogicalId>*OnFailureTopic)

可參考屬性：*<function-LogicalId>*.DestinationTopic

如果同時為 Amazon SNS 事件 OnFailure 指定 OnSuccess 和 `OnFailure`，若要區分產生的資源，您必須使用 *LogicalId*。

針對 Amazon SQS 事件指定 OnSuccess (或 OnFailure) 屬性 Amazon SQS

當 `AWS::Serverless::Function` 指定之 `DestinationConfig` 屬性的 `EventInvokeConfig` 屬性 OnSuccess (或 OnFailure) 屬性，且目的地類型為 `SQS` 但未指定目的地 ARN 時，AWS SAM 會產生下列 CloudFormation 資源：`AWS::Lambda::EventInvokeConfig` 和 `AWS::SQS::Queue`。

AWS::Lambda::EventInvokeConfig

LogicalId: *<function-LogicalId>*EventInvokeConfig

可參考屬性：N/A (您必須使用 *LogicalId* 來參考此 CloudFormation 資源)

AWS::SQS::Queue

LogicalId: *<function-LogicalId>*OnSuccessQueue (或 *<function-LogicalId>*OnFailureQueue)

可參考屬性：*<function-LogicalId>*.DestinationQueue

如果同時為 Amazon SQS 事件 `OnFailure` 指定 `OnSuccess` 和 `LogicalId`，若要區分產生的資源，您必須使用 `LogicalId`。

CloudFormation 指定 `AWS::Serverless::GraphQLApi` 時產生的資源

當您在 AWS Serverless Application Model (AWS SAM) 範本中指定 `AWS::Serverless::GraphQLApi` 資源時，AWS SAM 一律會建立下列基本 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` 來參考此 CloudFormation 資源）

`AWS::AppSync::FunctionConfiguration`

LogicalId: `<graphqlapi-LogicalId><function-RelativeId>`

可參考屬性：N/A（您必須使用 `LogicalId` 來參考此 CloudFormation 資源）

`AWS::AppSync::GraphQLApi`

LogicalId: `<graphqlapi-LogicalId>`

可參考屬性：N/A（您必須使用 `LogicalId` 來參考此 CloudFormation 資源）

`AWS::AppSync::GraphQLSchema`

LogicalId: `<graphqlapi-LogicalId>Schema`

可參考屬性：N/A（您必須使用 `LogicalId` 來參考此 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` 來參考此 CloudFormation 資源）

除了這些 CloudFormation 資源之外，`AWS::Serverless::GraphQLApi` 指定時，AWS SAM 也可能會產生下列 CloudFormation 資源。

AWS::AppSync::ApiCache

LogicalId: *<graphqlapi-LogicalId>*ApiCache

可參考屬性：N/A（您必須使用 LogicalId 來參考此 CloudFormation 資源）

AWS::AppSync::ApiKey

LogicalId: *<graphqlapi-LogicalId><apikey-RelativeId>*

可參考屬性：N/A（您必須使用 LogicalId 來參考此 CloudFormation 資源）

AWS::AppSync::DomainName

LogicalId: *<graphqlapi-LogicalId>*DomainName

可參考屬性：N/A（您必須使用 LogicalId 來參考此 CloudFormation 資源）

AWS::AppSync::DomainNameApiAssociation

LogicalId: *<graphqlapi-LogicalId>*DomainNameApiAssociation

可參考屬性：N/A（您必須使用 LogicalId 來參考此 CloudFormation 資源）

AWS SAM 也可以使用 `AWS::Serverless::Connector` 資源來佈建許可。如需詳細資訊，請參閱 [CloudFormation 當您指定 時產生的資源 AWS::Serverless::Connector](#)。

CloudFormation 指定 `AWS::Serverless::HttpApi` 時產生的資源

指定 `AWS::Serverless::HttpApi` 時，AWS Serverless Application Model (AWS SAM) 會產生 `AWS::ApiGatewayV2::Api` 基本 CloudFormation 資源。

AWS::ApiGatewayV2::Api

LogicalId: *<httpapi-LogicalId>*

可參考屬性：N/A（您必須使用 LogicalId 來參考此 CloudFormation 資源）

除了此 CloudFormation 資源之外，`AWS::Serverless::HttpApi` 指定時，AWS SAM 也會為下列案例產生 CloudFormation 資源：

案例

- [已指定 StageName 屬性](#)

- [未指定 StageName 屬性](#)
- [已指定 DomainName 屬性](#)

已指定 StageName 屬性

AWS::Serverless::HttpApi 指定的 StageName 屬性時，AWS SAM 會產生 AWS::ApiGatewayV2::Stage CloudFormation 資源。

AWS::ApiGatewayV2::Stage

LogicalId: *<httpapi-LogicalId><stage-name>*Stage

<stage-name> 是 StageName 屬性設定為的字串。例如，如果您將 StageName 設定為 Gamma，則 LogicalId 為：*MyHttpApiGamma* Stage。

可參考屬性：*<httpapi-LogicalId>*.Stage

未指定 StageName 屬性

AWS::Serverless::HttpApi 未指定的 StageName 屬性時，AWS SAM 會產生 AWS::ApiGatewayV2::Stage CloudFormation 資源。

AWS::ApiGatewayV2::Stage

LogicalId: *<httpapi-LogicalId>*ApiGatewayDefaultStage

可參考屬性：*<httpapi-LogicalId>*.Stage

已指定 DomainName 屬性

AWS::Serverless::HttpApi 指定 DomainName 的 Domain 屬性時，AWS SAM 會產生 AWS::ApiGatewayV2::DomainName CloudFormation 資源。

AWS::ApiGatewayV2::DomainName

LogicalId: ApiGatewayDomainNameV2*<sha>*

<sha> 是建立堆疊時產生的唯一雜湊值。例如，ApiGatewayDomainNameV2*926eeb5ff1*。

可參考屬性：*<httpapi-LogicalId>*.DomainName

CloudFormation 指定 AWS::Serverless::MicrovmImage 時產生的資源

當您指定 時AWS::Serverless::MicrovmImage，AWS Serverless Application Model (AWS SAM) 會產生AWS::Lambda::MicrovmImage基本 CloudFormation 資源。

AWS::Lambda::MicrovmImage

LogicalId: *<microvmimage-LogicalId>*

可參考屬性：N/A（您必須使用 LogicalId來參考此 CloudFormation 資源）

除了此 CloudFormation 資源之外，AWS::Serverless::MicrovmImage指定 時，AWS SAM 也會為下列案例產生 CloudFormation 資源：

案例

- [未指定 BuildRoleArn 屬性](#)

未指定 BuildRoleArn 屬性

當您未指定的 BuildRoleArn 屬性時AWS::Serverless::MicrovmImage，AWS SAM 會產生具有授予下列許可之內嵌政策AWS::IAM::Role CloudFormation 的資源：

- s3:GetObject — 從 範圍限定為 Amazon S3 儲存貯體 CodeUri
- logs:CreateLogGroup、logs:CreateLogStream、logs:PutLogEvents — 用於建置日誌

角色信任政策允許lambda.amazonaws.com服務主體擔任該角色。

AWS::IAM::Role

LogicalId: *<microvmimage-LogicalId>*BuildRole

CloudFormation 指定 AWS::Serverless::NetworkConnector 時產生的資源

當您指定 時AWS::Serverless::NetworkConnector，AWS Serverless Application Model (AWS SAM) 會產生AWS::Lambda::NetworkConnector基本 CloudFormation 資源。

AWS::Lambda::NetworkConnector

LogicalId: *<networkconnector-LogicalId>*

可參考屬性：N/A（您必須使用 `LogicalId` 來參考此 CloudFormation 資源）

除了此 CloudFormation 資源之外，`AWS::Serverless::NetworkConnector` 指定時，AWS SAM 也會為下列案例產生 CloudFormation 資源：

案例

- [未指定 `OperatorRole` 屬性](#)

未指定 `OperatorRole` 屬性

當您未指定的 `OperatorRole` 屬性時 `AWS::Serverless::NetworkConnector`，AWS SAM 會產生已連接 `AWSLambdaNetworkConnectorOperatorPolicy` AWS 受管政策 `AWS::IAM::Role` CloudFormation 的資源。

角色信任政策允許 `lambda.amazonaws.com` 服務主體擔任該角色。

`AWS::IAM::Role`

LogicalId: `<networkconnector-LogicalId>OperatorRole`

CloudFormation 指定 `AWS::Serverless::WebFunction` 時產生的資源

當您指定時 `AWS::Serverless::WebFunction`，AWS Serverless Application Model (AWS SAM) 會產生下列基本 CloudFormation 資源。

`AWS::Lambda::WebFunction`

LogicalId: `<webfunction-LogicalId>`

可參考屬性：N/A（您必須使用 `LogicalId` 來參考此 CloudFormation 資源）

`AWS::Lambda::WebFunctionRevision`

LogicalId: `<webfunction-LogicalId>Revision`

可參考屬性：N/A（您必須使用 `LogicalId` 來參考此 CloudFormation 資源）

`AWS::Lambda::WebFunctionEndpoint`

LogicalId: `<webfunction-LogicalId>Endpoint`

可參考屬性：N/A（您必須使用 `LogicalId` 來參考此 CloudFormation 資源）

這三個資源都使用相同的函數名稱。如果您不指定 `FunctionName` 屬性，AWS SAM 會產生格式的名稱 `<webfunction-LogicalId>-<stack-id-prefix>`，其中 `stack-id-prefix` 是堆疊 ID 唯一識別符的前 8 個字元。如果需要，請 AWS SAM 縮短邏輯 ID，以便名稱最多為 64 個字元。

除了這些 CloudFormation 資源之外，`AWS::Serverless::WebFunction` 指定時，AWS SAM 也會為下列案例產生 CloudFormation 資源：

案例

- [未指定 `ExecutionRoleArn` 屬性](#)

未指定 `ExecutionRoleArn` 屬性

當您未指定的 `ExecutionRoleArn` 屬性時 `AWS::Serverless::WebFunction`，AWS SAM 會產生已連接 `AWSLambdaBasicExecutionRole` AWS 受管政策 `AWS::IAM::Role` CloudFormation 的資源。

角色信任政策允許 `lambda.amazonaws.com` 服務主體擔任該角色。

`AWS::IAM::Role`

LogicalId: `<webfunction-LogicalId>Role`

CloudFormation 指定 `AWS::Serverless::WebSocketApi` 時產生的資源

指定 `AWS::Serverless::WebSocketApi` 時，AWS Serverless Application Model (AWS SAM) 會產生 `AWS::ApiGatewayV2::Api` 基本 CloudFormation 資源。此外，它也會一律產生 `AWS::ApiGatewayV2::Stage` 資源，而針對 `Routes` 屬性中定義的每個路由，AWS SAM 會產生 `AWS::ApiGatewayV2::Route`、`AWS::ApiGatewayV2::Integration` 和 `AWS::Lambda::Permission` 資源。

`AWS::ApiGatewayV2::Api`

LogicalId: `<websocketapi-LogicalId>`

可參考屬性：N/A（您必須使用 `LogicalId` 來參考此 CloudFormation 資源）

`AWS::ApiGatewayV2::Stage`

LogicalId: `<websocketapi-LogicalId><stage-name>Stage`

`<stage-name>` 是 StageName 屬性設定為的字串。如果 StageName 未指定，則預設值為 default。例如，如果您將 StageName 設定為 prod，則 LogicalId 為 `MyWebSocketApiprodStage`。

可參考屬性：`<websocketapi-LogicalId>.Stage`

AWS::ApiGatewayV2::Route

LogicalId: `<websocketapi-LogicalId><normalized-route-key>`Route

`<normalized-route-key>` 是已移除特殊字元的路由金鑰。例如，對於路由金鑰 `$connect`，LogicalId 是 `MyWebSocketApiconnectRoute`。

可參考屬性：N/A（您必須使用 LogicalId 來參考此 CloudFormation 資源）

AWS::ApiGatewayV2::Integration

LogicalId: `<websocketapi-LogicalId><normalized-route-key>`Integration

可參考屬性：N/A（您必須使用 LogicalId 來參考此 CloudFormation 資源）

AWS::Lambda::Permission

LogicalId: `<websocketapi-LogicalId><normalized-route-key>`Permission

可參考屬性：N/A（您必須使用 LogicalId 來參考此 CloudFormation 資源）

除了這些 CloudFormation 資源之外，AWS::Serverless::WebSocketApi 指定時，AWS SAM 也會為下列案例產生 CloudFormation 資源：

案例

- [Auth 屬性指定為 AuthType 設定為 CUSTOM](#)
- [已指定 DomainName 屬性](#)
- [已指定 BasePath 屬性](#)
- [已指定 Route53 屬性](#)

Auth 屬性指定為 AuthType 設定為 CUSTOM

將的 Auth 屬性指定 AWS::Serverless::WebSocketApi 為 AuthType 時 CUSTOM，AWS SAM 會產生 AWS::ApiGatewayV2::Authorizer CloudFormation 資源。如果 InvokeRole 未指定，AWS SAM 也會產生 AWS::Lambda::Permission 資源，以允許 API Gateway 叫用授權方函數。

AWS::ApiGatewayV2::Authorizer

LogicalId: *<websocketapi-LogicalId>*Authorizer

可參考屬性：N/A（您必須使用 *LogicalId* 來參考此 CloudFormation 資源）

AWS::Lambda::Permission (InvokeRole 未指定時)

LogicalId: *<websocketapi-LogicalId>*AuthorizerPermission

可參考屬性：N/A（您必須使用 *LogicalId* 來參考此 CloudFormation 資源）

已指定 DomainName 屬性

AWS::Serverless::WebSocketApi 指定 *DomainName* 的 *Domain* 屬性時，AWS SAM 會產生 AWS::ApiGatewayV2::DomainName CloudFormation 資源。

AWS::ApiGatewayV2::DomainName

LogicalId: ApiGatewayDomainNameV2*<sha>*

<sha> 是建立堆疊時產生的唯一雜湊值。例如：ApiGatewayDomainNameV2*926eeb5ff1*。

可參考屬性：N/A（您必須使用 *LogicalId* 來參考此 CloudFormation 資源）

已指定 BasePath 屬性

AWS::Serverless::WebSocketApi 指定 *BasePath* 的 *Domain* 屬性時，AWS SAM 會產生 AWS::ApiGatewayV2::ApiMapping CloudFormation 資源，每個指定的基礎路徑各一個。

AWS::ApiGatewayV2::ApiMapping

LogicalId: *<websocketapi-LogicalId><basepath>*ApiMapping

<basepath> 是基本路徑值。例如，如果您將 *v1* 指定為基本路徑，則 *LogicalId* 為 *MyWebSocketApi**v1*ApiMapping。

可參考屬性：N/A（您必須使用 *LogicalId* 來參考此 CloudFormation 資源）

已指定 Route53 屬性

Domain `AWS::Serverless::WebSocketApi` 指定 屬性的 `Route53` 屬性時，AWS SAM 會產生 `AWS::Route53::RecordSetGroup` CloudFormation 資源。

AWS::Route53::RecordSetGroup

LogicalId: `RecordSetGroup<sha>`

`<sha>` 是建立堆疊時產生的唯一雜湊值。例如：`RecordSetGroup926eeb5ff1`。

可參考屬性：N/A（您必須使用 `LogicalId` 來參考此 CloudFormation 資源）

CloudFormation 指定 `AWS::Serverless::LayerVersion` 時產生的資源

指定 `AWS::Serverless::LayerVersion` 時，AWS Serverless Application Model (AWS SAM) 會產生 `AWS::Lambda::LayerVersion` 基本 CloudFormation 資源。

AWS::Lambda::LayerVersion

LogicalId: `<layerversion-LogicalId>`

可參考屬性：N/A（您必須使用 `LogicalId` 來參考此 CloudFormation 資源）

CloudFormation 指定 `AWS::Serverless::SimpleTable` 時產生的資源

指定 `AWS::Serverless::SimpleTable` 時，AWS Serverless Application Model (AWS SAM) 會產生 `AWS::DynamoDB::Table` 基本 CloudFormation 資源。

AWS::DynamoDB::Table

LogicalId: `<simpletable-LogicalId>`

可參考屬性：N/A（您必須使用 `LogicalId` 來參考此 CloudFormation 資源）

CloudFormation 指定 `AWS::Serverless::StateMachine` 時產生的資源

指定 `AWS::Serverless::StateMachine` 時，AWS Serverless Application Model (AWS SAM) 會產生 `AWS::StepFunctions::StateMachine` 基本 CloudFormation 資源。

AWS::StepFunctions::StateMachine

LogicalId: *<statemachine-LogicalId>*

可參考屬性：N/A（您必須使用 LogicalId 來參考此 CloudFormation 資源）

除了此 CloudFormation 資源之外，AWS::Serverless::StateMachine 指定時，AWS SAM 也會為下列案例產生 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（或事件匯流排）事件來源](#)

未指定角色屬性

AWS::Serverless::StateMachine 未指定的 Role 屬性時，AWS SAM 會產生 AWS::IAM::Role CloudFormation 資源。

AWS::IAM::Role

LogicalId: *<statemachine-LogicalId>*Role

可參考屬性：N/A（您必須使用 LogicalId 來參考此 CloudFormation 資源）

已指定 API 事件來源

當的 Event 屬性 AWS::Serverless::StateMachine 設定為 Api，但未指定 RestApiId 屬性時，AWS SAM 會產生 AWS::ApiGateway::RestApi CloudFormation 資源。

AWS::ApiGateway::RestApi

LogicalId: ServerlessRestApi

可參考屬性：N/A（您必須使用 LogicalId 來參考此 CloudFormation 資源）

已指定事件橋接（或事件匯流排）事件來源

當的 Event 屬性 `AWS::Serverless::StateMachine` 設定為其中一個事件橋接（或事件匯流排）類型時，AWS SAM 會產生 `AWS::Events::Rule` CloudFormation 資源。這適用於下列類型：`EventBridgeRule`、`Schedule` 和 `CloudWatchEvents`。

`AWS::Events::Rule`

`LogicalId`: `<statemachine-LogicalId><event-LogicalId>`

可參考屬性：N/A（您必須使用 `LogicalId` 來參考此 CloudFormation 資源）

支援的資源屬性 AWS SAM

資源屬性是您可以新增至 AWS SAM 和 CloudFormation 資源的屬性，以控制其他行為和關係。如需資源屬性的詳細資訊，請參閱 AWS CloudFormation 《使用者指南》中的 [資源屬性參考](#)。

AWS SAM 支援由定義的資源屬性子集 CloudFormation。在支援的資源屬性中，有些只會複製到對應 CloudFormation 資源的基礎產生 AWS SAM 資源，有些則會複製到對應 CloudFormation 資源產生的所有產生 AWS SAM 資源。如需從對應 CloudFormation 資源產生 AWS SAM 之資源的詳細資訊，請參閱 [為產生 CloudFormation 資源 AWS SAM](#)。

下表摘要說明支援的資源屬性 AWS SAM，但需遵守下列 [例外狀況](#) 內容。

資源屬性	目的地產生的資源 (s)
DependsOn 中繼資料 ^{1、2}	僅限基本 CloudFormation 產生的資源。如需 AWS SAM 資源與基礎 CloudFormation 資源之間的映射資訊，請參閱 產生的 CloudFormation 資源案例 。
條件 DeletionPolicy UpdateReplacePolicy	來自對應 CloudFormation AWS SAM 資源的所有產生資源。如需所產生 CloudFormation 資源案例的資訊，請參閱 產生的 CloudFormation 資源案例 。

備註：

1. 如需搭配 Metadata 資源類型使用 `AWS::Serverless::Function` 資源屬性的詳細資訊，請參閱 [在中使用自訂執行時間建置 Lambda 事件函數 AWS SAM](#)。
2. 如需搭配 Metadata 資源類型使用 `AWS::Serverless::LayerVersion` 資源屬性的詳細資訊，請參閱 [在中建置 Lambda 層 AWS SAM](#)。

例外狀況

上述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 `DeletionPolicy` 會為產生的 CloudFormation 資源 `RetentionPolicy` 設定。其優先順序高於 `DeletionPolicy` 本身。如果兩者都未設定，則預設為 `DeletionPolicy Retain`。
- 對於 `AWS::Lambda::Version`，如果 `DeletionPolicy` 未指定，則預設值為 `Retain`。
- 對於 `DeploymentPreferences` 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 CloudFormation 資源：
 - `AWS::CodeDeploy::Application`
 - `AWS::CodeDeploy::DeploymentGroup`
 - 為此案例建立 `CodeDeployServiceRole` 的 `AWS::IAM::Role` 名稱
- 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，則對於產生的 `AWS::ApiGateway::RestApi` 資源，會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`

的 API Gateway 擴充功能 AWS SAM

API Gateway 擴充功能專為設計 AWS，提供額外的自訂功能，用於設計和管理 APIs。API Gateway 擴充功能是 OpenAPI 規格的擴充功能，可支援 AWS 特定授權和 API Gateway 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 擴充功能的子集。若要查看 支援哪些 API Gateway 延伸 AWS SAM，請參閱下表。

API Gateway 延伸模組	支援者 AWS SAM
x-amazon-apigateway-any-method 物件	是
x-amazon-apigateway-api-key-source 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-media-types 屬性	是
x-amazon-apigateway-documentation 物件	否
x-amazon-apigateway-endpoint-configuration 物件	否
x-amazon-apigateway-gateway-responses 物件	是
x-amazon-apigateway-gateway-responses.gatewayResponse 物件	是
x-amazon-apigateway-gateway-responses.responseParameters 物件	是
x-amazon-apigateway-gateway-responses.responseTemplates 物件	是
x-amazon-apigateway-integration 物件	是
x-amazon-apigateway-integration.requestTemplates 物件	是
x-amazon-apigateway-integration.requestParameters 物件	否

x-amazon-apigateway-integration.responses 物件	是
x-amazon-apigateway-integration.response 物件	是
x-amazon-apigateway-integration.responseTemplates 物件	是
x-amazon-apigateway-integration.responseParameters 物件	是
x-amazon-apigateway-request-validator 屬性	否
x-amazon-apigateway-request-validators 物件	否
x-amazon-apigateway-request-validators.requestValidator 物件	否

的內部 函數 AWS SAM

內部函數是內建函數，可讓您將值指派給只能在執行期使用的屬性。AWS SAM 對某些內部函數屬性的支援有限，因此無法解析某些內部函數。因此，我們建議您新增AWS::LanguageExtensions轉換來解決此問題。AWS::LanguageExtensions 是由託管的巨集 CloudFormation，可讓您使用預設不包含在 中的內部函數和其他功能 CloudFormation。

Transform:

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

Note

如果您在 CodeUri 屬性中使用內部函數，AWS SAM 將無法正確剖析值。請考慮改用AWS::LanguageExtensions轉換。

如需詳細資訊，請參閱 [AWS::Serverless::Function 的屬性一節](#)。

如需 AWS SAM CLI如何處理AWS::LanguageExtensions轉換的詳細資訊，包括在 中使用內部函數CodeUri和其他可封裝屬性，請參閱 [CloudFormation 語言延伸支援](#)。

如需內部函數的詳細資訊，請參閱AWS CloudFormation 《使用者指南》中的[內部函數參考](#)。

CloudFormation 語言延伸支援

AWS SAM CLI 支援使用 `AWS::LanguageExtensions` 轉換的範本，包括 `Fn::ForEach`、`Fn::ToJsonString`、`Fn::Length` 和 `Fn::FindInMap` 與 `DefaultValue`。如需這些建構模組的完整參考資訊，請參閱 AWS CloudFormation 《使用者指南》中的 [AWS::LanguageExtensions 轉換](#)。

若要在 AWS SAM 範本中使用語言延伸，請在 之前的 Transform 區段 `AWS::LanguageExtensions` 中列出 `AWS::Serverless-2016-10-31`：

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

當在 AWS SAM CLI 範本的 Transform 區段 `AWS::LanguageExtensions` 中偵測到，且您已選擇加入本機處理時，它會在執行 AWS SAM 轉換之前於本機擴展語言延伸建構。這可讓 `sam build`、`sam package`、`sam deploy`、`sam sync`、`sam validate`、`sam local start-api`、和 `sam local invokesam local start-lambda` 使用這些建構的範本。

本機處理預設為關閉。關閉時，AWS SAM CLI 會傳遞範本，並在部署時間 CloudFormation 處理 `AWS::LanguageExtensions` 轉換伺服器端。如需啟用方法，請參閱 [啟用語言延伸模組](#)。

啟用時，擴展會分兩個階段進行：

1. 階段 1（語言延伸） — `Fn::ForEach` 迴圈會展開，內部函數會盡可能解析，且範本會轉換為標準 CloudFormation。
2. 階段 2 (AWS SAM 轉換) — 擴展的範本會照常由 AWS SAM Translator 處理。

原始範本 (`Fn::ForEach` 完整) 會保留供 CloudFormation 部署使用，因為會 CloudFormation 處理 `AWS::LanguageExtensions` 轉換伺服器端。

啟用語言延伸模組

的本機處理 `AWS::LanguageExtensions` 是依命令選擇加入。有三種同等的啟用方法，依優先順序列出：

1. CLI flag — `--language-extensions` 傳遞單一調用：

```
sam build --language-extensions
```

```
sam package --language-extensions ...  
sam deploy --language-extensions ...
```

--no-language-extensions 明確停用，覆寫以下所述的 `samconfig.toml` 和 環境變數。

2. `samconfig.toml` — 保留每個專案的選擇：

```
[default.build.parameters]  
language_extensions = true  
  
[default.package.parameters]  
language_extensions = true  
  
[default.deploy.parameters]  
language_extensions = true  
  
[default.sync.parameters]  
language_extensions = true  
  
[default.local_invoke.parameters]  
language_extensions = true  
  
[default.local_start_api.parameters]  
language_extensions = true  
  
[default.local_start_lambda.parameters]  
language_extensions = true  
  
[default.validate.parameters]  
language_extensions = true
```

`samconfig.toml` 項目會載入為 命令的預設值，因此會像通過旗標一樣生效，因此會勝過環境變數。

3. 環境變數 — `SAM_CLI_ENABLE_LANGUAGE_EXTENSIONS=1` 設定為針對目前的 Shell 啟用：

```
export SAM_CLI_ENABLE_LANGUAGE_EXTENSIONS=1  
sam build  
sam local invoke MyFunction
```

真實值（不區分大小寫）為 1、true 和 yes。任何其他項目，包括空白字串，都會視為關閉。只有在 CLI 旗標或 都未 `samconfig.toml` 設定值時，才會參考環境變數。

Important

每個命令都需要自己的啟用。傳遞 `--language-extensions` 至 `sam build` 不會傳播至稍後 `sam local invoke`。本機處理會依命令叫用決定。使用環境變數或 `samconfig.toml` 項目跨命令啟用，而無需重複旗標。

Fn::ForEach

Fn::ForEach 從單一範本定義產生多個資源、條件或輸出：

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

Parameters:
  ServiceNames:
    Type: CommaDelimitedList
    Default: "Users,Orders,Products"

Resources:
  Fn::ForEach::Services:
    - Name
    - !Ref ServiceNames
    - ${Name}Function:
        Type: AWS::Serverless::Function
        Properties:
          Handler: index.handler
          Runtime: python3.12
          CodeUri: ./services/${Name}
```

執行 會將此項目 `sam build` 擴展至 `OrdersFunction`、`UsersFunction` 和 `ProductsFunction`，每個項目都是從其個別來源目錄建置。

動態成品屬性

當可封裝屬性使用迴圈變數（例如 `./services/${Name}`）時，AWS SAM CLI 會產生區段 CloudFormation Mappings，將每個集合值映射至其 Amazon S3URI。內 `Fn::ForEach` 文會重寫為使用 `Fn::FindInMap`，以便在部署時 CloudFormation 解析正確的成品。

辨識的成品屬性與今天 `sam package` 重新寫入的成品屬性相同。這包括：

辨識的動態成品屬性 AWS SAM CLI

Resource Type (資源類型)	屬性
AWS::Serverless::Function	CodeUri, ImageUri
AWS::Serverless::LayerVersion	ContentUri
AWS::Serverless::Api	DefinitionUri
AWS::Serverless::HttpApi	DefinitionUri
AWS::Serverless::StateMachine	DefinitionUri
AWS::Serverless::GraphQLApi	SchemaUri , CodeUri
AWS::Serverless::Application	Location
AWS::Lambda::Function	Code, Code.ImageUri
AWS::Lambda::LayerVersion	Content
AWS::ApiGateway::RestApi	BodyS3Location
AWS::ApiGatewayV2::Api	BodyS3Location
AWS::AppSync::GraphQLSchema	DefinitionS3Location
AWS::AppSync::Resolver	RequestMappingTemplateS3Location , ResponseMappingTemplateS3Location , CodeS3Location
AWS::AppSync::FunctionConfiguration	RequestMappingTemplateS3Location , ResponseMappingTemplateS3Location , CodeS3Location
AWS::StepFunctions::StateMachine	DefinitionS3Location

Resource Type (資源類型)	屬性
AWS::ElasticBeanstalk::ApplicationVersion	SourceBundle
AWS::Glue::Job	Command.ScriptLocation
AWS::CloudFormation::Stack	TemplateURL
AWS::CloudFormation::StackSet	TemplateURL
AWS::CloudFormation::ModuleVersion	ModulePackage
AWS::CloudFormation::ResourceVersion	SchemaHandlerPackage

對於虛線屬性（例如 `Command.ScriptLocation`，在 `AWS::Glue::Job` 或在 `Code.ImageUri` 上 `AWS::Lambda::Function`），值會在資源上的巢狀位置讀取和寫入。產生的映射名稱只會使用屬性路徑的分葉區段。

當屬性經過循環調整時，映射名稱為 `SAM<LeafProperty><LoopName>`（例如，`SAMCodeUriServices` 或 `SAMScriptLocationJobs`）。

Important

客戶編寫的映射不應以這些字 `SAM*` 首開頭，它們會保留給 AWS SAM CLI。請參閱 [限制](#)。

例如，在之後 `sam package`：

```
Mappings:
  SAMCodeUriServices:
    Users:
      CodeUri: s3://my-bucket/abc123
    Orders:
      CodeUri: s3://my-bucket/def456
    Products:
```

```
CodeUri: s3://my-bucket/ghi789
```

```
Resources:
```

```
  Fn::ForEach::Services:
```

- Name
- !Ref ServiceNames
- \${Name}Function:
 - Type: AWS::Serverless::Function
 - Properties:
 - Handler: index.handler
 - Runtime: python3.12
 - CodeUri: !FindInMap [SAMCodeUriServices, !Ref Name, CodeUri]

每個 ForEach 內文的多個資源

單一Fn::ForEach內文每次反覆運算可以發出多個資源。每個資源都會針對每個集合值產生：

```
Resources:
```

```
  Fn::ForEach::Tables:
```

- TableName
- [Users, Orders, Products]
- \${TableName}Table:
 - Type: AWS::DynamoDB::Table
 - Properties:
 - TableName: !Sub "\${AWS::StackName}-\${TableName}"
 - # ...

```
  ${TableName}StreamProcessor:
```

- Type: AWS::Serverless::Function
- Properties:
 - CodeUri: stream-processors/\${TableName}/
 - Events:
 - DDBStream:
 - Type: DynamoDB
 - Properties:
 - Stream: !GetAtt
 - !Sub "\${TableName}Table"
 - StreamArn

映射名稱碰撞解析度

當同一Fn::ForEach內文中的兩個資源宣告相同的動態成品屬性（例如，Api和StateMachine使用DefinitionUri）時，AWS SAM CLI會附加從資源邏輯 ID 範本的靜態部分取得的尾碼，以保持映射名稱是唯一的：

具有碰撞尾碼的映射名稱範例

資源範本	屬性	映射名稱
<code>\${Svc}Api</code>	DefinitionUri	<code>SAMDefinitionUriServicesApi</code>
<code>\${Svc}StateMachine</code>	DefinitionUri	<code>SAMDefinitionUriServicesStateMachine</code>

沒有碰撞時，會使用基本名稱（例如 `SAMDefinitionUriServices`）。

參數型集合

當Fn::ForEach集合是參數參考（例如 `!Ref ServiceNames`）且迴圈內文使用動態成品屬性（例如 `CodeUri: ./services/${Name}`）時，AWS SAM CLI需要集合值才能產生中所述的SAM*映射[動態成品屬性](#)。它會在處理範本時解決這些問題，從：

1. `--parameter-overrides` 傳遞至 AWS SAM CLI命令。
2. 範本中的參數Default值。

Important

由於SAM*映射是在封裝時間於中製作，因此每當您變更參數值（例如，新增服務時）時，都必須重新封裝，以便映射包含新值的項目。這僅適用於參數驅動動態成品迴圈時；其他參數覆寫可以照常在部署時間變更。

```
# Package with the values you intend to deploy with
sam package --language-extensions --parameter-overrides
  ServiceNames="Users,Orders,Products"

# Deploy with the same values
```

```
sam deploy --language-extensions --parameter-overrides
ServiceNames="Users,Orders,Products"
```

巢狀堆疊

`Fn::ForEach` 支援巢狀堆疊範本 (`AWS::CloudFormation::Stack`)。會將父堆疊的 `Parameters` 屬性 AWS SAM CLI 傳遞至子範本擴展，以便參考父系提供的參數的子 `Fn::ForEach` 集合正確解析。

```
# parent.yaml
Resources:
  ChildStack:
    Type: AWS::CloudFormation::Stack
    Properties:
      TemplateURL: ./child.yaml
      Parameters:
        ServiceNames: "Users,Orders,Products"
```

巢狀 Fn::ForEach

最多支援 5 個巢狀層級，符合 CloudFormation 限制：

```
Resources:
  Fn::ForEach::Envs:
    - Env
    - [Dev, Staging, Prod]
    - Fn::ForEach::Services:
      - Svc
      - [Users, Orders]
      - ${Env}${Svc}Function:
          Type: AWS::Serverless::Function
          Properties:
            CodeUri: ./services/${Svc}
            Environment:
              Variables:
                STAGE: ${Env}
```

輸出中的 ForEach

`Fn::ForEach` 區塊也會在 `Outputs` 區段中展開，因此您可以為每個集合值發出一個輸出：

```
Outputs:
  Fn::ForEach::FunctionArns:
```

- Name
- [alpha, beta]
- \${Name}FunctionArn:
 - Value: !GetAtt
 - !Sub "\${Name}Function"
 - Arn

條件和 DependsOn

發出的資源Fn::ForEach可以攜帶Condition，DependsOn就像任何其他資源一樣。條件或相依性會複寫到每個產生的資源：

```
Conditions:
  IsProd: !Equals [!Ref Environment, prod]

Resources:
  SharedTable:
    Type: AWS::DynamoDB::Table
    # ...

Fn::ForEach::Functions:
  - Name
  - [api, worker]
  - ${Name}Function:
    Type: AWS::Serverless::Function
    Condition: IsProd
    DependsOn: SharedTable
    Properties:
      Handler: main.handler
      CodeUri: functions/${Name}/
```

&{identifier} 語法

&{identifier} 語法會從取代的值中去除非英數字元，這對於從 IP 地址等值產生有效的邏輯 IDs 非常有用：

```
Fn::ForEach::Hosts:
  - IP
  - ["10.0.0.1", "10.0.0.2"]
  - Host&{IP}:
    Type: AWS::EC2::Instance
    # Expands to Host10001, Host10002
```

支援的內部函數

下列內部函數會在擴充期間於本機解析：

在語言擴展期間，內部函數會在本機解析

函式	說明
<code>Fn::ForEach</code>	迴圈擴展。
<code>Fn::Length</code>	傳回清單元素的計數。
<code>Fn::ToJsonString</code>	將值轉換為JSON字串。
<code>Fn::FindInMap</code>	地圖查詢，包括選用的 <code>DefaultValue</code> 。
<code>Fn::If</code>	條件值選擇。
<code>Fn::Sub</code>	字串替換。
<code>Fn::Join</code>	字串串連。
<code>Fn::Split</code>	字串分割。
<code>Fn::Select</code>	列出元素選擇。
<code>Fn::Base64</code>	Base64 編碼。
<code>Fn::Equals</code> , <code>Fn::And</code> , <code>Fn::Or</code> , <code>Fn::Not</code>	條件評估。
<code>Ref</code>	參數和虛擬參數參考。

需要部署資源 (`Fn::GetAtt`、`Fn::ImportValue`、`Fn::GetAZs`) 的函數會在部署時保留 CloudFormation 供 解析。

驗證錯誤

在 AWS SAM 轉換執行之前，會在本機發現下列範本問題：

語言擴充功能擴展期間引發的驗證錯誤

原因	錯誤訊息
<code>Fn::ForEach</code> 值格式錯誤：不是清單、沒有剛好 3 個元素，或具有非字串迴圈識別符。	<code>Fn::ForEach::<key> layout is incorrect</code> (提高為 <code>InvalidTemplateException</code>)。
超過 5 個層級的 <code>Fn::ForEach</code> 會巢狀化。	<code>Fn::ForEach nesting depth of <N> exceeds the maximum allowed depth of 5. CloudFormation supports up to 5 nested Fn::ForEach loops.</code>
集合會解析為空白清單 (例如，具有的 <code>CommaDelimitedList</code> 參數 <code>Default: ""</code>)。	沒有錯誤 — 無提示地略過迴圈，也不會發出任何資源。
集合 <code>!Ref</code> 中的 <code>會</code> 指向範本中未宣告的參數。	沒有錯誤 – 未解析的參考會保留在範本中。在部署時間，CloudFormation 會解析伺服器端。

限制

- 集合必須在建置/封裝時間解析。使用 `Fn::GetAtt`、`Fn::ImportValue` 或 `SSM/Secrets Manager` 動態參考的 `Fn::ForEach` 集合無法在本機擴展。請 `--parameter-overrides` 改用 參數搭配。
- 動態成品映射會在封裝時間修正。當 `Fn::ForEach` 集合是參數參考且迴圈內文使用動態成品屬性 (例如 `CodeUri: ./services/${Name}`) 時，產生的 SAM* 映射只會包含套件時間解析的參數值項目。如果您在部署時間 `--parameter-overrides` 變更該參數而不重新封裝，則新值不會有映射項目，且部署將會失敗。這不適用於不用於驅動動態成品 的參數 `Fn::ForEach`。
- **DeletionPolicy** 和 **UpdateReplacePolicy** 在擴展期間經過驗證和解決。它們支援 `Ref` 參數，但不支援其他內部函數。
- 巢狀限制。最多 `Fn::ForEach` 可以巢狀 5 個層級，符合 CloudFormation 伺服器端限制。
- 預留映射名稱。以下列任何項目開頭的映射名稱會保留給 AWS SAM CLI，請勿使用這些字首編寫您自己的映射：

- SAMCodeUri、SAMImageUri、SAMContentUri、SAMDefinitionUri、SAMSchemaUri、SAMBody、SAMContent — SAMTemplateURLsam package由 針對動態成品屬性發出。請參閱 [動態成品屬性](#) 資料表。
- SAMLayers — 當 Fn::ForEach產生的函數挑選自動產生的相依性層參考 (Lambda 會將 AWS SAM CLI建置分層為巢狀堆疊) sam build時由 發出。此字首沒有對應的使用者撰寫屬性；它會自動新增。

遙測

當 AWS SAM CLI使用 --language-extensions (或其環境變數對等項目) 叫用命令，且範本宣告AWS::LanguageExtensions轉換時，會發出CFNLanguageExtensions遙測事件。事件會在每次叫用時觸發一次，而且不會傳輸任何範本內容。當本機處理關閉 (預設值) 時，不會觸發事件。

的範例無伺服器應用程式 AWS SAM

本節包含兩個範例應用程式：一個用於處理 DynamoDB 事件，另一個用於處理 Amazon S3 事件。每個範例都會逐步引導您 step-by-step 建立應用程式。此外，這兩者都包含設定事件來源和資源 AWS 的詳細資訊。兩者都從識別開始之前必須執行的操作開始，並遵循初始化、測試、封裝和部署應用程式的步驟。

主題

- [使用 處理 DynamoDB 事件 AWS SAM](#)
- [使用 處理 Amazon S3 事件 AWS SAM](#)

使用 處理 DynamoDB 事件 AWS SAM

透過此範例應用程式，您可以建置在概觀和 Quick Start 指南中學到的內容，並安裝另一個範例應用程式。此應用程式包含由 DynamoDB 資料表事件來源調用的 Lambda 事件函數。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 中完成必要的設定[安裝 AWS SAM CLI](#)。

步驟 1：初始化應用程式

在本節中，您會下載應用程式套件，其中包含 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 AWS SAM CLI 執行下列命令。

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

請注意，在上述命令 gh: 中，會擴展至 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 應用程式所需的兩個 AWS 資源：Lambda 函數和 DynamoDB 資料表。範本也會定義兩個資源之間的對應。
- `read_dynamodb_event/` 目錄 – 包含 DynamoDB 應用程式碼。

步驟 2：在本機測試應用程式

針對本機測試，請使用 AWS SAM CLI 產生範例 DynamoDB 事件並叫用 Lambda 函數：

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

`generate-event` 命令會建立測試事件來源訊息，類似於當所有元件部署到 AWS 雲端時所建立的訊息。此事件來源訊息會輸送至 Lambda 函數 `ReadDynamoDBEvent`。

根據 中的原始碼，確認預期訊息已列印至 主控台 `app.py`。

步驟 3：封裝應用程式

在本機測試應用程式後，您可以使用 AWS SAM CLI 來建立部署套件，用來將應用程式部署到 AWS 雲端。

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. 在命令提示中執行下列 `package` CLI 命令來建立部署套件。

```
sam package \  
  --template-file template.yaml \  
  --output-template-file packaged.yaml \  
  --s3-bucket bucketname
```

在下一個步驟中部署應用程式時 `packaged.yaml`，您可以指定新的範本檔案。

步驟 4：部署應用程式

現在您已建立部署套件，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至 AWS 雲端

- 在 AWS SAM CLI 中，使用 `deploy` CLI 命令來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` 參數允許 AWS CloudFormation 建立 IAM 角色。

CloudFormation 會建立範本中定義的 AWS 資源。您可以在 CloudFormation 主控台中存取這些資源的名稱。

在 AWS 雲端中測試無伺服器應用程式

1. 開啟 DynamoDB 主控台。
2. 將記錄插入您剛建立的資料表。
3. 前往資料表的指標索引標籤，然後選擇檢視所有 CloudWatch 指標。在 CloudWatch 主控台中，選擇日誌來檢視日誌輸出。

後續步驟

若要繼續了解 AWS SAM，請參閱下列資源：

- [Serverless Land](#) – 將無伺服器模式、程式碼範例和學習資源集合在一起的網站，專門用於 AWS SAM 應用程式。

使用 處理 Amazon S3 事件 AWS SAM

透過此範例應用程式，您可以建置在先前範例中學到的內容，並安裝更複雜的應用程式。此應用程式包含由 Amazon S3 物件上傳事件來源調用的 Lambda 事件函數。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在 Amazon S3 中處理物件建立事件。對於上傳至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 資料表中。

Note

使用此範例應用程式，您可以執行與先前範例略有不同的步驟。這是因為此範例需要建立 AWS 資源並設定 IAM 許可，才能在本機測試 Lambda 函數。我們將利用 CloudFormation 來建立資源並為您設定許可。否則，您需要手動執行此操作，才能在本機測試 Lambda 函數。由於此範例更為複雜，因此請務必先熟悉安裝先前的範例應用程式，再執行此範例應用程式。

開始之前

請確定您已在 中完成必要的設定[安裝 AWS SAM CLI](#)。

步驟 1：初始化應用程式

在本節中，您會下載範例應用程式，其中包含 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 AWS SAM CLI 執行下列命令。

```
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 應用程式所需的三個 AWS 資源：Lambda 函數、Amazon S3 儲存貯體和 DynamoDB 資料表。範本也會定義這些資源之間的映射和許可。

- `src/` 目錄 – 包含 Amazon S3 應用程式碼。
- `SampleEvent.json` – 範例事件來源，用於本機測試。

步驟 2：封裝應用程式

在本機測試此應用程式之前，您必須使用 AWS SAM CLI 來建立部署套件，用來將應用程式部署到 AWS 雲端。此部署會建立在本機測試應用程式所需的必要 AWS 資源和許可。

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. 在命令提示中執行下列 package CLI 命令來建立部署套件。

```
sam package \  
  --template-file template.yaml \  
  --output-template-file packaged.yaml \  
  --s3-bucket bucketname
```

在下一個步驟中部署應用程式時 `packaged.yaml`，您可以指定新的範本檔案。

步驟 3：部署應用程式

現在您已建立部署套件，您可以使用它將應用程式部署到 AWS 雲端。然後，您可以在 AWS 雲端中調用應用程式來進行測試。

將無伺服器應用程式部署至 AWS 雲端

- 在 AWS SAM CLI 中，使用 `deploy` 命令來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，`--capabilities` 參數允許 CloudFormation 建立 IAM 角色。

CloudFormation 會建立範本中定義的 AWS 資源。您可以在 CloudFormation 主控台中存取這些資源的名稱。

在 AWS 雲端中測試無伺服器應用程式

1. 將映像上傳至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，請參閱資料表。
3. 確認 DynamoDB 資料表包含新記錄，其中包含 Amazon Rekognition 在上傳影像中找到的文字。

步驟 4：在本機測試應用程式

您必須先擷取由 建立 AWS 的資源名稱，才能在本機測試應用程式 CloudFormation。

- 從中擷取 Amazon S3 金鑰名稱和儲存貯體名稱 CloudFormation。透過取代物件金鑰、儲存貯體名稱和儲存貯體 ARN 的值來修改 `SampleEvent.json` 檔案。
- 擷取 DynamoDB 資料表名稱。此名稱用於下列 `sam local invoke` 命令。

使用 AWS SAM CLI 產生範例 Amazon S3 事件並叫用 Lambda 函數：

```
TABLE_NAME=Table name obtained from CloudFormation console sam local invoke --event  
SampleEvent.json
```

TABLE_NAME= 部分會設定 DynamoDB 資料表名稱。`--event` 參數會指定檔案，其中包含要傳遞給 Lambda 函數的測試事件訊息。

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

若要繼續了解 AWS SAM，請參閱下列資源：

- [Serverless Land](#) – 將無伺服器模式、程式碼範例和學習資源集合在一起的網站，專門用於 AWS SAM 應用程式。

使用 開發無伺服器應用程式 AWS SAM

本節包含有關驗證 AWS SAM 範本和使用相依性建置應用程式的主題。它還包含有關將 AWS SAM 用於特定使用案例的主題，例如使用 Lambda 層、使用巢狀應用程式、控制對 API Gateway APIs 存取、使用 Step Functions 協調 AWS 資源，以及簽署應用程式的程式碼。您需要完成的三個主要里程碑才能開發您的應用程式，如下所示。

主題

- [在 中建立您的應用程式 AWS SAM](#)
- [使用 定義您的基礎設施 AWS SAM](#)
- [使用 建置您的應用程式 AWS SAM](#)

在 中建立您的應用程式 AWS SAM

完成[入門](#)和閱讀 之後[如何使用 AWS Serverless Application Model \(AWS SAM\)](#)，您就可以在開發人員環境中建立 AWS SAM 專案。您的 AWS SAM 專案將做為撰寫無伺服器應用程式的起點。如需 `sam init` 命令選項的 AWS SAM CLI 清單，請參閱 [sam init](#)。

Command AWS Serverless Application Model Line Interface (AWS SAM CLI) `sam init` 命令提供選項來初始化新的無伺服器應用程式，其中包含：

- 用來定義基礎設施程式碼的 AWS SAM 範本。
- 組織您應用程式的資料夾結構。
- AWS Lambda 事件函數的組態。

若要建立 AWS SAM 專案，請參閱本節中的主題。

Note

`sam init` 在支援的執行時間 (TypeScript、Python、Java) 中包含用於持久函數的專案範本。這些範本提供用於建置具狀態無伺服器應用程式的入門程式碼和組態。

主題

- [初始化新的無伺服器應用程式](#)

- [sam init 的選項](#)
- [疑難排解](#)
- [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 SAM CLI

1. cd 到起始目錄。
2. 在命令列執行下列項目：

```
$ sam init
```

3. AWS SAM CLI 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：使用 部署 Hello World 應用程式 AWS SAM](#)，此命令會初始化您的無伺服器應用程式，建立您的專案目錄。此目錄將包含數個檔案和資料夾。最重要的檔案是 `template.yaml`。這是您的 AWS SAM 範本。您的 python 版本必須與 `sam init` 命令建立的 `template.yaml` 檔案中列出的 python 版本相符。

選擇開始範本

範本包含下列項目：

1. 基礎設施程式碼的 AWS SAM 範本。
2. 開始的專案目錄，可組織您的專案檔案。例如，這可能包括：
 - a. Lambda 函數程式碼及其相依性的結構。
 - b. 包含本機測試測試事件的 `events` 資料夾。
 - c. 支援單元測試的 `tests` 資料夾。
 - d. 用於設定專案設定 `samconfig.toml` 的檔案。
 - e. ReadMe 檔案和其他基本啟動專案檔案。

以下是開始專案目錄的範例：

```
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 Quick Start 範本清單中選擇，或提供您自己的自訂範本位置。

選擇 AWS Quick Start 範本

1. 出現提示時，選取 AWS Quick Start 範本。
2. 選取要開始的 AWS Quick Start 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
```

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 SAM CLI 將提示您提供範本位置。

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

將下列任何位置提供給範本 .zip 檔案封存：

- GitHub 儲存庫 – 儲存GitHub庫中 .zip 檔案的路徑。檔案必須位於儲存庫的根目錄中。
- Mercurial 儲存庫 – 儲存Mercurial庫中 .zip 檔案的路徑。檔案必須位於儲存庫的根目錄中。
- .zip 路徑 – .zip 檔案的 HTTPS 或本機路徑。

3. AWS SAM CLI 將使用自訂範本初始化無伺服器應用程式。

選擇執行時間

當您選擇 AWS Quick Start 範本時，AWS SAM CLI會提示您選取 Lambda 函數的執行時間。顯示 AWS SAM CLI的選項清單是 Lambda 原生支援的執行時間。

- [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。

您可以使用任何其他程式設計語言搭配自訂執行時間。若要這樣做，您需要手動建立您的啟動應用程式結構。然後，您可以使用 `設定自訂範本位置sam init`，快速初始化應用程式。

從您的選擇中，AWS SAM CLI會為您的 Lambda 函數程式碼和相依性建立起始目錄。

如果 Lambda 支援執行時間的多個相依性管理員，系統會提示您選擇偏好的相依性管理員。

選擇套件類型

當您選擇 AWS Quick Start 範本和執行時間時，會 AWS SAM CLI提示您選取套件類型。套件類型會決定如何部署 Lambda 函數以搭配 Lambda 服務使用。支援的兩種套件類型為：

1. 容器映像 – 包含基本作業系統、執行時間、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 Lambda 函數封裝為容器映像之應用程式的範例目錄結構。會 AWS SAM CLI下載映像，並在Dockerfile函數的目錄中建立 以指定映像。

```
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
```

以下是應用程式的範例目錄結構，其中包含封裝為 .zip 檔案封存的 函數。

```
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

您可以選擇啟用 AWS X-Ray 追蹤。若要進一步了解，請參閱[什麼是 AWS X-Ray？](#) 《AWS X-Ray 開發人員指南》中的 。

如果您啟用 ， AWS SAM CLI會設定您的 AWS SAM 範本。以下是範例：

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

使用 Amazon CloudWatch Application Insights 設定監控

您可以選擇使用 Amazon CloudWatch Application Insights 啟用監控。若要進一步了解，請參閱《[Amazon CloudWatch 使用者指南](#)》中的 [Amazon CloudWatch Application Insights](#)。 Amazon CloudWatch

如果您啟用 ， AWS SAM CLI會設定您的 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

```

為您的應用程式命名

為您的應用程式提供名稱。會使用此名稱為您的應用程式 AWS SAM CLI 建立頂層資料夾。

sam init 的選項

以下是您可以搭配 `sam init` 命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 `--location` 選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
```

在沒有互動式流程的情況下初始化應用程式

使用 `--no-interactive` 選項，並在命令列提供您的組態選擇，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
```

疑難排解

若要對 進行故障診斷 AWS SAM CLI，請參閱 [AWS SAM CLI 故障診斷](#)。

範例

使用 Hello World AWS Starter 範本初始化新的無伺服器應用程式

在此範例中，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

以下是為您的自訂範本提供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 可連線的路徑範例：

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

進一步了解

若要進一步了解如何使用 `sam init` 命令，請參閱以下內容：

- [學習 AWS SAM : sam init](#) – 上的Serverless Land「學習 AWS SAM」系列YouTube。
- [建構無伺服器應用程式以搭配 AWS SAM CLI \(使用 SAM S2E7 的工作階段 \)](#) – 在 上使用序列的 AWS SAM 工作階段YouTube。

後續步驟

現在您已建立 AWS SAM 專案，即可開始撰寫應用程式。[使用 定義您的基礎設施 AWS SAM](#) 如需您需要完成之任務的詳細說明，請參閱。

使用 定義您的基礎設施 AWS SAM

現在您已建立專案，即可使用 定義應用程式基礎設施 AWS SAM。透過設定您的 AWS SAM 範本來定義應用程式的資源和屬性來執行此操作，這是您 AWS SAM 專案中的 `template.yaml` 檔案。

本節中的主題提供有關在 AWS SAM 範本（您的 `template.yaml` 檔案）中定義基礎設施的內容。它還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 APIs 存取、使用 Step Functions 協調 AWS 資源、程式碼簽署應用程式，以及驗證 AWS SAM 範本。

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](#)
- [透過 使用 Lambda 層提高效率 AWS SAM](#)
- [在 中使用巢狀應用程式重複使用程式碼和資源 AWS SAM](#)
- [在 中使用 EventBridge 排程器管理以時間為基礎的事件 AWS SAM](#)
- [使用 協調 AWS SAM 資源 AWS Step Functions](#)
- [為您的 AWS SAM 應用程式設定程式碼簽署](#)
- [驗證 AWS SAM 範本檔案](#)

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 AWS 的資源。當您定義資源時，您可以識別資源是什麼、它如何與其他資源互動，以及如何存取它（即資源的許可）。

AWS SAM 範本的 `Resources` 區段可以包含 CloudFormation 資源和資源的組合 AWS SAM。此外，您可以針對下列資源使用 AWS SAM 的速記語法：

AWS SAM 速記語法	它會如何處理相關 AWS 資源
AWS::Serverless::Api	建立可透過 HTTPS 端點叫用之 API Gateway 資源和方法的集合。
AWS::Serverless::Application	從 AWS Serverless Application Repository 或從 Amazon S3 儲存貯體將無伺服器應用程式內嵌為巢狀應用程式。
AWS::Serverless::Connector	設定兩個資源之間的許可。如需連接器的簡介，請參閱 使用 AWS SAM 連接器管理資源許可 。
AWS::Serverless::Function	建立 AWS Lambda 事件函數、AWS Identity and Access Management (IAM) 執行角色，以及觸發函數的事件來源映射。
AWS::Serverless::GraphQLApi	會為您的無伺服器應用程式建立 AWS AppSync GraphQL 和設定 API。
AWS::Serverless::HttpApi	建立 Amazon API Gateway HTTP API，這可讓您建立延遲較低且成本低於 REST API 的 RESTful APIs。APIs
AWS::Serverless::LayerVersion	建立 Lambda LayerVersion，其中包含 Lambda 函數所需的程式庫或執行時間程式碼。
AWS::Serverless::SimpleTable	建立具有單一屬性主索引鍵的 DynamoDB 資料表。
AWS::Serverless::StateMachine	建立 AWS Step Functions 狀態機器，您可以使用它來協調 AWS Lambda 函數和其他 AWS 資源，以形成複雜且強大的工作流程。

上述資源也會列在 [中 AWS SAM 資源和屬性](#)。

如需所有 AWS 資源和屬性類型 CloudFormation 和 AWS SAM 支援的參考資訊，請參閱 AWS CloudFormation 《使用者指南》中的 [AWS 資源和屬性類型參考](#)。

在 AWS SAM 範本中設定和管理資源存取

若要讓您的 AWS 資源彼此互動，必須在資源之間設定適當的存取和許可。這樣做需要 AWS Identity and Access Management (IAM) 使用者、角色和政策的組態，才能以安全的方式完成您的互動。

本節中的主題都與設定對範本中定義之資源的存取有關。本節從一般最佳實務開始。以下兩個主題會檢閱您在無伺服器應用程式中參考的資源之間設定存取和許可的兩個選項：AWS SAM 連接器和 AWS SAM 政策範本。最後一個主題提供使用相同的機制來管理使用者存取的詳細資訊，CloudFormation 以管理使用者。

若要進一步了解，請參閱AWS CloudFormation 《使用者指南》中的[使用 控制存取 AWS Identity and Access Management](#)。

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和許可管理。

1. AWS SAM 連接器
2. AWS SAM 政策範本

AWS SAM 連接器

連接器是在兩個資源之間佈建許可的一種方式。您可以透過描述它們如何在 AWS SAM 範本中彼此互動來執行此操作。您可以使用Connectors資源屬性或AWS::Serverless::Connector資源類型來定義它們。連接器支援佈建Read和Write存取 AWS 資源組合之間的資料和事件。若要進一步了解 AWS SAM 連接器，請參閱 [使用 AWS SAM 連接器管理資源許可](#)。

AWS SAM 政策範本

AWS SAM 政策範本是預先定義的一組許可，您可以將這些許可新增至範本 AWS SAM，以管理 AWS Lambda Event Functions、AWS Step Functions 陳述機器與其互動的資源之間的存取和許可。若要進一步了解 AWS SAM 政策範本，請參閱 [AWS SAM 政策範本](#)。

AWS CloudFormation 機制

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。如需詳細資訊，請參閱 [使用 CloudFormation 機制管理 AWS SAM 許可](#)。

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許可。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一起使用多個選項。以下是選擇最適合您的選項時需要考慮的一些事項：

- AWS SAM 連接器和政策範本都降低了促進 AWS 資源之間安全互動所需的 IAM 專業知識。支援時使用連接器和政策範本。
- AWS SAM 連接器提供簡單且直覺式的速記語法，以定義 AWS SAM 範本中的許可，並需要最少的 IAM 專業知識。當同時支援 AWS SAM 連接器和政策範本時，請使用 AWS SAM 連接器。
- AWS SAM 連接器可以在支援的 AWS SAM 來源和目的地資源之間佈建 Read 和 Write 存取資料和事件。如需支援的資源清單，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、Step Functions 狀態機器與其互動 AWS 的資源之間的許可，但政策範本確實支援所有 CRUD 操作。支援時，以及當適用於您案例 AWS SAM 的政策範本可用時，請使用 AWS SAM 政策範本。如需可用政策範本的清單，請參閱 [AWS SAM 政策範本](#)。
- 對於所有其他案例，或需要精細程度時，請使用 CloudFormation 機制。

使用 AWS SAM 連接器管理資源許可

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範圍廣泛的許可。

AWS SAM 連接器的優點

透過在資源之間自動編寫適當的存取政策，連接器可讓您撰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、政策語言和服務特定安全設定的專業知識。因此，連接器對於可能剛接觸無伺服器開發的開發人員，或想要提高其開發速度的經驗豐富的開發人員來說，是很大的好處。

使用 AWS SAM 連接器

在來源 Connectors 資源中內嵌資源，以使用資源屬性。然後，定義目的地資源，並描述資料或事件應該如何在這些資源之間流動。AWS SAM 然後，編寫必要的存取政策，以促進必要的互動。

以下概述如何寫入此資源屬性：

```
AWS::Template::FormatVersion: '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>
    ...

```

連接器的運作方式

Note

本節說明連接器如何在幕後佈建必要的資源。這會在您使用連接器時自動發生。

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，以下是內嵌連接器：

```

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
```

這將產生下列 `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
```

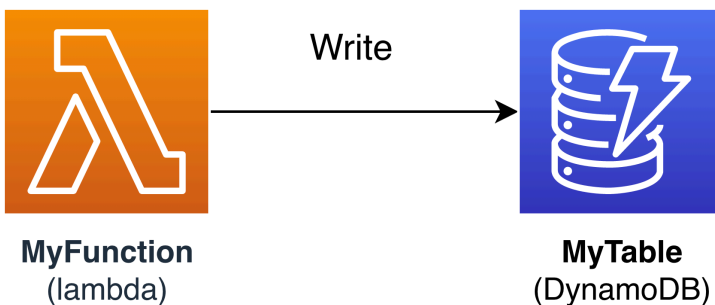
Note

您也可以使用此語法在 AWS SAM 範本中定義連接器。當您的來源資源在與連接器不同的範本上定義時，建議使用此選項。

接下來，會自動編寫此連線的必要存取政策。如需連接器產生之資源的詳細資訊，請參閱 [CloudFormation 當您指定 時產生的資源 AWS::Serverless::Connector](#)。

連接器範例

下列範例示範如何使用連接器將資料從 AWS Lambda 事件函數寫入 Amazon DynamoDB 資料表。



```
Transform: AWS::Serverless-2016-10-31
Resources:
  MyTable:
```

```

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

```

Connectors 資源屬性內嵌在 Lambda 函數來源資源中。DynamoDB 資料表是使用 Id 屬性定義為目的地資源。連接器將在這兩個資源之間佈建Write許可。

當您將 AWS SAM 範本部署到 時 CloudFormation，AWS SAM 會自動編寫此連線運作所需的必要存取政策。

來源和目的地資源之間的支援連線

連接器支援來源和目的地資源連線的選取組合之間的ReadWrite資料和事件許可類型。例如，連接器支援AWS::ApiGateway::RestApi來源資源與AWS::Lambda::Function目的地資源之間的Write連線。

您可以使用支援的屬性組合來定義來源和目的地資源。屬性需求將取決於您建立的連線，以及定義資源的位置。

 Note

連接器可以在支援的無伺服器和非無伺服器資源類型之間佈建許可。

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的地資源類型](#)。

在 中定義讀取和寫入許可 AWS SAM

在 Read 和 中 AWS SAM，Write 許可可以在單一連接器中佈建：

```
AWS::Serverless::Connector::TableConn
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
```

如需使用連接器的詳細資訊，請參閱 [AWS SAM 連接器參考](#)。

在 中使用其他支援的屬性來定義資源 AWS SAM

對於來源和目的地資源，當在相同範本中定義時，請使用 Id 屬性。或者，Qualifier 您可以新增 來縮小已定義資源的範圍。當資源不在相同的範本內時，請使用支援的屬性組合。

- 如需來源和目的地資源支援的屬性組合清單，請參閱 [連接器支援的來源和目的地資源類型](#)。
- 如需可與連接器搭配使用的屬性說明，請參閱 [AWS::Serverless::Connector](#)。

當您使用 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>

```

以下是使用 Qualifier 縮小 Amazon API Gateway 資源範圍的範例：

```

AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyApi:
    Type: AWS::Serverless::Api
    Connectors:
      ApiToLambdaConn:
        Properties:
          SourceReference:
            Qualifier: Prod/GET/foobar
          Destination:
            Id: MyFunction
          Permissions:
            - Write
    ...

```

以下是範例，使用支援的 Arn 和 組合 Type，從另一個範本定義目的地資源：

```

AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:

```

```
Type: AWS::Serverless::Function
Connectors:
  TableConn:
    Properties:
      Destination:
        Type: AWS::DynamoDB::Table
        Arn: !GetAtt MyTable.Arn
...

```

如需使用連接器的詳細資訊，請參閱 [AWS SAM 連接器參考](#)。

在 中從單一來源建立多個連接器 AWS SAM

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的地資源。

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Connectors:
      BucketConn:
        Properties:
          Destination:
            Id: amzn-s3-demo-bucket
          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
...

```

如需使用連接器的詳細資訊，請參閱 [AWS SAM 連接器參考](#)。

在 中建立多目的地連接器 AWS SAM

在來源資源中，您可以定義具有多個目的地資源的單一連接器。以下是連線至 Amazon Simple Storage Service (Amazon S3) 儲存貯體和 DynamoDB 資料表的 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

```

如需使用連接器的詳細資訊，請參閱 [AWS SAM 連接器參考](#)。

在 中使用連接器定義資源屬性 AWS SAM

您可以為資源定義資源屬性，以指定其他行為和關係。若要進一步了解資源屬性，請參閱AWS CloudFormation 《使用者指南》中的[資源屬性參考](#)。

您可以在與連接器屬性相同的層級上定義資源屬性，以將資源屬性新增至內嵌連接器。在部署時轉換 AWS SAM 範本時，屬性會傳遞至產生的資源。

```
AWS::Serverless::Function
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Connectors:
      MyConn:
        DeletionPolicy: Retain
        DependsOn: AnotherFunction
        Properties:
          ...
```

如需使用連接器的詳細資訊，請參閱 [AWS SAM 連接器參考](#)。

進一步了解

如需使用 AWS SAM 連接器的詳細資訊，請參閱下列主題：

- [AWS::Serverless::Connector](#)
- [在 中定義讀取和寫入許可 AWS SAM](#)
- [在 中使用其他支援的屬性來定義資源 AWS SAM](#)
- [在 中從單一來源建立多個連接器 AWS SAM](#)
- [在 中建立多目的地連接器 AWS SAM](#)
- [在 中定義讀取和寫入許可 AWS SAM](#)
- [在 中使用連接器定義資源屬性 AWS SAM](#)

提供意見回饋

若要提供連接器的意見回饋，請在 serverless-application-model AWS GitHub 儲存庫 [提交新問題](#)。

AWS SAM 政策範本

AWS Serverless Application Model (AWS SAM) 可讓您從政策範本清單中選擇，將 Lambda 事件函數和 AWS Step Functions 狀態機器的許可範圍限定為應用程式使用的資源。

AWS SAM 中使用 AWS Serverless Application Repository 政策範本的應用程式不需要任何特殊的客戶確認，即可從 部署應用程式 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 AWS SAM GitHub 專案develop分支中的 `policy_templates.json` 來源檔案提交提取請求。您可以在 GitHub 網站上的 [policy_templates.json](#) 中找到來源檔案。
2. 在 AWS SAM GitHub 專案中提交問題，其中包含提取請求的原因和請求的連結。使用此連結提交新問題：[AWS Serverless Application Model：問題](#)。

語法

對於您在 AWS SAM 範本檔案中指定的每個政策範本，您必須一律指定包含政策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
```

Note

如果您已設定一般 IAM 政策，或透過 Lambda 擁有受管政策，則可以不使用空白物件來設定政策範本。

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，或請求該佇列做為應用程式的參數。

```
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，您仍必須指定空物件，否則會產生錯誤。

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

範例 3：具有預留位置值和一般 IAM 政策的政策範本

下列範例包含 AmazonSQSFullAccess 政策和[DynamoDBCrudPolicy](#)政策範本。

AmazonSQSFullAccess 政策是 IAM 政策而非 AWS SAM 政策，因此您不需要指定空白物件，因為政策會直接傳遞給 CloudFormation。

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

Policies:

- AmazonSQSFullAccess // IAM policy could be set without passing an empty object
- DynamoDBCrudPolicy: // SAM specific policy, has a defined structure

TableName:

!Ref SampleTable

政策範本資料表

以下是可用政策範本的資料表。

政策範本	說明		
AcmGetCertificatePolicy	授予讀取憑證的許可 AWS Certificate Manager。		
AMIDescribePolicy	提供描述 Amazon Machine Image (AMIs) 許可。		
AthenaQueryPolicy	提供執行 Athena 查詢的許可。		
AWSSecretsManagerGetSecretValuePolicy	准許取得指定 AWS Secrets Manager 秘密的秘密值。		
AWSSecretsManagerRotationPolicy	授予在其中輪換秘密的許可 AWS Secrets Manager。		
CloudFormationDescribeStacksPolicy	提供描述 CloudFormation 堆疊的許可。		
CloudWatchDashboardPolicy	授予在 CloudWatch 儀表板上操作指標的許可。		

政策範本	說明		
CloudWatchDescribeAlarmHistoryPolicy	提供描述 CloudWatch 警示歷史記錄的許可。		
CloudWatchPutMetricPolicy	提供將指標傳送至 CloudWatch 的許可。		
CodeCommitCrudPolicy	授予在特定 CodeCommit 儲存庫中create/read/update/刪除物件的許可。		
CodeCommitReadPolicy	授予讀取特定 CodeCommit 儲存庫內物件的許可。		
CodePipelineLambdaExecutionPolicy	准許 CodePipeline 調用 Lambda 函數來報告任務的狀態。		
CodePipelineReadOnlyPolicy	提供讀取許可，以取得 CodePipeline 管道的詳細資訊。		
ComprehendBasicAccessPolicy	提供偵測實體、金鑰片語、語言和情緒的許可。		
CostExplorerReadOnlyPolicy	為帳單歷史記錄的唯讀 Cost Explorer APIs 提供唯讀許可。		
DynamoDBBackupFullAccessPolicy	為資料表的 DynamoDB 隨需備份提供讀取和寫入許可。		

政策範本	說明		
DynamoDBC rudPolicy	為 Amazon DynamoDB 資料表提供建立、讀取、更新和刪除許可。		
DynamoDBR eadPolicy	為 DynamoDB 資料表提供唯讀許可。		
DynamoDBR econfigur ePolicy	提供重新設定 DynamoDB 資料表的許可。		
DynamoDBR estoreFro mBackupPolicy	提供從備份還原 DynamoDB 資料表的許可。		
DynamoDBS treamRead Policy	提供描述和讀取 DynamoDB 串流和記錄的許可。		
DynamoDBW ritePolicy	為 DynamoDB 資料表提供唯寫許可。		
EC2CopyIm agePolicy	提供複製 Amazon EC2 映像的許可。		
EC2Descri bePolicy	提供描述 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體的許可。		
EcsRunTas kPolicy	提供為任務定義啟動新任務的許可。		
EFSWriteA ccessPolicy	准許掛載具有寫入存取權的 Amazon EFS 檔案系統。		
EKSDescri bePolicy	提供描述或列出 Amazon EKS 叢集的許可。		

政策範本	說明		
ElasticMapReduceJobFlowStepsPolicy	准許將新步驟新增至執行中的叢集。		
ElasticMapReduceCancelStepsPolicy	准許取消執行中叢集中的待處理步驟。		
ElasticMapReduceModifyInstanceFleetPolicy	准許列出叢集內執行個體機群的詳細資訊和修改容量。		
ElasticMapReduceModifyInstanceGroupsPolicy	准許列出叢集內執行個體群組的詳細資訊和修改設定。		
ElasticMapReduceSetTerminationProtectionPolicy	准許設定叢集的終止保護。		
ElasticMapReduceTerminateJobFlowsPolicy	提供關閉叢集的許可。		
ElasticsearchHttpPostPolicy	將 POST 許可授予 Amazon OpenSearch Service。		

政策範本	說明		
EventBridgePutEventsPolicy	提供將事件傳送至 EventBridge 的許可。		
FilterLogEventsPolicy	提供從指定日誌群組篩選 CloudWatch Logs 事件的許可。		
FirehoseCudPolicy	提供建立、寫入、更新和刪除 Firehose 交付串流的許可。		
FirehoseWritePolicy	提供寫入 Firehose 交付串流的許可。		
KinesisCudPolicy	提供建立、發佈和刪除 Amazon Kinesis 串流的許可。		
KinesisStreamReadPolicy	提供列出和讀取 Amazon Kinesis 串流的許可。		
KMSTDecryptPolicy	提供使用 AWS Key Management Service (AWS KMS) 金鑰解密的許可。		
KMSEncryptPolicy	提供使用 AWS Key Management Service (AWS KMS) 金鑰加密的許可。		
LambdaInvokePolicy	提供叫用 AWS Lambda 函數、別名或版本的許可。		
MobileAnalyticsWriteOnlyAccessPolicy	提供僅寫入許可，以放置所有應用程式資源的事件資料。		
OrganizationsListAccountsPolicy	提供唯讀許可，以列出子帳戶名稱和 IDs。		

政策範本	說明		
PinpointEndpointAccessPolicy	提供取得和更新 Amazon Pinpoint 應用程式端點的許可。		
PollyFullAccessPolicy	提供 Amazon Polly 語彙資源的完整存取許可。		
RekognitionDetectOnlyPolicy	提供偵測臉部、標籤和文字的許可。		
RekognitionFacesManagementPolicy	授予在 Amazon Rekognition 集合中新增、刪除和搜尋臉部的許可。		
RekognitionFacesPolicy	提供許可來比較和偵測臉部和標籤。		
RekognitionLabelsPolicy	提供偵測物件和管制標籤的許可。		
RekognitionNoDataAccessPolicy	提供許可來比較和偵測臉部和標籤。		
RekognitionReadPolicy	提供列出和搜尋臉部的許可。		
RekognitionWriteOnAccessPolicy	提供建立集合和索引臉部的許可。		
Route53ChangeResourceRecordSetsPolicy	提供在 Route 53 中變更資源紀錄集的許可。		

政策範本	說明		
S3CrudPolicy	授予建立、讀取、更新和刪除許可，以對 Amazon S3 儲存貯體中的物件採取行動。		
S3FullAccessPolicy	提供對 Amazon S3 儲存貯體中物件採取動作的完整存取許可。		
S3ReadPolicy	提供唯讀許可，以讀取 Amazon Simple Storage Service (Amazon S3) 儲存貯體中的物件。		
S3WritePolicy	提供將物件寫入 Amazon S3 儲存貯體的寫入許可。		
SageMakerCreateEndpointConfigurationPolicy	提供在 SageMaker AI 中建立端點組態的許可。		
SageMakerCreateEndpointPolicy	提供在 SageMaker AI 中建立端點的許可。		
ServerlessRepoReadWriteAccessPolicy	授予在 AWS Serverless Application Repository 服務中建立和列出應用程式的許可。		
SESBulkTemplatedCrudPolicy	提供傳送電子郵件、範本電子郵件、範本大量電子郵件和驗證身分的許可。		
SESBulkTemplatedCrudPolicy_v2	准許傳送 Amazon SES 電子郵件、範本電子郵件和範本大量電子郵件，以及驗證身分。		
SESCrudPolicy	提供傳送電子郵件和驗證身分的許可。		

政策範本	說明		
SESEmailTemplateCrudPolicy	提供建立、取得、列出、更新和刪除 Amazon SES 電子郵件範本的許可。		
SESSendBouncePolicy	將 SendBounce 許可授予 Amazon Simple Email Service (Amazon SES) 身分。		
SNSCrudPolicy	提供建立、發佈和訂閱 Amazon SNS 主題的許可。		
SNSPublishMessagePolicy	准許將訊息發佈至 Amazon Simple Notification Service (Amazon SNS) 主題。		
SQSPollerPolicy	提供輪詢 Amazon Simple Queue Service (Amazon SQS) 佇列的許可。		
SQSSendMessagePolicy	提供將訊息傳送至 Amazon SQS 佇列的許可。		
SSMParameterReadPolicy	准許從 Amazon EC2 Systems Manager (SSM) 參數存放區存取參數，以在此帳戶中載入秘密。當參數名稱沒有斜線字首時，請使用。		
SSMParameterWithSlashPrefixReadPolicy	准許從 Amazon EC2 Systems Manager (SSM) 參數存放區存取參數，以在此帳戶中載入秘密。當參數名稱具有斜線字首時，請使用。		
StepFunctionsExecutionPolicy	提供啟動 Step Functions 狀態機器執行的許可。		
TextractDetectAnalyzePolicy	提供使用 Amazon Textract 偵測和分析文件的存取權。		

政策範本	說明		
TextractGetResultPolicy	提供從 Amazon Textract 取得偵測和分析文件的存取權。		
TextractPolicy	提供 Amazon Textract 的完整存取權。		
VPCAccessPolicy	提供建立、刪除、描述和分離彈性網路介面的存取權。		

疑難排解

SAM CLI 錯誤：「必須為政策範本「<policy-template-name>」指定有效的參數值

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政策範本時，未傳遞空物件。

若要修正此問題，請宣告 政策，如下列 範例所示 [CloudWatchPutMetricPolicy](#)。

```
MyFunction:
  Policies:
    - CloudWatchPutMetricPolicy: {}
```

AWS SAM 政策範本清單

以下是可用的政策範本，以及套用至每個政策範本的許可。AWS Serverless Application Model (AWS SAM) 會自動將適當的資訊填入預留位置項目（例如 AWS 區域和帳戶 ID）。

主題

- [AcmGetCertificatePolicy](#)
- [AMIDescribePolicy](#)
- [AthenaQueryPolicy](#)
- [AWSSecretsManagerGetSecretValuePolicy](#)

- [AWS Secrets Manager Rotation Policy](#)
- [CloudFormation Describe Stacks Policy](#)
- [CloudWatch Dashboard Policy](#)
- [CloudWatch Describe Alarm History Policy](#)
- [CloudWatch Put Metric Policy](#)
- [CodePipeline Lambda Execution Policy](#)
- [CodePipeline Read Only Policy](#)
- [CodeCommit Crud Policy](#)
- [CodeCommit Read Policy](#)
- [Comprehend Basic Access Policy](#)
- [Cost Explorer Read Only Policy](#)
- [DynamoDB Backup Full Access Policy](#)
- [DynamoDB Crud Policy](#)
- [DynamoDB Read Policy](#)
- [DynamoDB Reconfigure Policy](#)
- [DynamoDB Restore From Backup Policy](#)
- [DynamoDB Stream Read Policy](#)
- [DynamoDB Write Policy](#)
- [EC2 Copy Image Policy](#)
- [EC2 Describe Policy](#)
- [ECS Run Task Policy](#)
- [EFS Write Access Policy](#)
- [EKS Describe Policy](#)
- [Elastic MapReduce Add Job Flow Steps Policy](#)
- [Elastic MapReduce Cancel Steps Policy](#)
- [Elastic MapReduce Modify Instance Fleet Policy](#)
- [Elastic MapReduce Modify Instance Groups Policy](#)
- [Elastic MapReduce Set Termination Protection Policy](#)
- [Elastic MapReduce Terminate Job Flows Policy](#)
- [Elasticsearch Http Post Policy](#)

- [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

授予讀取憑證的許可 AWS Certificate Manager。

```
"Statement": [  
  {  
    "Effect": "Allow",  
    "Action": [  
      "acm:GetCertificate"  
    ],  
    "Resource": {  
      "Fn::Sub": [  
        "${certificateArn}",  
        {  
          "certificateArn": {  
            "Ref": "CertificateArn"  
          }  
        }  
      ]  
    }  
  }  
]
```



```
]
```

AMIDescribePolicy

提供描述 Amazon Machine Image (AMIs) 許可。

```
"Statement": [  
  {  
    "Effect": "Allow",  
    "Action": [  
      "ec2:DescribeImages"  
    ],  
    "Resource": "*"   
  }  
]
```

AthenaQueryPolicy

提供執行 Athena 查詢的許可。

```
"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"
        }
      }
    ]
  }
}
]

```

AWS Secrets Manager GetSecretValue Policy

准許取得指定秘密的 AWS Secrets Manager 秘密值。

```

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "secretsmanager:GetSecretValue"
    ],
    "Resource": {
      "Fn::Sub": [
        "${secretArn}",
        {
          "secretArn": {
            "Ref": "SecretArn"
          }
        }
      ]
    }
  }
]

```

]

AWS Secrets Manager Rotation Policy

授予在其中輪換秘密的許可 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

提供描述 CloudFormation 堆疊的許可。

```
"Statement": [  
  {  
    "Effect": "Allow",  
    "Action": [  
      "cloudformation:DescribeStacks"  
    ],  
    "Resource": {  
      "Fn::Sub": "arn:${AWS::Partition}:cloudformation:${AWS::Region}:  
${AWS::AccountId}:stack/*"  
    }  
  }  
]
```

CloudWatchDashboardPolicy

授予在 CloudWatch 儀表板上操作指標的許可。

```
"Statement": [  
  {  
    "Effect": "Allow",  
    "Action": [  
      "cloudwatch:GetDashboard",  
      "cloudwatch:ListDashboards",  
      "cloudwatch:PutDashboard",  
      "cloudwatch:ListMetrics"  
    ],  
    "Resource": "*"   
  }  
]
```

CloudWatchDescribeAlarmHistoryPolicy

提供描述 Amazon CloudWatch 警示歷史記錄的許可。

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

```
    ],  
    "Resource": "*"    
  }  
]
```

CloudWatchPutMetricPolicy

提供將指標傳送至 CloudWatch 的許可。

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

CodePipelineLambdaExecutionPolicy

提供 叫用 AWS CodePipeline 以報告任務狀態的 Lambda 事件函數許可。

```
"Statement": [  
  {  
    "Effect": "Allow",  
    "Action": [  
      "codepipeline:PutJobSuccessResult",  
      "codepipeline:PutJobFailureResult"  
    ],  
    "Resource": "*"    
  }  
]
```

CodePipelineReadOnlyPolicy

提供讀取許可，以取得 CodePipeline 管道的詳細資訊。

```
"Statement": [  
  {  
    "Effect": "Allow",  
    "Action": [  
      "codepipeline:ListPipelineExecutions"  
    ]  
  }  
]
```

```

    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:codepipeline:${AWS::Region}:${AWS::AccountId}:",
        "${pipelineName}",
        {
          "pipelineName": {
            "Ref": "PipelineName"
          }
        }
      ]
    }
  }
}
]

```

CodeCommitCrudPolicy

授予在特定 CodeCommit 儲存庫中建立、讀取、更新和刪除物件的許可。

```

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "codecommit:GitPull",
      "codecommit:GitPush",
      "codecommit:CreateBranch",
      "codecommit>DeleteBranch",
      "codecommit:GetBranch",
      "codecommit:ListBranches",
      "codecommit:MergeBranchesByFastForward",
      "codecommit:MergeBranchesBySquash",
      "codecommit:MergeBranchesByThreeWay",
      "codecommit:UpdateDefaultBranch",
      "codecommit:BatchDescribeMergeConflicts",
      "codecommit>CreateUnreferencedMergeCommit",
      "codecommit:DescribeMergeConflicts",
      "codecommit:GetMergeCommit",
      "codecommit:GetMergeOptions",
      "codecommit:BatchGetPullRequests",
      "codecommit>CreatePullRequest",
      "codecommit:DescribePullRequestEvents",
      "codecommit:GetCommentsForPullRequest",
      "codecommit:GetCommitsFromMergeBase",
      "codecommit:GetMergeConflicts",

```

```

    "codecommit:GetPullRequest",
    "codecommit:ListPullRequests",
    "codecommit:MergePullRequestByFastForward",
    "codecommit:MergePullRequestBySquash",
    "codecommit:MergePullRequestByThreeWay",
    "codecommit:PostCommentForPullRequest",
    "codecommit:UpdatePullRequestDescription",
    "codecommit:UpdatePullRequestStatus",
    "codecommit:UpdatePullRequestTitle",
    "codecommit>DeleteFile",
    "codecommit:GetBlob",
    "codecommit:GetFile",
    "codecommit:GetFolder",
    "codecommit:PutFile",
    "codecommit>DeleteCommentContent",
    "codecommit:GetComment",
    "codecommit:GetCommentsForComparedCommit",
    "codecommit:PostCommentForComparedCommit",
    "codecommit:PostCommentReply",
    "codecommit:UpdateComment",
    "codecommit:BatchGetCommits",
    "codecommit>CreateCommit",
    "codecommit:GetCommit",
    "codecommit:GetCommitHistory",
    "codecommit:GetDifferences",
    "codecommit:GetObjectIdentifier",
    "codecommit:GetReferences",
    "codecommit:GetTree",
    "codecommit:GetRepository",
    "codecommit:UpdateRepositoryDescription",
    "codecommit:ListTagsForResource",
    "codecommit:TagResource",
    "codecommit:UntagResource",
    "codecommit:GetRepositoryTriggers",
    "codecommit:PutRepositoryTriggers",
    "codecommit:TestRepositoryTriggers",
    "codecommit:GetBranch",
    "codecommit:GetCommit",
    "codecommit:UploadArchive",
    "codecommit:GetUploadArchiveStatus",
    "codecommit:CancelUploadArchive"
  ],
  "Resource": {
    "Fn::Sub": [

```

```

    "arn:${AWS::Partition}:codecommit:${AWS::Region}:${AWS::AccountId}:
    ${repositoryName}",
    {
      "repositoryName": {
        "Ref": "RepositoryName"
      }
    }
  ]
}
]

```

CodeCommitReadPolicy

授予讀取特定 CodeCommit 儲存庫內物件的許可。

```

"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

提供偵測實體、金鑰片語、語言和情緒的許可。

```

"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "comprehend:BatchDetectKeyPhrases",
            "comprehend:DetectDominantLanguage",
            "comprehend:DetectEntities",
            "comprehend:BatchDetectEntities",
            "comprehend:DetectKeyPhrases",
            "comprehend:DetectSentiment",
            "comprehend:BatchDetectDominantLanguage",
            "comprehend:BatchDetectSentiment"
        ],
        "Resource": "*"
    }
]

```

```
]
```

CostExplorerReadOnlyPolicy

為帳單歷史記錄的唯讀 AWS Cost Explorer (Cost Explorer) APIs 提供唯讀許可。

```
"Statement": [  
  {  
    "Effect": "Allow",  
    "Action": [  
      "ce:GetCostAndUsage",  
      "ce:GetDimensionValues",  
      "ce:GetReservationCoverage",  
      "ce:GetReservationPurchaseRecommendation",  
      "ce:GetReservationUtilization",  
      "ce:GetTags"  
    ],  
    "Resource": "*"   
  }  
]
```

DynamoDBBackupFullAccessPolicy

為資料表的 DynamoDB 隨需備份提供讀取和寫入許可。

```
"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

為 Amazon DynamoDB 資料表提供建立、讀取、更新和刪除許可。

```

"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 資料表提供唯讀許可。

```

"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

提供重新設定 DynamoDB 資料表的許可。

```

"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "dynamodb:UpdateTable"
        ],
        "Resource": {
            "Fn::Sub": [
                "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
${tableName}",
                {
                    "tableName": {
                        "Ref": "TableName"
                    }
                }
            ]
        }
    }
]

```

```
]
```

DynamoDBRestoreFromBackupPolicy

提供從備份還原 DynamoDB 資料表的許可。

```
"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

提供描述和讀取 DynamoDB 串流和記錄的許可。

```
"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

為 DynamoDB 資料表提供唯寫許可。

```

"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

提供複製 Amazon EC2 映像的許可。

```
"Statement": [  
  {  
    "Effect": "Allow",  
    "Action": [  
      "ec2:CopyImage"  
    ],  
    "Resource": {  
      "Fn::Sub": [  
        "arn:${AWS::Partition}:ec2:${AWS::Region}:${AWS::AccountId}:image/${imageId}",  
        {  
          "imageId": {  
            "Ref": "ImageId"  
          }  
        }  
      ]  
    }  
  }  
]
```

EC2DescribePolicy

提供描述 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體的許可。

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

EcsRunTaskPolicy

提供為任務定義啟動新任務的許可。

```

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

```

EFSWriteAccessPolicy

准許掛載具有寫入存取權的 Amazon EFS 檔案系統。

```

"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

提供描述或列出 Amazon Elastic Kubernetes Service (Amazon EKS) 叢集的許可。

```

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "eks:DescribeCluster",
      "eks:ListClusters"
    ],
    "Resource": "*"
  }
]

```

ElasticMapReduceAddJobFlowStepsPolicy

准許將新步驟新增至執行中的叢集。

```

"Statement": [
  {
    "Action": "elasticmapreduce:AddJobFlowSteps",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:
${AWS::AccountId}:cluster/${clusterId}",

```

```

        {
            "clusterId": {
                "Ref": "ClusterId"
            }
        }
    ],
    "Effect": "Allow"
}
]

```

ElasticMapReduceCancelStepsPolicy

准許取消執行中叢集中的待處理步驟。

```

"Statement": [
    {
        "Action": "elasticmapreduce:CancelSteps",
        "Resource": {
            "Fn::Sub": [
                "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:${AWS::AccountId}:cluster/${clusterId}",
                {
                    "clusterId": {
                        "Ref": "ClusterId"
                    }
                }
            ]
        },
        "Effect": "Allow"
    }
]

```

ElasticMapReduceModifyInstanceFleetPolicy

准許列出叢集內執行個體機群的詳細資訊和修改容量。

```

"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

准許列出叢集內執行個體群組的詳細資訊和修改設定。

```

"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

准許設定叢集的終止保護。

```

"Statement": [
  {
    "Action": "elasticmapreduce:SetTerminationProtection",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:
${AWS::AccountId}:cluster/${clusterId}",
        {
          "clusterId": {
            "Ref": "ClusterId"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]

```

ElasticMapReduceTerminateJobFlowsPolicy

提供關閉叢集的許可。

```

"Statement": [
  {
    "Action": "elasticmapreduce:TerminateJobFlows",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:
${AWS::AccountId}:cluster/${clusterId}",
        {
          "clusterId": {
            "Ref": "ClusterId"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]

```

ElasticsearchHttpPostPolicy

將 POST 和 PUT 許可提供給 Amazon OpenSearch Service。

```

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "es:ESHttpPost",
      "es:ESHttpPut"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:es:${AWS::Region}:${AWS::AccountId}:domain/
${domainName}/*",
        {
          "domainName": {
            "Ref": "DomainName"
          }
        }
      ]
    }
  }
]

```

EventBridgePutEventsPolicy

提供將事件傳送至 Amazon EventBridge 的許可。

```

"Statement": [
  {
    "Effect": "Allow",
    "Action": "events:PutEvents",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:events:${AWS::Region}:${AWS::AccountId}:event-bus/
${eventBusName}",
        {
          "eventBusName": {
            "Ref": "EventBusName"
          }
        }
      ]
    }
  }
]

```

FilterLogEventsPolicy

提供從指定日誌群組篩選 CloudWatch Logs 事件的許可。

```
"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

提供建立、寫入、更新和刪除 Firehose 交付串流的許可。

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



```

        {
            "deliveryStreamName": {
                "Ref": "DeliveryStreamName"
            }
        }
    ]
}
]

```

FirehoseWritePolicy

提供寫入 Firehose 交付串流的許可。

```

"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

提供建立、發佈和刪除 Amazon Kinesis 串流的許可。

```

"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

提供列出和讀取 Amazon Kinesis 串流的許可。

```

"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

提供使用 AWS Key Management Service (AWS KMS) 金鑰解密的許可。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

提供使用 AWS KMS 金鑰加密的許可。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

提供叫用 AWS Lambda 函數、別名或版本的許可。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "lambda:InvokeFunction"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:lambda:${AWS::Region}:${AWS::AccountId}:function:${functionName}",
        {
          "functionName": {
            "Ref": "FunctionName"
          }
        }
      ]
    }
  }
]
```

]

}

}

]

}

MobileAnalyticsWriteOnlyAccessPolicy

提供僅寫入許可，以放置所有應用程式資源的事件資料。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "mobileanalytics:PutEvents"
    ],
    "Resource": "*"
  }
]
```

OrganizationsListAccountsPolicy

提供唯讀許可，以列出子帳戶名稱和 IDs。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "organizations:ListAccounts"
    ],
    "Resource": "*"
  }
]
```

PinpointEndpointAccessPolicy

提供取得和更新 Amazon Pinpoint 應用程式端點的許可。

```
"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

提供 Amazon Polly 語彙資源的完整存取許可。

```

"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

提供偵測臉部、標籤和文字的許可。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "rekognition:DetectFaces",
      "rekognition:DetectLabels",
      "rekognition:DetectModerationLabels",
      "rekognition:DetectText"
    ],
    "Resource": "*"
  }
]
```

RekognitionFacesManagementPolicy

授予在 Amazon Rekognition 集合中新增、刪除和搜尋臉部的許可。

```
"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

提供許可來比較和偵測臉部和標籤。

```

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "rekognition:CompareFaces",
      "rekognition:DetectFaces"
    ],
    "Resource": "*"
  }
]

```

RekognitionLabelsPolicy

提供偵測物件和管制標籤的許可。

```

"Statement": [
  {
    "Effect": "Allow",
    "Action": [

```



```
    "rekognition:DetectLabels",
    "rekognition:DetectModerationLabels"
  ],
  "Resource": "*"
}
```

RekognitionNoDataAccessPolicy

提供許可來比較和偵測臉部和標籤。

```
"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

提供列出和搜尋臉部的許可。

```
"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

提供建立集合和索引臉部的許可。

```

"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

提供在 Route 53 中變更資源紀錄集的許可。

```
"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、更新和刪除許可，以對 Amazon S3 儲存貯體中的物件採取行動。

```
"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

提供對 Amazon S3 儲存貯體中物件採取動作的完整存取許可。

```

"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

提供唯讀許可，以讀取 Amazon Simple Storage Service (Amazon S3) 儲存貯體中的物件。

```

"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

提供將物件寫入 Amazon S3 儲存貯體的寫入許可。

```

"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

提供在 SageMaker AI 中建立端點組態的許可。

```

"Statement": [
  {
    "Action": [
      "sagemaker:CreateEndpointConfig"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:sagemaker:${AWS::Region}:${AWS::AccountId}:endpoint-
        config/${endpointConfigName}",
        {
          "endpointConfigName": {
            "Ref": "EndpointConfigName"
          }
        }
      ]
    }
  ]
},

```

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

SageMakerCreateEndpointPolicy

提供在 SageMaker AI 中建立端點的許可。

```
"Statement": [
  {
    "Action": [
      "sagemaker:CreateEndpoint"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:sagemaker:${AWS::Region}:${AWS::AccountId}:endpoint/${endpointName}",
        {
          "endpointName": {
            "Ref": "EndpointName"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

ServerlessRepoReadWriteAccessPolicy

提供在 AWS Serverless Application Repository (AWS SAM) 服務中建立和列出應用程式的許可。

```
"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

准許傳送 Amazon SES 電子郵件、範本電子郵件和範本大量電子郵件，以及驗證身分。

Note

`ses:SendTemplatedEmail` 動作需要範本 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

准許傳送 Amazon SES 電子郵件、範本電子郵件和範本大量電子郵件，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

提供傳送電子郵件和驗證身分的許可。

```
"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

提供建立、取得、列出、更新和刪除 Amazon SES 電子郵件範本的許可。

```
"Statement": [  
  {  
    "Effect": "Allow",  
    "Action": [  
      "ses:CreateTemplate",  
      "ses:GetTemplate",  
      "ses:DeleteTemplate",  
      "ses:ListTemplates"  
    ]  
  }  
]
```

```

    "ses:ListTemplates",
    "ses:UpdateTemplate",
    "ses>DeleteTemplate",
    "ses:TestRenderTemplate"
  ],
  "Resource": "*"
}
]
```

SESSendBouncePolicy

將 SendBounce 許可授予 Amazon Simple Email Service (Amazon SES) 身分。

```

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ses:SendBounce"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:ses:${AWS::Region}:${AWS::AccountId}:identity/${identityName}",
        {
          "identityName": {
            "Ref": "IdentityName"
          }
        }
      ]
    }
  }
]
```

SNSCrudPolicy

提供建立、發佈和訂閱 Amazon SNS 主題的許可。

```

"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

准許將訊息發佈至 Amazon Simple Notification Service (Amazon SNS) 主題。

```

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "sns:Publish"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:sns:${AWS::Region}:${AWS::AccountId}:${topicName}",
        {
          "topicName": {
            "Ref": "TopicName"
          }
        }
      ]
    }
  }
]

```

SQSPollerPolicy

提供輪詢 Amazon Simple Queue Service (Amazon SQS) 佇列的許可。

```

"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

提供將訊息傳送至 Amazon SQS 佇列的許可。

```

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "sqs:SendMessage*"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:sqs:${AWS::Region}:${AWS::AccountId}:${queueName}",
        {
          "queueName": {
            "Ref": "QueueName"
          }
        }
      ]
    }
  }
]

```

```

    }
  }
]
```

SSMParameterReadPolicy

准許從 Amazon EC2 Systems Manager (SSM) 參數存放區存取參數，以在此帳戶中載入秘密。當參數名稱沒有斜線字首時，請使用。

Note

如果您不是使用預設金鑰，則還需要 KMSDecryptPolicy 政策。

```

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ssm:DescribeParameters"
    ],
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "ssm:GetParameters",
      "ssm:GetParameter",
      "ssm:GetParametersByPath"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:ssm:${AWS::Region}:${AWS::AccountId}:parameter/${parameterName}",
        {
          "parameterName": {
            "Ref": "ParameterName"
          }
        }
      ]
    }
  }
]
```

SSMParameterWithSlashPrefixReadPolicy

准許從 Amazon EC2 Systems Manager (SSM) 參數存放區存取參數，以在此帳戶中載入秘密。當參數名稱具有斜線字首時，請使用。

Note

如果您不是使用預設金鑰，則還需要 `KMSDecryptPolicy` 政策。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ssm:DescribeParameters"
    ],
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "ssm:GetParameters",
      "ssm:GetParameter",
      "ssm:GetParametersByPath"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:ssm:${AWS::Region}:${AWS::AccountId}:parameter:${parameterName}",
        {
          "parameterName": {
            "Ref": "ParameterName"
          }
        }
      ]
    }
  }
]
```

StepFunctionsExecutionPolicy

提供啟動 Step Functions 狀態機器執行的許可。


```

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "states:StartExecution"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:states:${AWS::Region}:${AWS::AccountId}:stateMachine:${stateMachineName}",
        {
          "stateMachineName": {
            "Ref": "StateMachineName"
          }
        }
      ]
    }
  }
]

```

TextractDetectAnalyzePolicy

提供使用 Amazon Textract 偵測和分析文件的存取權。

```

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "textract:DetectDocumentText",
      "textract:StartDocumentTextDetection",
      "textract:StartDocumentAnalysis",
      "textract:AnalyzeDocument"
    ],
    "Resource": "*"
  }
]

```

TextractGetResultPolicy

提供從 Amazon Textract 取得偵測和分析文件的存取權。

```

"Statement": [

```

```
{
  "Effect": "Allow",
  "Action": [
    "textract:GetDocumentTextDetection",
    "textract:GetDocumentAnalysis"
  ],
  "Resource": "*"
}
```

TextractPolicy

提供 Amazon Textract 的完整存取權。

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "textract:*"
    ],
    "Resource": "*"
  }
]
```

VPCAccessPolicy

提供建立、刪除、描述和分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 CloudFormation 機制管理 AWS SAM 許可

若要控制對 AWS 資源的存取，AWS Serverless Application Model (AWS SAM) 可以使用與相同的機制 CloudFormation。如需詳細資訊，請參閱《AWS CloudFormation 使用者指南》中的[使用 AWS Identity and Access Management 控制存取權限](#)。

授予使用者管理無伺服器應用程式的許可有三個主要選項。每個選項都會為使用者提供不同層級的存取控制。

- 授予管理員許可。
- 連接必要的 AWS 受管政策。
- 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他們有權存取之 AWS 資源的無伺服器應用程式。

以下各節會更詳細地描述每個選項。

授予管理員許可

如果您將管理員許可授予使用者，他們可以管理包含任意 AWS 資源組合的無伺服器應用程式。這是最簡單的選項，但也會授予使用者最廣泛的許可集，因此可讓使用者執行具有最高影響的動作。

如需將管理員許可授予使用者的詳細資訊，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](#)。

連接必要的 AWS 受管政策

您可以使用 [AWS 受管政策](#) 授予使用者一部分的許可，而不是授予完整的管理員許可。如果您使用此選項，請確定一組 AWS 受管政策涵蓋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 受管政策足以[部署範例 Hello World 應用程式](#)：

- AWSCloudFormationFullAccess
- IAMFullAccess
- AWSLambda_FullAccess
- AmazonAPIGatewayAdministrator
- AmazonS3FullAccess
- AmazonEC2ContainerRegistryFullAccess

如需將政策連接至 IAM 使用者的資訊，請參閱 [《IAM 使用者指南》中的變更 IAM 使用者的許可](#)。

授予特定 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 IAM 許可授予使用者。如果您使用此選項，請確定政策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。

使用此選項的最佳實務是拒絕使用者建立角色的許可，包括 Lambda 執行角色，以便他們無法授予自己提升的許可。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 執行角色的資訊，請參閱 [IAM 主控台](#)中的[建立執行角色](#)。

對於範例 [Hello World 應用程式](#)，AWSLambdaBasicExecutionRole 足以執行應用程式。建立 Lambda 執行角色之後，請修改範例 Hello World 應用程式的 AWS SAM 範本檔案，將下列屬性新增至AWS::Serverless::Function資源：

```
Role: lambda-execution-role-arn
```

使用修改過的 Hello World 應用程式時，下列政策陳述式會授予足夠許可，讓使用者部署、更新和刪除應用程式：

Note

本節中的範例政策陳述式會授予足夠的許可，讓您部署、更新和刪除[範例 Hello World 應用程式](#)。如果您將其他資源類型新增至應用程式，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許可。
2. 服務主體，如果服務動作需要的話。

例如，如果您新增 Step Functions 工作流程，您可能需要新增[此處](#)所列動作的許可和服務states.amazonaws.com主體。

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

控制對 API Gateway APIs存取有助於確保您的無伺服器應用程式安全，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 APIs。

AWS SAM 支援多種機制來控制對 API Gateway APIs 存取。支援的機制集在 `AWS::Serverless::HttpApi` 和 `AWS::Serverless::Api` 資源類型之間不同。

下表摘要說明每個資源類型支援的機制。

控制存取的機制	<code>AWS::Serverless::HttpApi</code>	<code>AWS::Serverless::Api</code>
Lambda 授權方	✓	✓
IAM 許可		✓
Amazon Cognito 使用者集區	✓ *	✓
API 金鑰		✓
資源政策		✓
OAuth 2.0/JWT 授權方	✓	

* 您可以使用 Amazon Cognito 做為具有 `AWS::Serverless::HttpApi` 資源類型的 JSON Web Token (JWT) 發行者。

- Lambda 授權方 – Lambda 授權方 (先前稱為自訂授權方) 是您提供的 Lambda 事件函數，用於控制對 API 的存取。呼叫 API 時，會使用用戶端應用程式提供的請求內容或授權字串叫用此 Lambda 函數。Lambda 函數會回應發起人是否有權執行請求的操作。

`AWS::Serverless::HttpApi` 和 `AWS::Serverless::Api` 資源類型都支援 Lambda 授權方。

如需使用的 Lambda 授權方詳細資訊 `AWS::Serverless::HttpApi`，請參閱《API Gateway 開發人員指南》中的 [使用 HTTP APIs 的授權 AWS Lambda 方](#)。如需搭配使用 Lambda 授權方的詳細資訊 `AWS::Serverless::Api`，請參閱《[API Gateway 開發人員指南](#)》中的 [使用 API Gateway Lambda 授權方](#)。

如需任一資源類型的 Lambda 授權方範例，請參閱 [的 Lambda 授權方範例 AWS SAM](#)。

- IAM 許可 – 您可以使用 [AWS Identity and Access Management \(IAM\) 許可](#) 來控制誰可以叫用您的 API。呼叫 API 的使用者必須使用 IAM 登入資料進行身分驗證。只有在連接至 IAM 使用者的 IAM 政策代表 API 發起人、包含使用者的 IAM 群組，或使用者擔任的 IAM 角色時，呼叫 API 才會成功。

只有 `AWS::Serverless::Api` 資源類型支援 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 許可控制對 API 的存取](#)。如需範例，請參閱 [的 IAM 許可範例 AWS SAM](#)。

- 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 的用戶端必須先登入使用者集區，並取得使用者的身分或存取權杖。然後，用戶端會使用其中一個傳回的字符來呼叫您的 API。只有在所需的字符有效時，API 呼叫才會成功。

AWS::Serverless::Api 資源類型支援 Amazon Cognito 使用者集區。AWS::Serverless::HttpApi 資源類型支援使用 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 [的 Amazon Cognito 使用者集區範例 AWS SAM](#)。

- 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 的存取權。

只有 AWS::Serverless::Api 資源類型支援 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[使用 API 金鑰建立和使用用量計劃](#)。如需 API 金鑰的範例，請參閱 [的 API 金鑰範例 AWS SAM](#)。

- 資源政策 – 資源政策是您可以連接到 API Gateway API 的 JSON 政策文件。使用資源政策來控制指定的委託人（通常是 IAM 使用者或角色）是否可以叫用 API。

只有 AWS::Serverless::Api 資源類型支援資源政策做為控制 API Gateway APIs 存取的機制。

如需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 資源政策控制對 API 的存取](#)。如需資源政策的範例，請參閱 [的資源政策範例 AWS SAM](#)。

- OAuth 2.0/JWT 授權方 – 您可以使用 JWTs 做為 [OpenID Connect \(OIDC\)](#) 和 [OAuth 2.0](#) 架構的一部分，以控制對 APIs 存取。API Gateway 會驗證用戶端隨 API 請求提交 JWTs，並根據權杖驗證以及可選擇權杖中的範圍來允許或拒絕請求。

只有 AWS::Serverless::HttpApi 資源類型支援 OAuth 2.0/JWT 授權方。

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 [的 OAuth 2.0/JWT 授權方範例 AWS SAM](#)。

選擇控制存取的機制

您選擇用於控制 API Gateway APIs 存取的機制取決於幾個因素。例如，如果您有一個未設定授權或存取控制的綠地專案，則 Amazon Cognito 使用者集區可能是您的最佳選項。這是因為當您設定使用者集區時，也會自動設定身分驗證和存取控制。

不過，如果您的應用程式已設定身分驗證，則使用 Lambda 授權方可能是您的最佳選項。這是因為您可以呼叫現有的身分驗證服務，並根據回應傳回政策文件。此外，如果您的應用程式需要使用者集區不支援的自訂身分驗證或存取控制邏輯，則 Lambda 授權方可能是您的最佳選項。

選擇要使用的機制時，請參閱 中的對應章節，[範例](#) 了解如何使用 AWS SAM 設定您的應用程式以使用該機制。

自訂錯誤回應

您可以使用 AWS SAM 來自訂某些 API Gateway 錯誤回應的內容。只有 `AWS::Serverless::Api` 資源類型支援自訂 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 [的自訂回應範例 AWS SAM](#)。

範例

- [的 Lambda 授權方範例 AWS SAM](#)
- [的 IAM 許可範例 AWS SAM](#)
- [的 Amazon Cognito 使用者集區範例 AWS SAM](#)
- [的 API 金鑰範例 AWS SAM](#)
- [的資源政策範例 AWS SAM](#)
- [的 OAuth 2.0/JWT 授權方範例 AWS SAM](#)
- [的自訂回應範例 AWS SAM](#)

的 Lambda 授權方範例 AWS SAM

`AWS::Serverless::Api` 資源類型支援兩種類型的 Lambda 授權方：TOKEN 授權方和 REQUEST 授權方。`AWS::Serverless::HttpApi` 資源類型僅支援 REQUEST 授權方。以下是每種類型的範例。

Lambda **TOKEN**授權方範例 (AWS::Serverless::Api)

您可以在 AWS SAM 範本中定義 Lambda TOKEN授權方，以控制對 APIs存取。若要這樣做，請使用 [ApiAuth](#) 資料類型。

以下是 Lambda TOKEN授權方的範例 AWS SAM 範本區段：

Note

在下列範例中，SAM FunctionRole是隱含產生的。

```
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 Lambda 授權方的詳細資訊，請參閱《[API Gateway 開發人員指南](#)》中的使用 [API Gateway Lambda 授權方](#)。

Lambda **REQUEST** 授權方範例 (AWS::Serverless::Api)

您可以在 AWS SAM 範本中定義 Lambda REQUEST 授權方，以控制對 APIs 存取。若要這樣做，請使用 [ApiAuth](#) 資料類型。

以下是 Lambda REQUEST 授權方的範例 AWS SAM 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 Lambda 授權方的詳細資訊，請參閱《[API Gateway 開發人員指南](#)》中的[使用 API Gateway Lambda 授權方](#)。

Lambda 授權方範例 (AWS::Serverless::HttpApi)

您可以在 AWS SAM 範本中定義 Lambda 授權方，以控制對 HTTP APIs 存取。若要這樣做，請使用 [HttpApiAuth](#) 資料類型。

以下是 Lambda 授權方的範例 AWS SAM 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

```

的 IAM 許可範例 AWS SAM

您可以在範本中定義 IAM 許可，以控制對 AWS SAM APIs 存取。若要這樣做，請使用 [ApiAuth](#) 資料類型。

以下是 用於 IAM 許可的範例 AWS SAM 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。

的 Amazon Cognito 使用者集區範例 AWS SAM

您可以在範本中定義 Amazon Cognito 使用者集區，以控制對 AWS SAM APIs 的存取。若要這樣做，請使用 [ApiAuth](#) 資料類型。

以下是使用者集區的範例 AWS SAM 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 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方控制對 REST API 的存取](#)。

的 API 金鑰範例 AWS SAM

您可以在 AWS SAM 範本中要求 API 金鑰，以控制對 API APIs 存取。若要這樣做，請使用 [ApiAuth](#) 資料類型。

以下是 API 金鑰的範例 AWS SAM 範本區段：

```

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[使用 API 金鑰建立和使用用量計劃](#)。

的資源政策範例 AWS SAM

您可以在範本中附加資源政策，以控制對 AWS SAM APIs 存取。若要這樣做，請使用 [ApiAuth](#) 資料類型。

以下是私有 API 的範例 AWS SAM 範本。私有 API 必須具有要部署的資源政策。

```
AWS::Serverless::Api
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: '*'
            - Effect: 'Deny'
              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 資源政策控制對 API 的存取](#)。如需私有 APIs 的詳細資訊，請參閱《[API Gateway 開發人員指南](#)》中的在 [Amazon API Gateway 中建立私有 API](#)。

的 OAuth 2.0/JWT 授權方範例 AWS SAM

您可以在 [OpenID Connect \(OIDC\)](#) 和 OAuth 2.0 架構中使用 JWTs 控制對 APIs 的存取。[OAuth](#) 若要這樣做，請使用 [HttpApiAuth](#) 資料類型。

以下是 OAuth 2.0/JWT 授權方的範例 AWS SAM 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 授權方的詳細資訊，請參閱《[APIs Gateway 開發人員指南](#)》中的[使用 JWT 授權方控制 HTTP API 的存取](#)。

的自訂回應範例 AWS SAM

您可以在 AWS SAM 範本中定義回應標頭，以自訂一些 API Gateway 錯誤回應。若要這樣做，您可以使用[閘道回應物件](#)資料類型。

以下是為DEFAULT_5XX錯誤建立自訂回應的範例 AWS SAM 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: "'*'"
              ErrorHandler: "'MyCustomErrorHandler'"
          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。

透過 使用 Lambda 層提高效率 AWS SAM

使用 AWS SAM，您可以在無伺服器應用程式中包含圖層。AWS Lambda 圖層可讓您從 Lambda 事件函數擷取程式碼到 Lambda 圖層，然後可用於多個 Lambda 函數。這樣做可讓您減少部署套件的大小、將核心函數邏輯與相依性分開，以及跨多個函數共用相依性。如需層的詳細資訊，請參閱《AWS Lambda 開發人員指南》中的 [Lambda 層](#)。

本主題提供有關下列項目的資訊：

- 在應用程式中包含圖層
- 如何在本機快取層

如需建置自訂 layer 的資訊，請參閱 [在中建置 Lambda 層 AWS SAM](#)。

在應用程式中包含圖層

若要在您的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 layer 的 Lambda 函數的範例 AWS SAM 範本：

```
ServerlessFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: .
    Handler: my_handler
    Runtime: Python3.7
    Layers:
      - <LayerVersion ARN>
```

如何在本機快取層

當您使用其中一個 `sam local` 命令叫用函數時，函數的 layer 套件會下載並在本機主機上快取。

下表顯示不同作業系統的預設快取目錄位置。

作業系統	Location
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，AWS SAM CLI 會將圖層疊加到用來叫用函數的 Docker 映像上。AWS SAM CLI 會產生其建置的映像名稱，以及快取中保留的 LayerVersions。您可以在以下章節中找到結構描述的詳細資訊。

若要檢查重疊圖層，請執行下列命令，在您要檢查的映像中啟動 bash 工作階段：

```
docker run -it --entrypoint=/bin/bash samcli/lambda:<Tag following the schema outlined in Docker Image Tag Schema> -i
```

Layer 快取目錄名稱結構描述

假設範本中定義的 LayerVersionArn，AWS SAM CLI 會從 ARN 擷取 LayerName 和版本。它會建立目錄，將 layer 內容放置在名為 `<LayerName>-Version-<first 10 characters of sha256 of ARN>` 的目錄中。

範例：

```
ARN = arn:aws:lambda:us-west-2:111111111111:layer:myLayer:1
Directory name = myLayer-1-926eeb5ff1
```

Docker Image 標籤結構描述

若要計算唯一圖層雜湊，請將所有唯一圖層名稱與 '-' 分隔符號結合，採用 SHA256 雜湊，然後採用前 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
```

唯一名稱的運算方式與 Layer Caching Directory 名稱結構描述相同：

```
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
```

若要計算唯一圖層雜湊，請將所有唯一圖層名稱與 '-' 分隔符號結合，採用 sha256 雜湊，然後採用前 25 個字元：

```
myLayer-1-926eeb5ff1-mySecondLayer-1-6bc1022bdf = 2dd7ac5ffb30d515926aef
```

然後將此值與函數的執行時間和架構結合，並以 '-' 為分隔符號：

```
python3.7-x86_64-2dd7ac5ffb30d515926aefffd
```

在 中使用巢狀應用程式重複使用程式碼和資源 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成品或較大應用程式的元件。巢狀應用程式可讓您將常用程式碼轉換為自己的應用程式，然後可在較大的無伺服器應用程式或多個無伺服器應用程式間重複使用。

隨著無伺服器架構的成長，常見模式通常會出現，其中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個別 AWS SAM 範本中重複使用常見程式碼、功能、資源和組態，只允許您從單一來源維護程式碼。這可減少重複的程式碼和組態。此外，此模組化方法可簡化開發、增強程式碼組織，並促進無伺服器應用程式的一致性。透過巢狀應用程式，您可以更專注於應用程式獨有的商業邏輯。

若要在無伺服器應用程式中定義巢狀應用程式，請使用 [AWS::Serverless::Application](#) 資源類型。

您可以從下列兩個來源定義巢狀應用程式：

- AWS Serverless Application Repository 應用程式 – 您可以使用 中可供您帳戶使用的應用程式來定義巢狀應用程式 AWS Serverless Application Repository。這些可以是您帳戶中的私有應用程式、與您的帳戶私有共用的應用程式，或在 中公開發共用的應用程式 AWS Serverless Application Repository。如需不同部署許可層級的詳細資訊，請參閱《AWS Serverless Application Repository 開發人員指南》中的[應用程式部署許可](#)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。

如需如何使用 在無伺服器應用程式中定義這兩種巢狀應用程式的詳細資訊 AWS SAM，請參閱下列各節。

Note

可在無伺服器應用程式中巢狀化的應用程式數量上限為 200。
巢狀應用程式可以擁有的參數數目上限為 60 個。

從 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 存放和分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。

若要在無伺服器應用程式 AWS SAM 範本的 AWS Serverless Application Repository 中定義託管的應用程式，請使用每個 AWS Serverless Application Repository 應用程式詳細資訊頁面上的複製為 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 管理主控台。
2. AWS Serverless Application Repository 使用《AWS Serverless Application Repository 開發人員指南》的[瀏覽、搜尋和部署應用程式](#)一節中的步驟，尋找您要在 中巢狀的應用程式。
3. 選擇複製為 SAM 資源按鈕。您正在檢視之應用程式的 SAM 範本區段現在位於剪貼簿中。
4. 將 SAM 範本區段貼到您要在此應用程式中巢狀化之應用程式的 SAM 範本檔案Resources: 區段。

以下是 中託管之巢狀應用程式的 SAM 範本範例區段 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:` 區段。

Important

包含託管在 中的巢狀應用程式的應用程式會 AWS Serverless Application Repository 繼承巢狀應用程式的共用限制。

例如，假設應用程式是公開共用的，但它包含的巢狀應用程式僅與建立父應用程式 AWS 的帳戶私下共用。在此情況下，如果 AWS 您的帳戶沒有部署巢狀應用程式的許可，您就無法部署父應用程式。如需部署應用程式許可的詳細資訊，請參閱《AWS Serverless Application Repository 開發人員指南》中的[應用程式部署許可](#)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範例 SAM 範本區段：

```
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:` 區段。

部署巢狀應用程式

您可以使用命令 部署巢狀應用程式 AWS SAM CLI `sam deploy`。如需詳細資訊，請參閱[使用 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，您必須確認其包含巢狀應用程式。您可以傳遞 `CAPABILITY_AUTO_EXPAND` 至 [CreateCloudFormationChangeSet API](#)，或使用 [aws serverlessrepo create-cloud-formation-change-set](#) AWS CLI 命令來執行此操作。

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[在部署應用程式時確認 IAM 角色、資源政策和巢狀應用程式](#)。

在 中使用 EventBridge 排程器管理以時間為基礎的事件 AWS SAM

本主題中的內容提供有關什麼是 Amazon EventBridge 排程器、支援 AWS SAM 提供什麼、如何建立排程器事件，以及在建立排程器事件時可以參考的範例的詳細資訊。

什麼是 Amazon EventBridge 排程器？

使用 EventBridge 排程器來排程 AWS SAM 範本中的事件。Amazon EventBridge Scheduler 是一種排程服務，可讓您在所有 AWS 服務中建立、啟動和管理數百萬個事件和任務。此服務對於時間相關事件特別有用。您可以使用它來排程事件和週期性的時間型調用。它還支援一次性事件，以及具有開始和結束時間的速率和時間運算式。

若要進一步了解 Amazon EventBridge 排程器，請參閱 [EventBridge 排程器使用者指南中的什麼是 Amazon EventBridge 排程器？](#)。EventBridge

主題

- [中的 EventBridge 排程器支援 AWS SAM](#)
- [在 中建立 EventBridge 排程器事件 AWS SAM](#)
- [範例](#)
- [進一步了解](#)

中的 EventBridge 排程器支援 AWS SAM

AWS Serverless Application Model (AWS SAM) 範本規格提供簡單的速記語法，您可以使用 AWS Lambda 和 的 EventBridge 排程器來排程事件 AWS Step Functions。

在 中建立 EventBridge 排程器事件 AWS SAM

將 ScheduleV2 屬性設定為 AWS SAM 範本中的事件類型，以定義 EventBridge 排程器事件。此屬性支援 `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)。如需無效字母佇列的詳細資訊，請參閱《[EventBridge 排程器使用者指南](#)》中的[為 EventBridge 排程器設定無效字母佇列](#)。

EventBridge

指定 DLQ ARN 時，AWS SAM 會設定排程器排程的許可，以傳送訊息至 DLQ。未指定 DLQ ARN 時，AWS SAM 會建立 DLQ 資源。

範例

使用 定義 EventBridge 排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:
    Schedule:
      Type: ScheduleV2
      Properties:
        ScheduleExpression: rate(1 minute)
        Input: '{"hello": "simple"}'
```

進一步了解

若要進一步了解如何定義 ScheduleV2 EventBridge 排程器屬性，請參閱：

- [ScheduleV2](#) 適用於 `AWS::Serverless::Function`。
- [ScheduleV2](#) 適用於 `AWS::Serverless::StateMachine`。

使用 協調 AWS SAM 資源 AWS Step Functions

您可以使用 [AWS Step Functions](#) 來協調 AWS Lambda 事件函數和其他 AWS 資源，以形成複雜且強大的工作流程。Step Functions 可告知應用程式使用 AWS 資源，例如 AWS Lambda 函式的時機和條件。這可簡化形成複雜且強大工作流程的程序。使用 [AWS::Serverless::StateMachine](#)，您可以定義工作流程中的個別步驟、將每個步驟中的資源建立關聯，然後將這些步驟排序在一起。您也可以需要在需要轉換和條件的地方新增轉換和條件。這可簡化建立複雜且強大工作流程的程序。

Note

若要管理包含 Step Functions 狀態機器的 AWS SAM 範本，您必須使用 0.52.0 版或更新版本 AWS SAM CLI。若要檢查您擁有的版本，請執行命令 `sam --version`。

Step Functions 是以[任務](#)和[狀態機器](#)的概念為基礎。您可以使用 JSON 型 [Amazon States Language](#) 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，讓您可以以視覺化方式檢查狀態機器的邏輯並監控執行。

使用 Step Functions 支援 in AWS Serverless Application Model (AWS SAM)，您可以執行下列動作：

- 直接在 AWS SAM 範本內或在個別檔案中定義狀態機器
- 透過 AWS SAM 政策範本、內嵌政策或受管政策建立狀態機器執行角色
- 使用 API Gateway 或 Amazon EventBridge 事件、依照 AWS SAM 範本中的排程，或直接呼叫 APIs 來觸發狀態機器執行
- 針對常見的 Step Functions 開發模式使用可用的[AWS SAM 政策範本](#)。

範例

下列 AWS SAM 範本檔案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 States Language](#) 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》中的[使用 建立 Step Functions 狀態機器 AWS SAM](#)。

其他資訊

若要進一步了解 Step Functions 並搭配使用 AWS SAM，請參閱以下內容：

- [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 AWS SAM 應用程式設定程式碼簽署

為了確保只部署信任的程式碼，您可以使用 AWS SAM 來啟用無伺服器應用程式的程式碼簽署。簽署程式碼有助於確保程式碼在簽署後沒有遭到更改，而且只有來自信任發佈者的簽署程式碼套件才會在您的 Lambda 事件函數中執行。這有助於讓組織免於在其部署管道中建置 gatekeeper 元件的負擔。

如需程式碼簽署的詳細資訊，請參閱《AWS Lambda 開發人員指南》中的[設定 Lambda 函數的程式碼簽署](#)。

您必須先使用 Signer 建立簽署設定檔，才能為無伺服器應用程式設定程式碼 AWS 簽署。您可以針對下列任務使用此簽署描述檔：

1. 建立程式碼簽署組態 – 宣告 [AWS::Lambda::CodeSigningConfig](#) 資源以指定信任發佈者的簽署設定檔，並設定驗證檢查的政策動作。您可以在與無伺服器函數相同的 AWS SAM 範本、不同 AWS SAM 範本或 CloudFormation 範本中宣告此物件。然後，您可以使用 [AWS::Lambda::CodeSigningConfig](#) 資源的 Amazon Resource Name (ARN) 指定函數的 [CodeSigningConfigArn](#) 屬性，為無伺服器函數啟程式碼簽署。
2. 簽署您的程式碼 – 使用 [sam package](#) 或 [sam deploy](#) 命令搭配 `--signing-profiles` 選項。

Note

為了使用 `sam package` 或 `sam deploy` 命令成功簽署程式碼，您必須為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 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 Simple Storage Service 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 會對您已啟程式碼簽署的所有函數執行驗證檢查。Lambda 也會對這些函數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[簽章驗證](#)。

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，您會看到簽章設定檔的 ARN 傳回。例如：

```
{
  "arn": "arn:aws:signer:us-east-1:111122223333:/signing-profiles/MySigningProfile",
  "profileVersion": "SAMPLEverx",
  "profileVersionArn": "arn:aws:signer:us-east-1:111122223333:/signing-
profiles/MySigningProfile/SAMPLEverx"
}
```

profileVersionArn 欄位包含當您建立程式碼簽署組態時要使用的 ARN。

建立程式碼簽署組態並啟用函數的程式碼簽署

下列範例 AWS SAM 範本會宣告 [AWS::Lambda::CodeSigningConfig](#) 資源，並啟用 Lambda 函數的程式碼簽署。在此範例中，有一個信任的設定檔，而且如果簽章檢查失敗，部署會遭到拒絕。

```
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 `aws sam package` 或 `aws sam deploy` 命令指定 `--signing-profiles` 選項，如下列範例命令所示。

在封裝應用程式時簽署函數程式碼：

```
sam package --signing-profiles HelloWorld=MySigningProfile --s3-bucket amzn-s3-demo-bucket --output-template-file packaged.yaml
```

在封裝應用程式時，同時簽署函數程式碼和函數依賴的圖層：

```
sam package --signing-profiles HelloWorld=MySigningProfile MyLayer=MySigningProfile --s3-bucket amzn-s3-demo-bucket --output-template-file packaged.yaml
```

簽署函數程式碼和 layer，然後執行部署：

```
sam deploy --signing-profiles HelloWorld=MySigningProfile MyLayer=MySigningProfile --s3-bucket amzn-s3-demo-bucket --template-file packaged.yaml --stack-name --region us-east-1 --capabilities CAPABILITY_IAM
```

Note

為了使用 `sam package` 或 `sam deploy` 命令成功簽署程式碼，您必須為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 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 Simple Storage Service 使用者指南》中的 [在 Amazon S3 儲存貯體中使用版本控制](#)。

使用 提供簽署設定檔 `sam deploy --guided`

當您使用以程式碼簽署設定的無伺服器應用程式執行 `sam deploy --guided` 命令時，會 AWS SAM 提示您提供用於程式碼簽署的簽署設定檔。如需 `sam deploy --guided` 提示的詳細資訊，請參閱命令參考 [sam deploy](#) 中的 AWS SAM CLI。

驗證 AWS SAM 範本檔案

使用 驗證您的範本 [sam validate](#)。目前，此命令會驗證提供的範本是否為有效的 JSON/YAML。與大多數 AWS SAM CLI 命令一樣，它預設會在您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

sam validate 命令需要設定 AWS 登入資料。如需詳細資訊，請參閱[設定 AWS SAM CLI](#)。

使用 建置您的應用程式 AWS SAM

將基礎設施新增為程式碼 (IaC) 到 AWS SAM 範本後，您就可以使用 sam build 命令開始建置應用程式。此命令會從應用程式專案目錄中的檔案（即您的 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）建立建置成品。這些建置成品可讓您的無伺服器應用程式準備好進行應用程式的後續開發步驟，例如本機測試和部署至 AWS 雲端。測試和部署都會使用建置成品做為輸入。

您可以使用 sam build 來建置整個無伺服器應用程式。此外，您可以建立自訂組建，例如具有特定函數、layer 或自訂執行時間的組建。若要進一步了解如何使用和為什麼使用 sam build，請參閱本節中的主題。如需使用 sam build 命令的簡介，請參閱 [使用 建置 簡介 AWS SAM](#)。

主題

- [使用 建置 簡介 AWS SAM](#)
- [使用的預設建置 AWS SAM](#)
- [使用 自訂組建 AWS SAM](#)

使用 建置 簡介 AWS SAM

使用 AWS Serverless Application Model Command Line Interface (AWS SAM CLI) sam build 命令，為您的無伺服器應用程式準備開發工作流程中的後續步驟，例如本機測試或部署到 AWS 雲端。此命令會建立 .aws-sam 目錄，以和 sam deploy 所需的格式 sam local 和位置來建構您的應用程式。

- 如需的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)。
- 如需 sam build 命令選項的清單，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 `sam build` 需要您從開發機器上無伺服器應用程式的基本元件開始。這包括 AWS SAM 範本、AWS Lambda 事件函數程式碼，以及任何語言特定的檔案和相依性。如需詳細資訊，請參閱 [在 中建立您的應用程式 AWS SAM](#)。

主題

- [使用 `sam build` 建置應用程式](#)
- [本機測試和部署](#)
- [最佳實務](#)
- [sam 建置的選項](#)
- [疑難排解](#)
- [範例](#)
- [進一步了解](#)

使用 `sam build` 建置應用程式

使用 之前 `sam build`，請考慮設定下列項目：

1. Lambda 函數和 layer – `sam build` 命令可以建置 Lambda 函數和 layer。若要進一步了解 Lambda 層，請參閱 [在 中建置 Lambda 層 AWS SAM](#)。
2. Lambda 執行時間 – 執行時間提供特定語言的環境，可在叫用時於執行環境中執行函數。您可以設定原生和自訂執行時間。
 - a. 原生執行時間 – 在支援的 Lambda 執行時間中編寫 Lambda 函數，並建置函數以在 中使用原生 Lambda 執行時間 AWS 雲端。
 - b. 自訂執行時間 – 使用任何程式設計語言撰寫 Lambda 函數，並使用 `makefile` 或第三方建置器中定義的自訂程序建置執行時間，例如 `esbuild`。如需詳細資訊，請參閱 [在 中使用自訂執行時間建置 Lambda 事件函數 AWS SAM](#)。
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、執行時間、Lambda 延伸模組、您的應用程式程式碼及其相依性。

在使用 初始化應用程式時，可以設定這些應用程式設定 `sam init`。

- 若要進一步了解如何使用 `sam init`，請參閱 [在中建立您的應用程式 AWS SAM](#)。
- 若要進一步了解如何在應用程式中設定這些設定，請參閱 [使用的預設建置 AWS SAM](#)。

建置應用程式

1. `cd` 您專案的根目錄。這與 AWS SAM 範本的位置相同。

```
$ 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`。如需詳細資訊，請參閱 [在提供的容器內建置 Lambda 事件函數](#)。

以下是輸出的範例 AWS SAM CLI：

```
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. AWS SAM CLI 會建立 .aws-sam 建置目錄。以下是範例：

```

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

```

根據您的應用程式設定方式，AWS SAM CLI 會執行下列動作：

1. 下載、安裝和組織 .aws-sam/build 目錄中的相依性。
2. 準備您的 Lambda 程式碼。這可能包括編譯程式碼、建立可執行的二進位檔，以及建置容器映像。
3. 將建置成品複製到 .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 套件類型，成品尚未壓縮，因此可用於本機測試。使用 AWS SAM CLI 時，會壓縮您的應用程式 `sam deploy`。
 - b. 對於容器映像套件類型，容器映像會在本機建立，並在 .aws-sam/build.toml 檔案中參考。
4. 將 AWS SAM 範本複製到 .aws-sam 目錄，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 目錄中建置成品的主要元件：

- 建置目錄 – 包含彼此獨立建構的 Lambda 函數和圖層。這會導致 .aws-sam/build 目錄中每個函數或 layer 的唯一結構。
- AWS SAM 範本 – 根據建置過程中的變更，以更新的值進行修改。
- build.toml 檔案 – 包含所用建置設定的組態檔案 AWS SAM CLI。

本機測試和部署

使用 執行本機測試`sam local`或使用 部署時`sam deploy`，AWS SAM CLI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中。如果符合這些條件，AWS SAM CLI會將此視為應用程式的根目錄。
2. 如果不符合這些條件，AWS SAM CLI會將 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變更，請執行 `sam build` 以在本機測試之前更新`.aws-sam`目錄。

最佳實務

- 請勿在 `.aws-sam/build` 目錄下編輯任何程式碼。請改為更新專案資料夾中的原始原始原始程式碼，並執行 `sam build` 以更新`.aws-sam/build`目錄。
- 當您修改原始檔案時，請執行 `sam build` 以更新`.aws-sam/build`目錄。
- 您可能希望 AWS SAM CLI參考專案的原始根目錄，而不是 `.aws-sam` 目錄，例如使用 進行開發和測試時`sam local`。刪除 `.aws-sam` 目錄中的 目錄或 `.aws-sam` 目錄中的 AWS SAM 範本，讓 將 AWS SAM CLI原始專案目錄辨識為根專案目錄。準備就緒時，請`sam build`再次執行 以建立`.aws-sam`目錄。
- 當您執行 時`sam build`，每次都會覆寫`.aws-sam/build`目錄。`.aws-sam` 目錄不會。如果您想要存放日誌等檔案，請將它們存放在 `.aws-sam`，以防止它們遭到覆寫。

sam 建置的選項

建置單一資源

提供資源的邏輯 ID，以僅建置該資源。以下是範例：

```
$ sam build HelloWorldFunction
```

若要建置巢狀應用程式或堆疊的資源，請使用 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
```

在提供的容器內建置 Lambda 事件函數

`--use-container` 選項會下載容器映像，並使用它來建置 Lambda 函數。然後，您的 `.aws-sam/build.toml` 檔案中會參考本機容器。

Docker 需要安裝此選項。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。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，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
```

使用 建置容器映像 BuildKit

使用 Image 套件類型建置 Lambda 函數時，您可以透過傳遞 `--use-buildkit` 選項來使用 BuildKit 做為建置引擎。BuildKit 提供比舊版建置引擎更好的建置效能、快取和增強型 Docker 建置輸出。

此選項需要在您的 中 FinchCLIs 安裝 Docker 或 和 PATH。

您可以 `--use-buildkit` 與其他建置選項結合。以下是範例：

```
$ sam build --use-buildkit --parallel
```

Note

`--use-buildkit` 旗標僅適用於容器映像組建。它不會對使用 建置的 zip 函數有任何影響 `--use-container`。

將環境變數傳遞至建置容器

使用 `--container-env-var` 將環境變數傳遞至建置容器。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` 檔案的範例：

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

加速建置包含多個 函數的應用程式

當您在具有多個函數的應用程式 `sam build` 上執行時，AWS SAM CLI 會一次建立一個函數。若要加速建置程序，請使用 `--parallel` 選項。這會同時建置所有函數和層。

以下是此命令的範例：

```
$ sam build --parallel
```

在來源資料夾中建置您的專案，以加快建置時間

對於支援的執行時間和建置方法，您可以使用 `--build-in-source` 選項直接在來源資料夾中建置專案。根據預設，會在暫存目錄中 AWS SAM CLI 建置，這涉及透過原始程式碼和專案檔案進行複製。使用 `--build-in-source` 時，AWS SAM 會直接在您的來源資料夾中 CLI 建置，而不需要將檔案複製到暫時目錄，即可加速建置程序。

如需支援的執行時間和建置方法清單，請參閱 [--build-in-source](#)。

疑難排解

若要對 進行故障診斷 AWS SAM CLI，請參閱 [AWS SAM CLI 故障診斷](#)。

範例

建置使用原生執行時間和 .zip 套件類型的應用程式

如需此範例，請參閱 [教學課程：使用 部署 Hello World 應用程式 AWS SAM](#)。

建置使用原生執行時間和映像套件類型的應用程式

首先，我們會執行 `sam init` 來初始化新的應用程式。在互動式流程中，我們會選取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

...

會 AWS SAM CLI 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，我們會執行 `sam build` 來建置應用程式：

```
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
```

建置包含編譯程式設計語言的應用程式

在此範例中，我們會建置應用程式，其中包含使用 Go 執行時間的 Lambda 函數。

首先，我們使用 初始化新的應用程式，`sam init`並將應用程式設定為使用 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

...

會 AWS SAM CLI 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

```

我們針對此應用程式的需求參考 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，我們會執行 `sam local invoke` 來測試函數。Go 因為未安裝在本機電腦上，所以此命令錯誤：

```
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，我們會執行 `sam build` 來建置應用程式。我們遇到錯誤，因為 Go 未安裝在本機電腦上：

```
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
```

雖然我們可以設定本機機器以正確建置函數，但我們改為將 `--use-container` 選項與 搭配使用 `sam build`。會 AWS SAM CLI 下載容器映像、使用原生 `GoModulesBuilder` 建置函數，並將產生的二進位檔複製到我們的 `.aws-sam/build/HelloWorldFunction` 目錄。

```
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 `.aws-sam` 目錄的範例：

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

接下來，我們會執行 `sam local invoke`。我們的函數已成功調用：

```
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

若要進一步了解如何使用 `sam build` 命令，請參閱以下內容：

- [學習 AWS SAM : sam build](#) – 上的 Serverless Land "Learning AWS SAM" 系列YouTube。
- [Learning AWS SAM | sam build | E3](#) – 上的 Serverless Land "Learning AWS SAM" 系列YouTube。
- [AWS SAM build : 它如何為部署提供成品 \(使用 SAM S2E8 的工作階段 \)](#) – 在 上使用 AWS SAM 序列的工作階段YouTube。
- [AWS SAM 自訂組建 : 如何使用 Makefiles 在 SAM \(S2E9\) 中自訂組建](#) – 在 上使用 AWS SAM 序列的工作階段YouTube。

使用 的預設建置 AWS SAM

若要建置無伺服器應用程式，請使用 `sam build` 命令。此命令也會收集應用程式相依性的建置成品，並將其置於適當的格式和位置，以供後續步驟使用，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`(Python) 或 `package.json`(Node.js)，或使用函數資源的 `Layers` 屬性。`Layers` 屬性包含 Lambda 事件函數所依賴的[AWS Lambda 層](#)資源清單。

應用程式建置成品的格式取決於每個函數的 `PackageType` 屬性。此屬性的選項為：

- **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 執行時間。
- **Image** – 容器映像，除了您的應用程式程式碼及其相依性之外，還包含基本作業系統、執行時間和擴充功能。

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

- [建置 .zip 檔案封存](#)
- [建置容器映像](#)
- [容器環境變數檔案](#)
- [在來源資料夾中建置您的專案，以加快建置時間](#)
- [範例](#)
- [在外部建置函數 AWS SAM](#)

建置 .zip 檔案封存

若要將無伺服器應用程式建置為 .zip 檔案封存，PackageType: Zip 請為無伺服器函數宣告。

AWS SAM 會為您指定的[架構](#)建置您的應用程式。如果您未指定架構，x86_64 預設 AWS SAM 會使用。

如果您的 Lambda 函數取決於具有原生編譯程式的套件，請使用 `--use-container` 旗標。此旗標會在類似 Lambda 環境的容器中本機編譯函數，因此當您將函數部署到 AWS 雲端時，它們的格式正確。

當您使用 `--use-container` 選項時，預設會從 [Amazon ECR Public](#) AWS SAM 提取容器映像。如果您想要從另一個儲存庫或特定版本的 CLI AWS SAM 提取容器映像，您可以使用 `--build-image` 選項並提供替代容器映像的 URI。以下是使用來自特定 CLI AWS SAM 版本之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 for a specific version of AWS SAM CLI (1.136.0)
sam build --use-container --build-image public.ecr.aws/sam/build-nodejs22.x:1.136.0

# Build a function resource using the Python 3.13 container image from a specific version of AWS SAM CLI (1.136.0)(
```

```
sam build --use-container --build-image Function1=public.ecr.aws/sam/build-python3.13:1.136.0
```

如需建置 .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。

建置容器映像

若要將無伺服器應用程式建置為容器映像，`PackageType: Image`請為無伺服器函數宣告。您還必須使用下列項目宣告Metadata資源屬性：

Dockerfile

與 Lambda 函數相關聯的 Dockerfile 名稱。

DockerContext

Dockerfile 的位置。

DockerTag

(選用) 要套用至建置映像的標籤。

DockerBuildArgs

建置組建的引數。

Important

AWS SAM CLI 不會修訂或混淆您在DockerBuildArgs引數中包含的任何資訊。我們強烈建議您不要使用此區段來儲存機密資訊，例如密碼或秘密。

DockerImage

(選用) 建置 Docker 映像的自訂名稱。如果未指定，則會使用函數的邏輯資源 ID。

以下是範例Metadata資源屬性區段：

```
Metadata:
  Dockerfile: Dockerfile
  DockerContext: ./hello_world
  DockerTag: v1
  DockerImage: my-custom-image
```

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：使用 部署 Hello World 應用程式 AWS SAM](#)。在詢問您要安裝之套件類型的提示中，選擇 Image。

Note

如果您在 Dockerfile 中指定多架構基礎映像，會為主機機器的架構 AWS SAM 建置容器映像。若要為不同的架構建置，請指定使用特定目標架構的基本映像。

根據預設，AWS SAM CLI 會在建置容器映像時使用舊版 Docker 建置引擎。若要使用 BuildKit，一種改善效能並處理更複雜建置的容器後端，請使用您的建置命令傳遞 `--use-buildkit` 旗標：

```
$ sam build --use-buildkit
```

使用 `--use-buildkit` 需要 Finch CLI 安裝 Docker 或。

容器環境變數檔案

若要提供包含建置容器環境變數的 JSON 檔案，請使用 `--container-env-var-file` 引數搭配 `sam build` 命令。您可以提供套用至所有無伺服器資源的單一環境變數，或每個資源的不同環境變數。

格式

將環境變數傳遞至建置容器的格式取決於您為資源提供的環境變數數量。

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，名為 `env.json`。下列命令使用此檔案將環境變數傳遞至建置容器：

```
sam build --use-container --container-env-var-file env.json
```

優先順序

- 您提供給特定資源的環境變數優先於所有資源的單一環境變數。
- 您在命令列提供的環境變數優先於檔案中的環境變數。

在來源資料夾中建置您的專案，以加快建置時間

對於支援的執行時間和建置方法，您可以使用 `--build-in-source` 選項直接在來源資料夾中建置專案。根據預設，會在暫存目錄中 AWS SAM CLI 建置，這涉及透過原始程式碼和專案檔案進行複製。使用 `--build-in-source` 時，AWS SAM 會直接在您的來源資料夾中 CLI 建置，而不需要將檔案複製到暫時目錄，即可加速建置程序。

如需支援的執行時間和建置方法清單，請參閱 [--build-in-source](#)。

範例

範例 1：.zip 檔案封存

下列 `sam build` 命令會建置 .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 for a specific version of AWS
SAM CLI (1.136.0)
sam build --use-container --build-image public.ecr.aws/sam/build-nodejs22.x:1.136.0
```



```
# Build a function resource using the Python 3.13 container image from a specific
version of AWS SAM CLI (1.136.0)(
sam build --use-container --build-image Function1=public.ecr.aws/sam/build-
python3.13:1.136.0

# Build and run your functions locally
sam build && sam local invoke

# For more options
sam build --help
```

範例 2：容器映像

下列 AWS SAM 範本建置為容器映像：

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      PackageType: Image
      ImageConfig:
        Command: ["app.lambda_handler"]
    Metadata:
      Dockerfile: Dockerfile
      DockerContext: ./hello_world
      DockerTag: v1
```

以下是 Dockerfile 範例：

```
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：npm ci

對於 Node.js 應用程式，您可以使用 `npm ci` 而不是 `npm install` 安裝相依性。若要使用 `npm ci`，請在 Lambda 函數 Metadata 的資源屬性 `BuildProperties` 中指定 `UseNpmCi: True` 下的。若要

使用 `npm ci`，您的應用程式必須在 Lambda 函數 `CodeUri` 中具有 `package-lock.json` 或 `npm-shrinkwrap.json` 檔案。

當您執行 `npm ci`，下列範例會使用安裝相依性 `npm ci`：

```
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
```

Python 父系套件

對於 Python 應用程式，您可以在建置過程中保留套件結構，以啟用絕對匯入。若要保留套件結構，請在 Lambda 函數 `Metadata` 的資源屬性 `BuildProperties` 中指定 `ParentPackageMode` 下的。

當您執行時，下列範例會保留 `app` 套件結構 `npm ci`：

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: hello-world/
      Handler: app.main.handler
      Runtime: python3.12
      Architectures:
        - x86_64
    Metadata:
      BuildProperties:
```

```
ParentPackageMode: explicit
ParentPackages: app
```

透過此組態，您的程式碼可以使用等絕對匯入 `from app.utils import logger`，而不是等相對匯入 `from .utils import logger`。

在 外部建置 函數 AWS SAM

根據預設，當您執行 `sam build`，會 AWS SAM 建置所有函數資源。其他選項包括：

- 在 外部建置所有函數資源 AWS SAM – 如果您手動或透過其他工具建置所有函數資源，`sam build`則不需要。您可以略過`sam build`並繼續進程序的下一個步驟，例如執行本機測試或部署應用程式。
- 在 外部建置一些函數資源 AWS SAM – 如果您想要在外部建置其他函數資源的同時 AWS SAM 建置一些函數資源 AWS SAM，您可以在 AWS SAM 範本中指定此選項。

在 外部建置一些函數資源 AWS SAM

若要在使用時 AWS SAM 略過函數`sam build`，請在 AWS SAM 範本中設定下列項目：

1. 將`SkipBuild: True`中繼資料屬性新增至函數。
2. 指定您建置函數資源的路徑。

以下是範例，其中 `TestFunction` 設定為略過。其建置的資源位於 `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，當您執行時`sam build`，AWS SAM 會執行下列動作：

1. AWS SAM 會略過使用設定的函數`SkipBuild: True`。
2. AWS SAM 會建置所有其他函數資源，並將其快取在 `.aws-sam` 建置目錄中。

3. 對於略過的函數，其在 `.aws-sam` 建置目錄中的範本會自動更新，以參考所建置函數資源的指定路徑。

以下是 `TestFunction.aws-sam` 建置目錄中的快取範本範例：

```
TestFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: ../../built-resources/TestFunction.zip
    Handler: TimeHandler::handleRequest
    Runtime: java11
  Metadata:
    SkipBuild: True
```

使用 自訂組建 AWS SAM

您可以自訂建置，以包含特定的 Lambda 事件函數或 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，然後在多個 Lambda 函數之間重複使用。當您想要專注於開發和部署個別的無伺服器函數，而無需管理共用相依性或資源的複雜性時，可以選擇使用特定 Lambda 函數自訂您的建置。此外，您可以選擇建置 Lambda 層，以協助您減少部署套件的大小、將核心函數邏輯與相依性分開，以及讓您在多個函數之間共用相依性。

本節中的主題會探索您可以使用的一些不同方式來建置 Lambda 函數 AWS SAM。這包括使用客戶執行時間建置 Lambda 函數，以及建置 Lambda 層。《AWS Lambda 開發人員指南》中的自訂執行階段可讓您安裝和使用 Lambda 執行階段中未列出的語言。這可讓您建立專用執行環境，以執行無伺服器函數和應用程式。僅建置 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。它可協助您減少部署套件的大小、將核心函數邏輯與相依性分開，並允許您跨多個函數共用相依性。

如需 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

- [在 中使用 esbuild 建置 Node.js Lambda 事件函數 AWS SAM](#)
- [在 中使用原生 AOT 編譯建置 .NET Lambda 事件函數 AWS SAM](#)
- [在 Cargo Lambda 中使用 建置 Rust Lambda 事件函數 AWS SAM](#)
- [在 uv 中使用 建置 Python Lambda 事件函數 AWS SAM](#)
- [在 中使用自訂執行時間建置 Lambda 事件函數 AWS SAM](#)

- [在中建置 Lambda 層 AWS SAM](#)

在中使用 esbuild 建置 Node.js Lambda 事件函數 AWS SAM

若要建置和封裝 Node.js AWS Lambda Event Functions，您可以使用 AWS SAM CLI 搭配 esbuild JavaScript bundler。esbuild bundler 支援您在 TypeScript 中寫入的 Lambda 函數。

若要使用 esbuild 建置 Node.js Lambda 函數，請將 Metadata 物件新增至您的 `AWS::Serverless::Function` 資源，並為指定 esbuild BuildMethod。當您執行 `sam build` 命令時，AWS SAM 會使用 esbuild 來綁定 Lambda 函數程式碼。

中繼資料屬性

Metadata 物件支援 esbuild 的下列屬性。

BuildMethod

為您的應用程式指定 Bundler。唯一支援的值為 esbuild。

BuildProperties

指定 Lambda 函數程式碼的建置屬性。

BuildProperties 物件支援 esbuild 的下列屬性。所有屬性都是選用的。根據預設，AWS SAM 會將 Lambda 函數處理常式用於進入點。

EntryPoint

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 網站的[外部](#)。

格式

指定應用程式中產生的 JavaScript 檔案的輸出格式。如需詳細資訊，請參閱 esbuild 網站上的[格式](#)。

載入器

指定用於載入指定檔案類型之資料的組態清單。

MainFields

指定解析套件時要嘗試匯入 `package.json` 的欄位。預設值為 `main,module`。

Minify

指定是否將綁定的輸出程式碼縮小。預設值為 `true`。

OutExtension

自訂 esbuild 產生的檔案副檔名。如需詳細資訊，請參閱 esbuild 網站上的 [Out 擴充](#) 功能。

來源地圖

指定 Bundler 是否產生來源映射檔案。預設值為 `false`。

設定為 `true`，`NODE_OPTIONS: --enable-source-maps` 會附加到 Lambda 函數的環境變數，並產生來源映射並包含在函數中。

或者，當 `NODE_OPTIONS: --enable-source-maps` 包含在函數的環境變數中時，`Sourcemap` 會自動設定為 `true`。

衝突時，`Sourcemap: false` 優先於 `NODE_OPTIONS: --enable-source-maps`。

Note

根據預設，Lambda 會使用 AWS Key Management Service () 加密所有靜態環境變數 AWS KMS。使用來源映射時，為了讓部署成功，函數的執行角色必須具有執行 `kms:Encrypt` 動作的許可。

SourcesContent

指定是否要在來源映射檔案中包含您的原始程式碼。當設定為 `true`，設定此屬性 `'true'`。

- 指定 `SourcesContent: 'true'` 以包含所有原始程式碼。
- 指定 `SourcesContent: 'false'` 以排除所有原始程式碼。這會導致較小的來源映射檔案大小，這可透過縮短啟動時間在生產環境中很有用。不過，偵錯工具中無法使用原始程式碼。

預設值為 `SourcesContent: true`。

如需詳細資訊，請參閱 esbuild 網站上的 [來源內容](#)。

Target

指定目標 ECMAScript 版本。預設值為 `es2020`。

TypeScript Lambda 事件函數範例

下列範例 AWS SAM 範本程式碼片段使用 esbuild 從 中的 TypeScript 程式碼建立 Node.js Lambda 函數hello-world/app.ts。

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: hello-world/
      Handler: app.handler
      Runtime: nodejs20.x
      Architectures:
        - x86_64
      Events:
        HelloWorld:
          Type: Api
          Properties:
            Path: /hello
            Method: get
      Environment:
        Variables:
          NODE_OPTIONS: --enable-source-maps
      Metadata:
        BuildMethod: esbuild
        BuildProperties:
          Format: esm
          Minify: false
          OutExtension:
            - .js=.mjs
          Target: "es2020"
          Sourcemap: true
          EntryPoints:
            - app.ts
      External:
        - "<package-to-exclude>"
```

在 中使用原生 AOT 編譯建置 .NET Lambda 事件函數 AWS SAM

使用 AWS Serverless Application Model (AWS SAM) 建置和封裝您的 .NET 8 AWS Lambda Event Functions，利用原生Ahead-of-Time(AOT) 編譯來改善 AWS Lambda 冷啟動時間。

主題

- [.NET 8 原生 AOT 概觀](#)
- [AWS SAM 搭配 .NET 8 Lambda 事件函數使用](#)
- [安裝先決條件](#)
- [在 AWS SAM 範本中定義 .NET 8 Lambda 事件函數](#)
- [使用 建置您的應用程式 AWS SAM CLI](#)
- [進一步了解](#)

.NET 8 原生 AOT 概觀

過去，.NET Lambda 函數的冷啟動時間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。透過 .NET 原生 AOT 編譯，您可以改善 Lambda 函數的冷啟動時間。若要進一步了解適用於 .NET 8 的原生 AOT，請參閱 Dotnet GitHub 儲存庫中的[使用原生 AOT](#)。

AWS SAM 搭配 .NET 8 Lambda 事件函數使用

執行下列動作以使用 () 設定 .NET 8 Lambda AWS Serverless Application Model 函數AWS SAM：

- 在開發機器上安裝先決條件。
- 在 AWS SAM 範本中定義 .NET 8 Lambda 函數。
- 使用 建置您的應用程式 AWS SAM CLI。

安裝先決條件

以下是必要的先決條件：

- 的 AWS SAM CLI
- .NET Core CLI
- Amazon.Lambda.Tools .NET Core 全球工具
- Docker

安裝 AWS SAM CLI

1. 若要檢查是否已安裝 AWS SAM CLI，請執行下列動作：

```
sam --version
```


- 若要安裝 AWS SAM CLI，請參閱 [安裝 AWS SAM CLI](#)。
- 若要升級已安裝的版本 AWS SAM CLI，請參閱 [升級 AWS SAM CLI](#)。

安裝 .NET Core CLI

- 若要下載並安裝 .NET Core CLI，請參閱從 Microsoft [網站下載 .NET](#)。
- 如需 .NET Core CLI 的詳細資訊，請參閱《AWS Lambda 開發人員指南》中的 [.NET Core CLI](#)。

安裝 Amazon.Lambda.Tools .NET Core Global Tool

- 執行以下命令：

```
dotnet tool install -g Amazon.Lambda.Tools
```

- 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
```

- 如需有關 Amazon.Lambda.Tools .NET Core Global Tool 的詳細資訊，請參閱 GitHub 上的 [.NET CLI 儲存庫的AWS 延伸項目](#)。

安裝 Docker

- 使用原生 AOT 建置 Docker 需要安裝。如需安裝指示，請參閱 [安裝 Docker 以搭配使用 AWS SAM CLI](#)。

在 AWS SAM 範本中定義 .NET 8 Lambda 事件函數

若要在 AWS SAM 範本中定義 .NET8 Lambda 函數，請執行下列動作：

- 從您選擇的開始目錄執行下列命令：

```
sam init
```

- 選取 AWS Quick Start Templates 以選擇啟動範本。
- 選擇 Hello World Example 範本。
- 透過輸入，選擇不使用最熱門的執行時間和套件類型。

5. 針對執行時間，選擇 dotnet8。
6. 針對套件類型，選擇 Zip。
7. 針對您的入門範本，選擇 Hello World Example using native AOT。

安裝 Docker

- 使用原生 AOT 建置 Docker 需要安裝。如需安裝指示，請參閱[安裝 Docker 以搭配使用 AWS SAM CLI](#)。

```
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
```

Note

當的 Event 屬性 `AWS::Serverless::Function` 設定為 `Api`，但未指定 `RestApiId` 屬性時，AWS SAM 會產生 `AWS::ApiGateway::RestApi` CloudFormation 資源。

使用 建置您的應用程式 AWS SAM CLI

從專案的根目錄中，執行 `sam build` 命令以開始建置您的應用程式。如果您的 .NET 8 專案檔案中已定義 `PublishAot` 屬性，則 AWS SAM CLI 將使用原生 AOT 編譯建置。若要進一步了解 `PublishAot` 屬性，請參閱 Microsoft .NET 文件中的[原生 AOT 部署](#)。

若要建置函數，AWS SAM CLI 會叫用使用 Amazon.Lambda.Tools .NET Core Global Tool 的 .NET Core CLI。

 Note

建置時，如果.sln檔案存在於專案的相同或父目錄中，則包含該.sln檔案的目錄會掛載到容器。如果找不到.sln檔案，則只會掛載專案資料夾。因此，如果您要建置多專案應用程式，請確保.sln檔案位於 屬性。

進一步了解

如需建置 .NET 8 Lambda 函數的詳細資訊，請參閱 [的 .NET 8 執行時間簡介 AWS Lambda](#)。

如需 sam build命令的參考，請參閱 [sam build](#)。

在 Cargo Lambda中使用 建置 Rust Lambda 事件函數 AWS SAM

將 AWS Serverless Application Model 命令列界面 (AWS SAM CLI) 與 Rust AWS Lambda 事件函數搭配使用。

主題

- [先決條件](#)
- [設定 AWS SAM 以搭配 Rust Lambda 事件函數使用](#)
- [範例](#)
- [在 中最佳化 Rust 建置 GitHub Actions](#)

先決條件

Rust 語言

若要安裝 Rust，請參閱在Rust語言網站中[安裝 Rust](#)。

Cargo Lambda

AWS SAM CLI 需要安裝 [Cargo Lambda](#)，這是 的子命令Cargo。如需安裝說明，請參閱 Cargo Lambda 文件中的[安裝](#)。

Docker

建置和測試 Rust Lambda 函數需要 Docker。如需安裝指示，請參閱[安裝 Docker](#)。

設定 AWS SAM 以搭配 Rust Lambda 事件函數使用

步驟 1：設定您的 AWS SAM 範本

使用下列項目設定您的 AWS SAM 範本：

- 二進位 – 選用。指定單一Cargo套件何時定義多個二進位檔，以識別要為此函數建置的二進位檔。當每個函數都是自己的Cargo套件時，例如在Cargo工作區中，您不需要此屬性。
- BuildMethod – rust-cargolambda。
- CodeUri – Cargo.toml 檔案的路徑。
- 處理常式 – bootstrap。
- 執行時間 – provided.al2023。

若要進一步了解自訂執行期，請參閱《AWS Lambda 開發人員指南》中的[自訂 AWS Lambda 執行期](#)。

以下是已設定 AWS SAM 範本的範例：

```
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.al2023
  ...
```

步驟 2：搭配 Rust Lambda 事件函數使用 AWS SAM CLI

將任何 AWS SAM CLI 命令與 AWS SAM 範本搭配使用。如需詳細資訊，請參閱[AWS SAM CLI](#)。

範例

Hello World 範例

在此範例中，我們使用 Rust 做為執行時間來建置範例 Hello World 應用程式。

首先，我們使用 初始化新的無伺服器應用程式 `sam init`。在互動式流程中，我們會選取 Hello World 應用程式，然後選擇 Rust 執行時間。

```
$ 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 - dotnet8
    2 - dotnet6
    3 - go (provided.al2)
    ...
    18 - python3.11
    19 - python3.10
    20 - ruby4.0
    21 - ruby3.3
    22 - ruby3.2
    23 - rust (provided.al2)
    24 - rust (provided.al2023)
Runtime: 24

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.al2023)
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 Hello World 應用程式的結構：

```

hello-rust
### README.md
### events
#   ### event.json
### rust_app
#   ### Cargo.toml
#   ### src
#       ### main.rs
### samconfig.toml
### template.yaml

```

在我們的 AWS SAM 範本中，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.al2023
      Architectures:
        - x86_64
      Events:
        HelloWorld:
          Type: Api
          Properties:
            Path: /hello
            Method: get
```

接下來，我們會執行 `sam build` 來建置應用程式並準備部署。AWS SAM CLI 會建立 `.aws-sam` 目錄，並在其中組織我們的建置成品。我們的函數是使用 `bootstrap` 建置，Cargo Lambda 並存放在 `.aws-sam/build/HelloWorldFunction/bootstrap` 的可執行二進位檔。

Note

如果您打算在 MacOS 中執行 `sam local invoke` 命令，您需要先建置不同的函數，才能叫用。若要執行此作業，請使用下列命令：

- `SAM_BUILD_MODE=debug sam build`

只有在完成本機測試時，才需要此命令。建置以進行部署時，不建議這麼做。

```
hello-rust$ sam build
Starting Build use cache
Cache is invalid, running build and copying resources for following functions
(HelloWorldFunction)
```

```

Building codeuri: /Users/.../hello-rust/rust_app runtime: provided.al2023 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

```

接著，我們使用 部署應用程式 `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-samclisam-s3-demo-
bucket-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
```

```
-----
```

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!%

```

若要在本機測試函數，首先我們會確保函數的Architectures屬性符合本機機器。

```

...
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.al2023
      Architectures:
        - arm64
...

```

由於我們在此範例中arm64將架構從 修改x86_64為，因此我們執行 sam build來更新建置成品。然後，我們會執行 sam local invoke以本機叫用函數。

```

hello-rust$ sam local invoke
Invoking bootstrap (provided.al2023)
Local image was not found.

```

```

Removing rapid images for repo public.ecr.aws/sam/emulation-provided.al2023
Building
image.....
Using local image: public.ecr.aws/lambda/provided:al2023-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 事件函數專案

以下是包含一個 Rust Lambda 函數的無伺服器應用程式範例。

專案目錄結構：

```

.
### Cargo.lock
### Cargo.toml
### src
#   ### main.rs
### template.yaml

```

AWS SAM 範本：

```

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.al2023
...

```

多個 Lambda 事件函數專案

以下是無伺服器應用程式的範例，其中包含多個 Rust Lambda 函數，組織為Cargo工作區。

對於具有多個 Rust Lambda 函數的應用程式，我們建議使用Cargo工作區。每個函數都是自己的套件，因此函數可以在透過程式庫套件共用常見程式碼時宣告獨立相依性。每個套件都會產生以套件命名的單一二進位檔，因此您不需要設定Binary建置屬性。

專案目錄結構：

```
.  
### Cargo.lock  
### Cargo.toml  
### function_a  
#   ### Cargo.toml  
#   ### src  
#       ### main.rs  
### function_b  
#   ### Cargo.toml  
#   ### src  
#       ### main.rs  
### template.yaml
```

工作區Cargo.toml檔案，位於專案根目錄：

```
[workspace]  
resolver = "2"  
members = [  
    "function_a",  
    "function_b",  
]  
  
[workspace.dependencies]  
lambda_runtime = "0.13"  
serde = { version = "1", features = ["derive"] }  
tokio = { version = "1", features = ["macros", "rt"] }
```

Cargo.toml 每個函數的 檔案，例如 function_a/Cargo.toml：

```
[package]  
name = "function_a"
```

```
version = "0.1.0"
edition = "2021"

[dependencies]
lambda_runtime = { workspace = true }
serde = { workspace = true }
tokio = { workspace = true }
```

AWS SAM 範本。每個函數CodeUri的 指向該函數的套件目錄：

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  FunctionA:
    Type: AWS::Serverless::Function
    Metadata:
      BuildMethod: rust-cargolambda
    Properties:
      CodeUri: ./function_a
      Handler: bootstrap
      Runtime: provided.al2023
  FunctionB:
    Type: AWS::Serverless::Function
    Metadata:
      BuildMethod: rust-cargolambda
    Properties:
      CodeUri: ./function_b
      Handler: bootstrap
      Runtime: provided.al2023
```

Note

會將工作區中的每個函數 AWS SAM CLI建置到工作區的共用target目錄中，因此 會為每個函數Cargo編譯一次共用相依性，而不是一次。此行為需要 AWS SAM CLI 1.165.0 版或更新版本。在舊版上，每個函數都內建在自己的target目錄中，並且會針對每個函數重新編譯完整的相依性樹狀結構，這使得在新增函數時建置速度變慢。

為每個函數套件提供一個唯一的二進位名稱。套件名稱在工作區中是唯一的，因此預設的二進位名稱已經是唯一的。如果您使用[[bin]]區段覆寫二進位名稱，請勿為兩個套件提供相同的二進位名稱。它

們會編譯至共用target目錄中的相同路徑，並互相覆寫。當偵測到此問題時，會 AWS SAM CLI記錄警告。

或者，單一套件可以定義多個二進位檔。在這種情況下，請使用Binary建置屬性來選取每個函數的二進位檔：

```
Resources:
  FunctionA:
    Type: AWS::Serverless::Function
    Metadata:
      BuildMethod: rust-cargolambda
      BuildProperties:
        Binary: function_a
    Properties:
      CodeUri: ./
      Handler: bootstrap
      Runtime: provided.al2023
```

在 中最佳化 Rust 建置 GitHub Actions

Rust 建置是運算密集型，而持續整合執行器會從沒有編譯成品開始。具有數個共用大型相依性之函數的應用程式，例如 AWS SDK，可以花費其大部分建置時間來編譯相同的相依性。下列實務可減少 中的建置時間GitHub Actions。

針對工作區使用 AWS SAM CLI 1.165.0 版或更新版本

1.165.0 版和更新版本會將Cargo工作區的每個成員建置到工作區的共用target目錄中，因此共用相依性會針對每個組建編譯一次，而不是針對每個函數編譯一次。在安裝 AWS SAM CLI 時指定最低版本，讓組建不會無提示地回復到較慢的行為。

快取Cargo登錄檔和target目錄

在執行之間快取Cargo登錄檔 (~/.cargo/registry 和 ~/.cargo/git/db) 和工作區target目錄，以便還原不變的相依性，而不是重新編譯。為每個編譯目標使用單獨的快取。跨編譯發行成品的任務arm64會產生與原生編譯的任務不同的成品x86_64，因此共用快取永遠不會相符。

在快取金鑰中包含建置設定

Cargo 包含 opt-level和 等設定codegen-units，用於決定是否可以重複使用編譯的成品。如果您變更工作區Cargo.toml檔案的 [profile.release]區段而不變更快取金鑰，快取會還原，但無論如何都會重新編譯每個木箱。在快取金鑰中包含工作區Cargo.toml檔案的雜湊，以便變更設定檔設定會啟動新的快取。

遞交您的Cargo.lock檔案

Lambda 函數是可執行檔，因此請遞交您的Cargo.lock檔案。這為您提供可重現的組建和穩定的快取金鑰，只有在相依性變更時才會變更。

調校建置時間和冷啟動的發行設定檔

您的函數程式碼會在每次執行時重新編譯，因為它變更的頻率高於您的相依性。預設版本描述檔會最佳化執行時間輸送量，許多 Lambda 函數不需要。大小最佳化會產生較小的二進位檔，這也有助於冷啟動時間，並且增加程式碼產生單位的數量會在編譯期間增加平行處理。停用連結時間最佳化(lto)，因為它會使編譯速度變慢。將下列項目新增至工作區Cargo.toml檔案：

```
[profile.release]
opt-level = "s"
codegen-units = 256
lto = false
strip = true
```

測量對您自己應用程式的影響。這些設定會針對建置時間和二進位大小，交換少量的執行時間效能。

避免重複的工作流程執行

在 push和 pull_request事件上執行的工作流程會針對相同的遞交執行兩次。GitHub Actions快取的範圍依分支和提取請求而定，因此兩者會執行寫入不同的快取範圍，而且不會重複使用另一個快取。使用在前端遞交上鍵控的並行群組，以便只有一個執行會建置每個遞交。

下列工作流程會為 建置 Rust Lambda 函數的Cargo工作區arm64，並套用上述實務：

```
name: Build

on:
  push:
    branches: [main]
  pull_request:

# Collapse the push and pull_request runs for the same commit into a single run.
concurrency:
  group: ${{ github.workflow }}-${{ github.event.pull_request.head.sha || github.sha }}
  cancel-in-progress: true

jobs:
```



```

build:
  runs-on: ubuntu-latest
  steps:
    - uses: actions/checkout@v5

    - uses: dtolnay/rust-toolchain@stable
      with:
        targets: aarch64-unknown-linux-gnu

    # Cache the Cargo registry and the workspace target directory. The key covers
    # the compilation target, Cargo.lock, and the workspace Cargo.toml, so that
    # changing a dependency or a release profile setting starts a new cache
    # instead of restoring one whose artifacts Cargo discards.
    - uses: actions/cache@v4
      with:
        path: |
          ~/.cargo/registry/index
          ~/.cargo/registry/cache
          ~/.cargo/git/db
          target
        key: cargo-arm64-${{ hashFiles('Cargo.lock', 'Cargo.toml') }}
        restore-keys: |
          cargo-arm64-

    - name: Install build tools
      run: pip install cargo-lambda 'aws-sam-cli>=1.165.0'

    - name: Build
      run: sam build

```

當金鑰不完全相符時，`restore-keys`項目可讓執行從最新的快取開始，因此相依性變更會重複使用未變更的木箱，而不是再次編譯所有項目。

在 uv 中使用 建置 Python Lambda 事件函數 AWS SAM

此功能目前為 的預覽版本 AWS SAM ，可能會有所變更。

使用 AWS Serverless Application Model 命令列界面 (AWS SAM CLI) 搭配 uv，這是一個快速的 Python 套件安裝程式和解析程式，來建置您的 Python AWS Lambda 事件函數。

主題

- [先決條件](#)
- [設定 AWS SAM 以搭配 Python Lambda 事件函數和 使用 uv](#)
- [範例](#)

先決條件

Python

若要安裝 Python，請參閱 [Python 網站上的下載](#) Python。

uv

需要安裝 [uv](#)，這是非常快速的 Python AWS SAM CLI 套件安裝程式和解析程式。如需安裝說明，請參閱 uv 文件中的 [安裝](#)。

選擇加入 AWS SAM CLI Beta 版功能

由於此功能處於預覽狀態，您必須使用下列其中一種方法選擇加入：

1. 使用環境變數：SAM_CLI_BETA_PYTHON_UV=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 SAM CLI 命令時，請使用 --beta-features 選項。例如：

```
$ sam build --beta-features
```

4. 當提示您選擇加入 y 時，AWS SAM CLI 請選擇選項。以下是範例：

```
$ sam build
Starting Build use cache
Build method "python-uv" 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 以搭配 Python Lambda 事件函數和 使用 uv

步驟 1：設定您的 AWS SAM 範本

使用下列項目設定您的 AWS SAM 範本：

- BuildMethod – python-uv。
- CodeUri – 函數程式碼目錄的路徑，其中包含 pyproject.toml 或 requirements.txt。
- 處理常式 – 您的函數處理常式（例如 app.lambda_handler）。
- 執行時間 – Python 執行時間版本（例如 python3.12）。

以下是已設定 AWS SAM 範本的範例：

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: ./my_function
      Handler: app.lambda_handler
      Runtime: python3.12
    Metadata:
      BuildMethod: python-uv
  ...
```

範例

Hello World 範例

在此範例中，我們使用 Python 搭配 uv 做為套件管理員，建置範例 Hello World 應用程式。

uv 可以使用 pyproject.toml 或 requirements.txt 讀取相依性。如果同時提供兩者，sam build 將從 requirements.txt 讀取相依性。

以下是 Hello World 應用程式的結構：

```
hello-python-uv
### README.md
### events
```

```
#   ### event.json
### hello_world
#   ### __init__.py
#   ### app.py
#   ### pyproject.toml
### samconfig.toml
### template.yaml
```

pyproject.toml 檔案：

```
[project]
name = "my-function"
version = "0.1.0"
requires-python = ">=3.12"
dependencies = [
    "requests>=2.31.0",
    "boto3>=1.28.0",
]
```

在我們的 AWS SAM 範本中，我們的 Python 函數定義如下：

```
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: python3.12
      Architectures:
        - x86_64
      Metadata:
        BuildMethod: python-uv
```

接下來，我們會執行 `sam build` 來建置應用程式並準備部署。AWS SAM CLI 會建立 `.aws-sam` 目錄，並在其中組織我們的建置成品。我們的函數相依性是使用安裝 `uv`，並存放在 `.aws-sam/build/HelloWorldFunction/`。

```
hello-python-uv$ sam build
Starting Build use cache
```

```

Build method "python-uv" 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-python-uv/hello_world runtime: python3.12 metadata:
{'BuildMethod': 'python-uv'} architecture: x86_64 functions: HelloWorldFunction
Running PythonUvBuilder:UvBuild
Running PythonUvBuilder: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

```

Note

python-uv 建置方法會依 Metadata區段中的函數設定。範本中的每個函數都可以使用不同的建置方法，讓您可以在相同的 AWS SAM 範本中混合 uv型函數與 pip型函數。如果未指定建置方法，預設pip會使用。

在 中使用自訂執行時間建置 Lambda 事件函數 AWS SAM

您可以使用 `sam build` 命令來建置 Lambda 事件函數所需的自訂執行期。您可以為函數指定 `Lambda Runtime: provided` 函數使用自訂執行期。

若要建置自訂執行期，請使用 `BuildMethod: makefile` 項目宣告 Metadata 資源屬性。您提供自訂 `makefile`，在此宣告 `build-function-logical-id` 包含執行時間建置命令之表單的建置目標。如

有必要，您的 `makefile` 負責編譯自訂執行期，並將建置成品複製到工作流程中後續步驟所需的適當位置。`makefile` 的位置是由函數資源的 `CodeUri` 屬性指定，且必須命名為 `Makefile`。

範例

範例 1：以 Rust 撰寫之函數的自訂執行時間

Note

我們建議您使用 建置 Lambda 函數 `Cargo Lambda`。如需詳細資訊，請參閱 [在 Cargo Lambda 中使用 建置 Rust Lambda 事件函數 AWS SAM](#)。

下列 AWS SAM 範本宣告 函數，該函數使用以 Rust 撰寫的 Lambda 函數的自訂執行期，並指示 `sam build` 執行 `build-HelloRustFunction` 建置目標的命令。

```
Resources:
  HelloRustFunction:
    Type: AWS::Serverless::Function
    Properties:
      FunctionName: HelloRust
      Handler: bootstrap.is.real.handler
      Runtime: provided
      MemorySize: 512
      CodeUri: .
    Metadata:
      BuildMethod: makefile
```

下列 `makefile` 包含將執行的建置目標和命令。請注意，`CodeUri` 屬性設定為 `.`，因此 `makefile` 必須位於專案根目錄（也就是與應用程式 AWS SAM 範本檔案相同的目錄）。檔案名稱必須為 `Makefile`。

```
build-HelloRustFunction:
  cargo build --release --target x86_64-unknown-linux-musl
  cp ./target/x86_64-unknown-linux-musl/release/bootstrap $(ARTIFACTS_DIR)
```

如需設定開發環境以在前一個 中執行 `cargo build` 命令的詳細資訊 `makefile`，請參閱部落格文章的 [Rust 執行期 AWS Lambda](#)。

範例 2：Makefile builder for Python3.12（使用綁定建置器的替代方法）

您可能想要使用未包含在綁定建置器中的程式庫或模組。此範例顯示具有 makefile 建置器的 Python3.12 執行時間 AWS SAM 範本。

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: hello_world/
      Handler: app.lambda_handler
      Runtime: python3.12
    Metadata:
      BuildMethod: makefile
```

下列 makefile 包含將執行的建置目標和命令。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，且檔案名稱必須為 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
```

在 中建置 Lambda 層 AWS SAM

您可以使用 AWS SAM 來建置自訂 Lambda 層。Lambda 層可讓您從 Lambda 事件函數擷取程式碼，然後在多個 Lambda 函數之間重複使用。僅建置 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。它可協助您減少部署套件的大小、將核心函數邏輯與相依性分開，並允許您跨多個函數共用相依性。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 中建置 Lambda 層 AWS SAM

Note

您必須先在 AWS SAM 範本中寫入 Lambda 層，才能建置 Lambda 層。如需執行此作業的資訊和範例，請參閱 [透過 使用 Lambda 層提高效率 AWS SAM](#)。

若要建置自訂 layer，請在您的 AWS Serverless Application Model (AWS SAM) 範本檔案中宣告它，並在 BuildMethod 項目中包含 Metadata 資源屬性區段。的有效值 BuildMethod 是 [AWS Lambda 執行時間](#) 的識別符，或 makefile。包含 BuildArchitecture 項目以指定 layer 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 build-*layer-logical-id* 包含 layer 建置命令之格式的建置目標。如有必要，您的 makefile 負責編譯 layer，並將建置成品複製到工作流程中後續步驟所需的適當位置。makefile 的位置是由 layer 資源的 ContentUri 屬性指定，且必須命名為 Makefile。

Note

當您建立自訂 layer 時，AWS Lambda 取決於環境變數來尋找您的 layer 程式碼。Lambda 執行時間包含 layer 程式碼複製到其中的 /opt 目錄中的路徑。專案的建置成品資料夾結構必須符合執行時間的預期資料夾結構，才能找到您的自訂層程式碼。

例如，對於 Python，您可以將程式碼放在 python/子目錄中。對於 NodeJS，您可以將程式碼放在 nodejs/node_modules/子目錄中。

如需詳細資訊，請參閱《AWS Lambda 開發人員指南》中的 [在 layer 中包含程式庫相依性](#)。

以下是範例 Metadata 資源屬性區段。

```
Metadata:
  BuildMethod: python3.12
  BuildArchitecture: arm64
```

Note

如果您未包含 Metadata 資源屬性區段，AWS SAM 則不會建置 layer。反之，它會從 layer 資源的 CodeUri 屬性中指定的位置複製建置成品。如需詳細資訊，請參閱 AWS::Serverless::LayerVersion 資源類型的 [ContentUri](#) 屬性。

當您包含 Metadata 資源屬性區段時，您可以使用 [sam build](#) 命令來建置 layer，無論是做為獨立物件，或是做為 AWS Lambda 函數的相依性。

- 做為獨立物件。您可能只想建置 layer 物件，例如，當您在本機測試對 layer 的程式碼變更，而且不需要建置整個應用程式時。若要獨立建置 layer，請使用 `sam build layer-logical-id` 命令指定 layer 資源。
- 做為 Lambda 函數的相依性。當您在相同 AWS SAM 範本檔案中 Lambda 函數的 Layers 屬性中包含 layer 的邏輯 ID 時，layer 是該 Lambda 函數的相依性。當該 layer 也包含具有 BuildMethod 項目 Metadata 的資源屬性區段時，您可以透過使用 `sam build` 命令建置整個應用程式或使用 `sam build function-logical-id` 命令指定函數資源來建置 layer。

範例

範本範例 1：根據 Python 3.12 執行時間環境建置 layer

下列範例 AWS SAM 範本會根據 Python 3.12 執行時間環境建置 layer。

```
Resources:
  MyLayer:
    Type: AWS::Serverless::LayerVersion
    Properties:
      ContentUri: my_layer
      CompatibleRuntimes:
        - python3.12
    Metadata:
      BuildMethod: python3.12  # Required to have AWS SAM build this layer
```

範本範例 2：使用自訂 makefile 建置 layer

下列範例 AWS SAM 範本使用自訂 makefile 來建置 layer。

```
Resources:
  MyLayer:
    Type: AWS::Serverless::LayerVersion
    Properties:
      ContentUri: my_layer
      CompatibleRuntimes:
        - python3.12
    Metadata:
      BuildMethod: makefile
```

以下makefile包含將執行的建置目標和命令。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，且檔案名稱必須為 Makefile。另請注意，建置成品會複製到 python/子目錄，以便 AWS Lambda 能夠找到 layer 程式碼。

```
build-MyLayer:
  mkdir -p "${ARTIFACTS_DIR}/python"
  cp *.py "${ARTIFACTS_DIR}/python"
  python -m pip install -r requirements.txt -t "${ARTIFACTS_DIR}/python"
```

Note

呼叫 makefile 時，會觸發適當的目標，並將成品複製到公開的環境變數 \$ARTIFACTS_DIR。如需詳細資訊，請參閱 [GitHub 中的 aws-lambda-builders](#)。

範例 sam 建置命令

下列sam build命令會建置包含Metadata資源屬性區段的層。

```
# 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
```

使用 測試無伺服器應用程式 AWS SAM

撰寫和建置應用程式後，您將準備好測試應用程式，以驗證其是否正常運作。使用 AWS SAM 命令列界面 (CLI)，您可以在本機測試無伺服器應用程式，然後再將其上傳至 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）。

本節提供您可以遵循以測試應用程式的常見實務指引。本節中的主題主要著重於在 AWS 雲端部署之前，您可以執行的本機測試。部署前的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的成本。本節中的每個主題都會說明您可以執行的測試，並包含示範如何執行測試的範例。測試應用程式之後，您就可以偵錯發現的任何問題。

主題

- [使用 sam local 命令進行測試的簡介](#)
- [使用 本機叫用 Lambda 事件函數 AWS SAM](#)
- [使用 本機執行 API Gateway AWS SAM](#)
- [使用 進行雲端測試的簡介 sam remote test-event](#)
- [在雲端使用 進行測試的簡介 sam remote invoke](#)
- [使用 自動化本機整合測試 AWS SAM](#)
- [使用 產生範例事件承載 AWS SAM](#)
- [測試和偵錯耐用函數](#)

使用 sam local 命令進行測試的簡介

使用 AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam local` 命令在本機測試無伺服器應用程式。

如需的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)。

先決條件

若要使用 `sam local`，請完成下列步驟來 AWS SAM CLI 安裝：

- [AWS SAM 先決條件](#).
- [安裝 AWS SAM CLI](#).

使用之前 `sam local`，我們建議您對以下內容有基本的了解：

- [設定 AWS SAM CLI.](#)
- [在 中建立您的應用程式 AWS SAM.](#)
- [使用 建置 簡介 AWS SAM.](#)
- [使用 部署簡介 AWS SAM.](#)

使用 sam local命令

使用 `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>
```

若要進一步了解每個子命令，請參閱以下內容：

- [簡介 sam local generate-event](#) – 產生本機測試 AWS 服務 的事件。
- [簡介 sam local invoke](#) – 在本機啟動事件函數的 AWS Lambda 一次性調用。
- [簡介 sam local start-api](#) – 使用本機 HTTP 伺服器執行 Lambda 函數。
- [簡介 sam local start-lambda](#) – 使用本機 HTTP 伺服器執行 Lambda 函數，以搭配 AWS CLI 或 SDKs使用。

使用 進行測試的簡介 sam local generate-event

使用 AWS Serverless Application Model 命令列界面 (AWS SAM CLI) `sam local generate-event`子命令來產生受支援的事件承載範例 AWS 服務。然後，您可以修改這些事件並將其傳遞至本機資源進行測試。

- 如需 的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)
- 如需`sam local generate-event`命令選項的清單，請參閱 [sam local generate-event](#)。

事件是在 AWS 服務 執行動作或任務時產生的 JSON 物件。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 產生事件，每個服務的事件都會針對其服務進行唯一的格式化。

由一個服務產生的事件會做為事件來源傳遞給其他 服務。例如，放置在 Amazon Simple Storage Service (Amazon S3) 儲存貯體中的項目可以產生事件。然後，此事件可以用作 AWS Lambda 事件函數的事件來源，以進一步處理資料。

您使用產生的事件 `sam local generate-event` 格式與 AWS 服務建立的實際事件相同。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。

先決條件

若要使用 `sam local generate-event`，請完成下列步驟來 AWS SAM CLI 安裝：

- [AWS SAM 先決條件](#).
- [安裝 AWS SAM CLI](#).

使用之前 `sam local generate-event`，我們建議您對以下內容有基本的了解：

- [設定 AWS SAM CLI](#).
- [在中建立您的應用程式 AWS SAM](#).
- [使用建置簡介 AWS SAM](#).
- [使用部署簡介 AWS SAM](#).

產生範例事件

使用 AWS SAM CLI `sam local generate-event` 子命令為支援的產生事件 AWS 服務。

查看支援的清單 AWS 服務

1. 執行下列命令：

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

2. 支援的清單 AWS 服務 隨即顯示。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。這會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": "sam-s3-demo-bucket",
            "ownerIdentity": {
                "principalId": "EXAMPLE"
            },
            "arn": "arn:aws:s3:::sam-s3-demo-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。

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，請執行下列動作：

```
$ 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:
```

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

2. 在命令提示字元使用這些選項來修改您的範例事件承載。以下是範例：

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

{
  "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": "sam-s3-demo-bucket",

```



```

    "ownerIdentity": {
      "principalId": "EXAMPLE"
    },
    "arn": "arn:aws:s3:::sam-s3-demo-bucket"
  },
  "object": {
    "key": "test/key",
    "size": 1024,
    "eTag": "0123456789abcdef0123456789abcdef",
    "sequencer": "0A1B2C3D4E5F678901"
  }
}
]
}

```

使用產生的事件進行本機測試

在本機儲存您產生的事件，並使用其他 `sam local` 子命令進行測試。

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>
```

以下是將事件儲存為專案 `events` 資料夾中 `s3.json` 檔案的範例。

```
sam-app$ sam local generate-event s3 put --bucket amzn-s3-demo-bucket > events/
s3.json
```

使用產生的事件進行本機測試

- 使用 `--event` 選項與其他 `sam local` 子命令傳遞事件。

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](#)。

如需使用的示範 `sam local`，請參閱 [AWS SAM 以取得本機開發。在上的 Serverless Land Sessions 中使用 SAM 系列YouTube測試來自本機開發環境 AWS 雲端的資源。](#)

使用 進行測試的簡介 sam local invoke

使用 AWS Serverless Application Model 命令列界面 (AWS SAM CLI) `sam local invoke` 子命令在本機啟動事件 AWS Lambda 函數的一次性調用。

- 如需的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)
- 如需 `sam local invoke` 命令選項的清單，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：（選用）在本機測試您的應用程式。](#)

Note

不建議在不受信任的程式碼中使用 SAM CLI 的本機調用功能。若要與本機環境完全隔離，請直接在 Lambda 服務中執行程式碼。

Note

`sam local invoke` 支援具有自動檢查點和重播功能的耐用函數。在本機叫用耐用函數時，會自動管理執行狀態。

先決條件

若要使用 `sam local invoke`，請完成下列步驟來 AWS SAM CLI 安裝：

- [AWS SAM 先決條件](#).
- [安裝 AWS SAM CLI](#).

使用之前 `sam local invoke`，我們建議您對以下內容有基本的了解：

- [設定 AWS SAM CLI](#).
- [在 中建立您的應用程式 AWS SAM](#).
- [使用 建置 簡介 AWS SAM](#).
- [使用 部署簡介 AWS SAM](#).

在本機叫用 Lambda 事件函數

當您執行時 `sam local invoke`，AWS SAM CLI 會假設您目前的工作目錄是專案的根目錄。AWS SAM CLI 會先尋找 `.aws-sam` 子資料夾中 `template.[yaml|yml]` 的檔案。如果找不到，AWS SAM CLI 會在您目前的工作目錄中尋找 `template.[yaml|yml]` 檔案。

在本機叫用 Lambda 事件函數

1. 從專案的根目錄中，執行下列動作：

```
$ 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. 會使用在 AWS SAM CLI 本機容器中建置您的 函數 Docker。然後，它會叫用您的函數並輸出函數的回應。

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 函數執行期輸出（例如日誌）會輸出至 `stderr`，Lambda 函數結果則會輸出至 `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本機開發程序。

選項

傳遞自訂事件以叫用 Lambda 事件函數

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。如需詳細資訊，請參閱 [使用 進行測試的簡介 sam local generate-event](#)。

叫用 Lambda 事件函數時傳遞環境變數

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` 選項傳遞它們。這是使用已在雲端部署的應用程式中的服務在本機測試 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

若要指定範本供 AWS SAM CLI 參考，請使用 `--template` 選項。AWS SAM CLI 只會載入該 AWS SAM 範本及其指向的資源。

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 Terraform 專案測試 Lambda 事件函數

使用 `--hook-name` 選項從 Terraform 專案本機測試 Lambda 函數。如需詳細資訊，請參閱 [搭配 使用 AWS SAM CLITerraform 進行本機偵錯和測試](#)。

以下是範例：

```
$ sam local invoke --hook-name terraform --beta-features
```

最佳實務

如果您的應用程式具有執行的 `.aws-sam` 目錄 `sam build`，請務必 `sam build` 在每次更新函數程式碼時執行。然後，執行 `sam local invoke` 以在本機測試您更新的函數程式碼。

在部署到雲端之前，本機測試是快速開發和測試的絕佳解決方案。不過，本機測試不會驗證所有項目，例如雲端中資源之間的許可。盡可能在雲端測試您的應用程式。建議使用 [sam sync](#) 來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 事件函數

首先，我們產生 API Gateway HTTP API 事件承載，並將其儲存至我們的 `events` 資料夾。

```
$ sam local generate-event apigateway http-api-proxy > events/apigateway_event.json
```

接著，我們會修改 Lambda 函數，以從事件傳回參數值。

```
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\"}"}
```

在本機叫用 Lambda 事件函數時傳遞環境變數

此應用程式具有 Lambda 函數，其使用 Amazon DynamoDB 資料表名稱的環境變數。以下是 AWS SAM 範本中定義的 函數範例：

```
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我們的 Lambda 函數，同時讓它與雲端中的 DynamoDB 資料表互動。若要這樣做，我們會建立環境變數檔案，並將其儲存在專案的根目錄中，做為 `locals.json`。此處提供的 值會 `SAMPLE_TABLE` 參考雲端中的 DynamoDB 資料表。

```
{
  "getAllItemsFunction": {
    "SAMPLE_TABLE": "dev-demo-SampleTable-1U991234LD5UM98"
  }
}
```

接下來，我們會使用 `--env-vars` 選項執行 `sam local invoke` 並傳入環境變數。

```
$ 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](#)。

如需使用的示範 `sam local`，請參閱 [AWS SAM 以取得本機開發。在上的 Serverless Land Sessions 中使用 SAM 系列YouTube測試來自本機開發環境 AWS 雲端的資源。](#)

使用 進行測試的簡介 sam local start-api

使用 AWS Serverless Application Model 命令列界面 (AWS SAM CLI) `sam local start-api` 子命令在本機執行事件 AWS Lambda 函數，並透過本機 HTTP 伺服器主機進行測試。這種類型的測試有助於 Amazon API Gateway 端點調用的 Lambda 函數。

- 如需的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)
- 如需 `sam local start-api` 命令選項的清單，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：（選用）在本機測試您的應用程式。](#)

Note

`sam local start-api` 支援具有自動檢查點和重播功能的耐用函數。耐用的函數可與本機 API Gateway 端點無縫搭配使用。

先決條件

若要使用 `sam local start-api`，請完成下列步驟來 AWS SAM CLI 安裝：

- [AWS SAM 先決條件](#)。
- [安裝 AWS SAM CLI](#)。

使用 `之前 sam local start-api`，我們建議您對以下內容有基本的了解：

- [設定 AWS SAM CLI](#).
- [在 中建立您的應用程式 AWS SAM](#).
- [使用 建置 簡介 AWS SAM](#).
- [使用 部署簡介 AWS SAM](#).

使用 sam 本機 start-api

當您執行 `時 sam local start-api`，AWS SAM CLI 會假設您目前的工作目錄是專案的根目錄。AWS SAM CLI 會先尋找 `.aws-sam` 子資料夾中 `template.[yaml|yml]` 的檔案。如果找不到，AWS SAM CLI 會在您目前的工作目錄中尋找 `template.[yaml|yml]` 檔案。

啟動本機 HTTP 伺服器

1. 從專案的根目錄中，執行下列動作：

```
$ sam local start-api <options>
```

2. 會在本機 Docker 容器中建置 Lambda AWS SAM CLI 函數。然後，它會輸出 HTTP 伺服器端點的本機地址。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，請考慮下列事項以重新整理本機 HTTP 伺服器：

- 如果您的應用程式沒有 `.aws-sam` 目錄，且您的函數使用解譯語言，則 AWS SAM CLI 會透過建立新的容器並託管它，自動更新您的函數。
- 如果您的應用程式有 `.aws-sam` 目錄，則需要執行 `sam build` 以更新函數。然後 `sam local start-api` 再次執行以託管函數。
- 如果您的函數使用編譯語言，或您的專案需要複雜的封裝支援，請執行您自己的建置解決方案來更新您的函數。然後 `sam local start-api` 再次執行以託管函數。

使用 Lambda 授權方的 Lambda 事件函數

 Note

此功能是 1.80.0 版的新功能 AWS SAM CLI。若要升級，請參閱[升級 AWS SAM CLI](#)。

對於使用 Lambda 授權方的 Lambda 函數，在 AWS SAM CLI 叫用 Lambda 函數端點之前，會自動叫用您的 Lambda 授權方。

以下是為使用 Lambda 授權方的函數啟動本機 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 SAM CLI 第一個叫用您的 Lambda 授權方。如果授權成功，AWS SAM CLI 將調用您的 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

持續重複使用容器以加速本機函數叫用

根據預設，每次透過本機 HTTP 伺服器叫用函數時，AWS SAM CLI 都會建立新的容器。使用 `--warm-containers` 選項自動重複使用您的容器進行函數調用。這可加快 AWS SAM CLI 準備 Lambda 函數以進行本機調用所需的時間。您可以提供 `eager` 或 `lazy` 引數，進一步自訂此選項。

- `eager` – 所有函數的容器會在啟動時載入，並在叫用之間保留。
- `lazy` – 容器只有在第一次叫用每個函數時才會載入。然後，它們會保留用於其他調用。

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：

- 如果您的應用程式有 `.aws-sam` 目錄，請執行 `sam build` 以更新應用程式建置成品中的函數程式碼。
- 偵測到程式碼變更時，AWS SAM CLI 會自動關閉 Lambda 函數容器。
- 當您再次叫用 函數時，AWS SAM CLI 會自動建立新的容器。

如果檔案監看造成高 CPU 或 I/O 額外負荷，例如，在大型單一儲存、具有防毒軟體的環境，或多個函數共用的專案 `CodeUri` 中，您可以使用 `--no-watch` 選項將其停用：

```
$ sam local start-api --warm-containers eager --no-watch
```

使用 `--no-watch`，不會自動偵測程式碼和範本變更。停止並重新執行 命令以接收變更。

指定要用於 Lambda 事件函數的容器映像

根據預設，AWS SAM CLI 會使用來自 Amazon Elastic Container Registry (Amazon ECR) 的 Lambda 基礎映像，在本機叫用您的函數。使用 `--invoke-image` 選項來參考自訂容器映像。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

若要指定範本供 AWS SAM CLI 參考，請使用 `--template` 選項。AWS SAM CLI 只會載入該 AWS SAM 範本及其指向的資源。以下是範例：

```
$ sam local start-api --template myTemplate.yaml
```

指定 Lambda 事件函數的主機開發環境

根據預設，`sam local start-api` 子命令會使用 IP 地址為 `localhost` 的建立 HTTP 伺服器 `127.0.0.1`。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` 選項來指定主機。以下是範例：

```
$ sam local start-api --container-host host.docker.internal
```

使用 `--container-host-interface` 選項指定容器連接埠應繫結之主機網路的 IP 地址。以下是範例：

```
$ sam local start-api --container-host-interface 0.0.0.0
```

最佳實務

如果您的應用程式具有執行的 `.aws-sam` 目錄 `sam build`，請務必 `sam build` 在每次更新函數程式碼時執行。然後，執行 `sam local start-api` 以在本機測試您更新的函數程式碼。

在部署到雲端之前，本機測試是快速開發和測試的絕佳解決方案。不過，本機測試不會驗證所有項目，例如雲端中資源之間的許可。盡可能在雲端測試您的應用程式。建議使用 [sam sync](#) 來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](#)。

使用 進行測試的簡介 `sam local start-lambda`

使用 AWS SAM CLI 子命令透過 AWS CLI 和 SDKs `sam local start-lambda` 叫用 Lambda 事件函數。此命令會啟動模擬 Lambda 的本機端點。

- 如需的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)
- 如需 `sam local start-lambda` 命令選項的清單，請參閱 [sam local start-lambda](#)。

Note

`sam local start-lambda` 支援具有自動檢查點和重播功能的耐用函數。耐用的函數可與本機 Lambda 端點無縫搭配使用。

先決條件

若要使用 `sam local start-lambda`，請完成下列步驟來 AWS SAM CLI 安裝：

- [AWS SAM 先決條件](#).
- [安裝 AWS SAM CLI](#).

使用之前 `sam local start-lambda`，我們建議您對以下內容有基本的了解：

- [設定 AWS SAM CLI](#).
- [在中建立您的應用程式 AWS SAM](#).
- [使用建置簡介 AWS SAM](#).
- [使用部署簡介 AWS SAM](#).

使用 sam 本機 start-lambda

當您執行時 `sam local start-lambda`，AWS SAM CLI 會假設您目前的工作目錄是專案的根目錄。AWS SAM CLI 會先尋找 `.aws-sam` 子資料夾中 `template.[yaml|yml]` 的檔案。如果找不到，AWS SAM CLI 會在您目前的工作目錄中尋找 `template.[yaml|yml]` 檔案。

使用 `sam local start-lambda`

1. 從專案的根目錄中，執行下列動作：

```
$ sam local start-lambda <options>
```

2. 會在本機 Docker 容器中建置 Lambda AWS SAM CLI 函數。然後，它會將本機地址輸出到您的 HTTP 伺服器端點。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 SDKs 在本機叫用 Lambda 函數。

以下是使用 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)
```

以下是使用適用於 的 AWS SDK 的範例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

若要指定範本供 AWS SAM CLI 參考，請使用 `--template` 選項。AWS SAM CLI 只會載入該 AWS SAM 範本及其指向的資源。以下是範例：

```
$ sam local start-lambda --template myTemplate.yaml
```

如需 AWS SAM 範本的詳細資訊，請參閱 [AWS SAM 範本結構](#)。

最佳實務

如果您的應用程式具有執行的 `.aws-sam` 目錄 `sam build`，請務必 `sam build` 在每次更新函數程式碼時執行。然後，執行 `sam local start-lambda` 以在本機測試您更新的函數程式碼。

在部署到雲端之前，本機測試是快速開發和測試的絕佳解決方案。不過，本機測試不會驗證所有項目，例如雲端中資源之間的許可。盡可能在雲端測試您的應用程式。建議使用 [sam sync](#) 來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 事件函數 AWS SAM

在雲端中測試或部署之前，在本機叫用 Lambda 事件函數可以有各種好處。它可讓您更快地測試函數的邏輯。本機測試會先降低在雲端或在部署期間測試時發現問題的可能性，這可協助您避免不必要的成本。此外，本機測試可讓您更輕鬆地進行除錯。

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` 做為事件。如需事件的詳細資訊，請參閱《AWS Lambda 開發人員指南》中的 [事件](#)。如需不同 AWS 服務的事件訊息格式相關資訊，請參閱《AWS Lambda 開發人員指南》中的 [搭配使用 AWS Lambda 與其他 服務](#)。

Note

不建議在不受信任的程式碼中使用 SAM CLI 的本機調用功能。若要與本機環境完全隔離，請直接在 Lambda 服務中執行程式碼。

 Note

`sam local invoke` 命令對應至 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。

您必須在專案目錄中執行 `sam local invoke` 命令，其中包含您要叫用的 函數。

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值的環境變數，請執行下列動作：

1. 建立 JSON 或 `.env` 檔案，其中包含要覆寫的環境變數。
2. 使用 `--env-vars` 引數覆寫範本中定義的值。

`--env-vars` 選項支援兩種檔案格式。根據檔案內容自動偵測檔案格式。

使用 JSON 宣告環境變數

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": "amzn-s3-demo-bucket",
    "STAGE": "dev"
  }
}
```

若要為每個資源宣告不同的環境變數，請為每個資源指定物件，如下所示：

```
{
  "MyFunction1": {
    "TABLE_NAME": "localtable",
    "BUCKET_NAME": "amzn-s3-demo-bucket",
  },
  "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符，依最高至最低優先順序列出：

1. logical_id
2. function_id
3. function_name
4. 完整路徑識別符

您可以使用上述兩種在單一檔案中同時宣告環境變數的方法。執行此操作時，您針對特定資源提供的環境變數優先於全域環境變數。

將您的環境變數儲存在 JSON 檔案中，例如 `env.json`。

使用 .env 檔案宣告環境變數

您也可以使用 `.env` 檔案來宣告環境變數。`.env` 檔案中宣告的變數會全域套用至所有函數，相當於 JSON 格式的 `Parameters` 物件。

```
TABLE_NAME=localtable
BUCKET_NAME=amzn-s3-demo-bucket
STAGE=dev
```

`.env` 格式支援註解（以開頭的行#）和引號值。

Note

`.env` 格式僅支援全域環境變數。若要宣告函數特定的環境變數，請使用 JSON 格式。

覆寫環境變數值

若要將環境變數覆寫為環境變數檔案中定義的變數，請使用 `--env-vars` 引數搭配 `invoke` 或 `start-api` 命令。例如：

```
# Using a JSON file
sam local invoke --env-vars env.json

# Using a .env file
sam local invoke --env-vars .env
```

層

如果您的應用程式包含 layer，如需如何偵錯本機主機上 layer 問題的相關資訊，請參閱 [透過使用 Lambda 層提高效率 AWS SAM](#)。

進一步了解

如需在本機叫用 函數的實作範例，請參閱 完成 AWS SAM 研討會中的 [單元 2 - 在本機執行](#)。

使用 本機執行 API Gateway AWS SAM

在本機執行的 Amazon API Gateway 可能有各種優點。例如，在本機執行 API Gateway 可讓您在本機測試 API 端點，然後再部署到 AWS 雲端。如果您先在本機進行測試，通常可以減少雲端的測試和開發，這有助於降低成本。此外，在本機執行 可讓偵錯更容易。

若要啟動可用於測試 HTTP 請求/回應功能的 API Gateway 本機執行個體，請使用 `sam local start-api` AWS SAM CLI 命令。此功能具有熱重新載入功能，讓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時，應用程式的狀態會遺失。

如需使用 `sam local start-api` 命令的指示，請參閱 [使用 進行測試的簡介 sam local start-api](#)。

根據預設，AWS SAM 會使用 AWS Lambda 代理整合，並支援 `HttpApi` 和 `Api` 資源類型。如需 `HttpApi` 資源類型的代理整合詳細資訊，請參閱《API Gateway 開發人員指南》中的 [使用 HTTP](#)

[APIs AWS Lambda 代理整合](#)。如需與Api資源類型進行代理整合的詳細資訊，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，它會將函數掛載在定義的 HTTP 路徑。

在下列Api範例中，`Ratings`函數`ratings.py:handler()`會在掛載`/ratings`GET請求：

```
Ratings:
  Type: AWS::Serverless::Function
  Properties:
    Handler: ratings.handler
    Runtime: python3.9
    Events:
      Api:
        Type: Api
        Properties:
          Path: /ratings
          Method: get
```

以下是Api回應範例：

```
// Example of a Proxy Integration response
exports.handler = (event, context, callback) => {
  callback(null, {
    statusCode: 200,
    headers: { "x-custom-header" : "my custom header value" },
    body: "hello world"
  });
}
```

如果您修改函數的程式碼，請執行的 `sam build` 命令 `sam local start-api` 來偵測您的變更。

環境變數檔案

若要在本機宣告覆寫範本中定義值的環境變數，請執行下列動作：

1. 建立 JSON 檔案，其中包含要覆寫的環境變數。
2. 使用 `--env-vars` 引數覆寫範本中定義的值。

宣告環境變數

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": "amzn-s3-demo-bucket",
    "STAGE": "dev"
  }
}
```

若要宣告每個資源的不同環境變數，請為每個資源指定物件，如下所示：

```
{
  "MyFunction1": {
    "TABLE_NAME": "localtable",
    "BUCKET_NAME": "amzn-s3-demo-bucket",
  },
  "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符，依最高至最低優先順序列出：

1. `logical_id`
2. `function_id`
3. `function_name`
4. 完整路徑識別符

您可以使用上述兩種在單一檔案中同時宣告環境變數的方法。執行此操作時，您提供給特定資源的環境變數優先於全域環境變數。

將您的環境變數儲存在 JSON 檔案中，例如 `env.json`。

覆寫環境變數值

若要使用 JSON 檔案中定義的變數覆寫環境變數，請使用 `--env-vars` 引數搭配 `invoke` 或 `start-api` 命令。例如：

```
$ sam local start-api --env-vars env.json
```

層

如果您的應用程式包含 layer，如需如何偵錯本機主機上 layer 問題的相關資訊，請參閱 [透過 使用 Lambda 層提高效率 AWS SAM](#)。

使用 進行雲端測試的簡介 sam remote test-event

使用 AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam remote test-event` 命令來存取和管理事件函數的可共用測試 AWS Lambda 事件。

若要進一步了解可共用測試事件，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 SAM CLI 以使用 sam remote test-event](#)
- [使用 sam remote test-event 命令](#)
- [使用可共用的測試事件](#)
- [管理可共用的測試事件](#)

先決條件

若要使用 `sam remote test-event`，請完成下列步驟來 AWS SAM CLI 安裝：

- [AWS SAM 先決條件](#).
- [安裝 AWS SAM CLI](#).

如果您已安裝 AWS SAM CLI，建議您升級至最新版本的 AWS SAM CLI 版本。如需詳細資訊，請參閱 [升級 AWS SAM CLI](#)。

使用 之前 `sam remote test-event`，我們建議您對以下內容有基本的了解：

- [設定 AWS SAM CLI.](#)
- [在 中建立您的應用程式 AWS SAM.](#)
- [使用 建置 簡介 AWS SAM.](#)
- [使用 部署簡介 AWS SAM.](#)
- [使用 `sam sync` 同步至 的簡介 AWS 雲端.](#)

設定 AWS SAM CLI以使用 `sam remote test-event`

完成下列設定步驟以使用 AWS SAM CLI的 `sam remote test-event` 命令：

1. 設定 AWS SAM CLI以使用您的 AWS 帳戶 – 相同 中的使用者可以存取和管理 Lambda 的可共用測試事件 AWS 帳戶。若要設定 AWS SAM CLI以使用您的 AWS 帳戶，請參閱 [設定 AWS SAM CLI](#)。
2. 設定可共用測試事件的許可 – 若要存取和管理可共用測試事件，您必須擁有適當的許可。若要進一步了解，請參閱《AWS Lambda 開發人員指南》中的 [可共用測試事件](#)。

使用 `sam remote test-event` 命令

`sam remote test-event` 命令 AWS SAM CLI提供下列子命令，您可以使用這些子命令來存取和管理可共用的測試事件：

- `delete` – 從 Amazon EventBridge 結構描述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 結構描述登錄檔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 結構描述登錄檔。

若要使用 列出這些子命令 AWS SAM CLI，請執行下列動作：

```
$ sam remote test-event --help
```

刪除可共用的測試事件

您可以使用 `delete` 子命令以及下列項目來刪除可共用的測試事件：

- 提供要刪除的可共用測試事件的名稱。

- 提供與事件相關聯的 Lambda 函數可接受的 ID。
- 如果您要提供 Lambda 函數邏輯 ID，您還必須提供與 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
```

如需搭配 delete 子命令使用的選項清單，請參閱 [sam remote test-event delete](#)。您也可以從 執行下列項目 AWS SAM CLI：

```
$ sam remote test-event delete --help
```

取得可共用的測試事件

您可以使用 get 子命令以及下列項目，從 EventBridge 結構描述登錄檔取得可共用的測試事件：

- 提供要取得的可共用測試事件的名稱。
- 提供與事件相關聯的 Lambda 函數可接受的 ID。
- 如果您要提供 Lambda 函數邏輯 ID，您還必須提供與 Lambda 函數相關聯的 AWS CloudFormation 堆疊名稱。

以下是取得名為 demo-event 且與 sam-app 堆疊的 HelloWorldFunction Lambda 函數相關聯的可共用測試事件的範例。此命令會將事件列印到您的主控台。

```
$ 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 --output-file 選項並提供檔案路徑和名稱。以下是將 另存 demo-event 為 demo-event.json 目前工作目錄中的範例：

```
$ sam remote test-event get HelloWorldFunction --stack-name sam-app --name demo-event --output-file demo-event.json
```

如需搭配 get 子命令使用的選項清單，請參閱 [sam remote test-event get](#)。您也可以從 執行下列項目 AWS SAM CLI：


```
$ sam remote test-event get --help
```

列出可共用的測試事件

您可以從結構描述登錄檔列出特定 Lambda 函數的所有可共用測試事件。使用 `list` 子命令以及下列項目：

- 提供與事件相關聯的 Lambda 函數可接受的 ID。
- 如果您要提供 Lambda 函數邏輯 ID，您還必須提供與 Lambda 函數相關聯的 CloudFormation 堆疊名稱。

以下是取得與 `sam-app` 堆疊的 `HelloWorldFunction` Lambda 函數相關聯的所有可共用測試事件清單的範例：

```
$ sam remote test-event list HelloWorldFunction --stack-name sam-app
```

如需搭配 `list` 子命令使用的選項清單，請參閱 [sam remote test-event list](#)。您也可以從 執行下列項目 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到 EventBridge 結構描述登錄檔。使用 `put` 子命令以及下列項目：

- 提供與可共用測試事件相關聯的 Lambda 函數可接受的 ID。
- 提供可共用測試事件的名稱。
- 將檔案路徑和名稱提供給要上傳的本機事件。

下列範例會將本機 `demo-event.json` 事件儲存為 `demo-event` 並將其與 `sam-app` 堆疊的 `HelloWorldFunction` Lambda 函數建立關聯：

```
$ sam remote test-event put HelloWorldFunction --stack-name sam-app --name demo-event --file demo-event.json
```

如果 EventBridge 結構描述登錄檔中存在同名的可共用測試事件，則 AWS SAM CLI 不會覆寫它。若要覆寫，請將 `--force` 選項新增至命令。

如需搭配 `put` 子命令使用的選項清單，請參閱 [sam remote test-event put](#)。您也可以從 執行下列項目 AWS SAM CLI：

```
$ sam remote test-event put --help
```

使用可共用的測試事件

使用可共用的測試事件，AWS 雲端 透過 `sam remote invoke` 命令在 中測試 Lambda 函數。如需詳細資訊，請參閱 [將可共用的測試事件傳遞至雲端中的 Lambda 事件函數](#)。

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。

取得可共用的測試事件、修改並加以使用

您可以從 EventBridge 結構描述登錄檔取得可共用的測試事件，在本機進行修改，並在 中使用本機測試事件與 Lambda 函數 AWS 雲端。以下是範例：

1. 擷取可共用測試事件 – 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. 修改可共用測試事件 – 使用您選擇的文字編輯器來修改可共用測試事件。
3. 使用可共用的測試事件 – 使用 `sam remote invoke` 命令，並使用 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、上傳和使用

您可以從 EventBridge 結構描述登錄檔取得可共用的測試事件，在本機進行修改並上傳。然後，您可以將可共用的測試事件直接傳遞至 中的 Lambda 函數 AWS 雲端。以下是範例：

1. 擷取可共用測試事件 – 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. 修改可共用測試事件 – 使用您選擇的文字編輯器來修改可共用測試事件。
3. 上傳可共用測試事件 – 使用 `sam remote test-event put` 子命令將可共用測試事件上傳並儲存到 EventBridge 結構描述登錄檔。在此範例中，我們使用 `--force` 選項覆寫可共用測試的較舊版本：

```
$ sam remote test-event put HelloWorldFunction --stack-name sam-app --name demo-event --file demo-event.json --force
```

4. 將可共用測試事件傳遞至 Lambda 函數 – 使用 `sam remote invoke` 命令將可共用測試事件直接傳遞至 中的 Lambda 函數 AWS 雲端：

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --test-event-name demo-event
```

在雲端使用 進行測試的簡介 `sam remote invoke`

使用 AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam remote invoke` 命令與 中支援 AWS 的資源互動 AWS 雲端。您可以使用 `sam remote invoke` 叫用下列資源：

- Amazon Kinesis Data Streams – 將資料記錄傳送至 Kinesis Data Streams 應用程式。
- AWS Lambda – 叫用事件並將其傳遞到您的 Lambda 事件函數。
- Amazon Simple Queue Service (Amazon SQS) – 傳送訊息至 Amazon SQS 佇列。
- AWS Step Functions – 叫用 Step Functions 狀態機器以開始執行。

如需 的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)

如需在典型開發工作流程 `sam remote invoke` 期間使用的範例，請參閱 [步驟 5：與 中的 函數互動 AWS 雲端](#)。

主題

- [使用 `sam` 遠端叫用命令](#)
- [使用 `sam` 遠端叫用命令選項](#)

- [設定您的專案組態檔案](#)
- [範例](#)
- [相關連結](#)

先決條件

若要使用 `sam remote invoke`，請完成下列步驟來 AWS SAM CLI 安裝：

- [AWS SAM 先決條件](#).
- [安裝 AWS SAM CLI](#).

我們也建議您升級至最新版本的 AWS SAM CLI。如需詳細資訊，請參閱 [升級 AWS SAM CLI](#)。

使用之前 `sam remote invoke`，我們建議您對以下內容有基本的了解：

- [設定 AWS SAM CLI](#).
- [在 中建立您的應用程式 AWS SAM](#).
- [使用 建置 簡介 AWS SAM](#).
- [使用 部署簡介 AWS SAM](#).
- [使用 `sam sync` 同步至 的簡介 AWS 雲端](#).

使用 sam 遠端叫用命令

使用此命令之前，您的資源必須部署到 AWS 雲端。

使用下列命令結構，並從專案的根目錄執行：

```
$ sam remote invoke <arguments> <options>
```

Note

此頁面會顯示命令提示中提供的選項。您也可以專案的組態檔案中設定選項，而不是在命令提示字元中傳遞。如需進一步了解，請參閱 [設定專案設定](#)。

如需 `sam remote invoke` 引數和選項的描述，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 會傳送您的資料記錄並傳回碎片 ID 和序號。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": "49646251411914806775980850790050483811301135051202232322"
}%
```

傳送資料記錄

1. 提供資源 ID 值做為 Kinesis Data Streams 應用程式的引數。如需有效資源 IDs 的資訊，請參閱[資源 ID](#)。
2. 提供資料記錄做為要傳送至 Kinesis Data Streams 應用程式的事件。您可以使用 `--event` 選項在命令列提供事件，或使用 `從 檔案提供事件--event-file`。如果您不提供事件，AWS SAM CLI 會傳送空事件。

搭配 Lambda 事件函數使用

您可以在雲端中叫用 Lambda 函數並傳遞空事件，或在命令列或從檔案提供事件。會叫用您的 Lambda AWS SAM CLI 函數並傳回其回應。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\"}"}%
```

叫用 Lambda 事件函數

1. 提供資源 ID 值做為 Lambda 函數的引數。如需有效資源 IDs 的資訊，請參閱[資源 ID](#)。
2. 提供要傳送至 Lambda 函數的事件。您可以使用 `--event` 選項在命令列提供事件，或使用 `從檔案提供事件--event-file`。如果您不提供事件，AWS SAM CLI 會傳送空事件。

使用回應串流設定的 Lambda 事件函數

`sam remote invoke` 命令支援設定為串流回應的 Lambda 函數。您可以設定 Lambda 函數，以使用 AWS SAM 範本中的 [FunctionUrlConfig](#) 屬性串流回應。當您使用 `sam remote invoke`，AWS SAM CLI 會自動偵測您的 Lambda 組態，並使用回應串流叫用。

如需範例，請參閱 [叫用設定為串流回應的 Lambda 事件函數](#)。

將可共用的測試事件傳遞至雲端中的 Lambda 事件函數

可共用的測試事件是您可以在相同中與其他入共用的測試事件 AWS 帳戶。若要進一步了解，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 AWS SAM CLI `sam remote test-event` 命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` 執行下列動作：

- 從 Amazon EventBridge 結構描述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 結構描述登錄檔。
- 從 EventBridge 結構描述登錄檔中刪除可共用的測試事件。

如需詳細資訊，請參閱 [使用 進行雲端測試的簡介 sam remote test-event](#)。

將可共用的測試事件傳遞至雲端中的 Lambda 事件函數

若要將可共用測試事件從 EventBridge 結構描述登錄檔傳遞至雲端中的 Lambda 函數，請使用 `--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

您可以將訊息傳送至 Amazon SQS 佇列。AWS SAM CLI 會傳回下列項目：

- 訊息 ID
- 訊息內文的 MD5
- 回應中繼資料

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 IDs 的資訊，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 `--event` 選項在命令列提供事件，或使用從檔案提供事件 `--event-file`。如果您不提供事件，AWS SAM CLI 會傳送空事件。

使用 搭配 Step Functions

您可以叫用 Step Functions 狀態機器來開始執行。AWS SAM CLI 會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"%
```

叫用狀態機器

1. 提供資源 ID 值做為 Step Functions 狀態機器的引數。如需有效資源 IDs 的資訊，請參閱[資源 ID](#)。
2. 提供事件以傳送至您的狀態機器。您可以使用 `--event` 選項在命令列提供事件，或使用 `從 檔案` 提供事件 `--event-file`。如果您不提供事件，AWS SAM CLI 會傳送空事件。

使用 sam 遠端叫用命令選項

本節涵蓋您可以搭配 `sam remote invoke` 命令使用的一些主要選項。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

使用 在命令列 `--event` 傳遞事件做為字串值：

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --event '{"message": "hello!"}'
```

Invoking Lambda Function HelloWorldFunction

```
START RequestId: b992292d-1fac-4aa2-922a-c9dc5c6fceb Version: $LATEST
END RequestId: b992292d-1fac-4aa2-922a-c9dc5c6fceb
REPORT RequestId: b992292d-1fac-4aa2-922a-c9dc5c6fceb 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!\"}"%
```

使用 從檔案 `--event-file` 傳遞事件，並提供檔案的路徑：

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

設定 AWS SAM CLI 回應輸出

當您使用 叫用支援的資源時 `sam remote invoke` , AWS SAM CLI 會傳回包含下列項目的回應 :

- 請求中繼資料 – 與請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 – 在雲端中調用 資源後的回應。

您可以使用 `--output` 選項來設定 AWS SAM CLI 輸出回應。下列選項值可供使用 :

- `json` – 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` – 中繼資料會以文字結構傳回。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": "$LATEST",
      "x-amz-log-result":
        "U1RBULQgUmVxdWVzdElkOiAzYmRmOWEzMC03NzZkLTRhOTAtOTRhNi00Y2NjYzBmYzdiNDEgVmVyc2lvbjoJExBVEVTV
        "x-amzn-trace-id":
        "root=1-64908d42-17dab270273fcc6b527dd6b8;samled=0;lineage=2301f8dc:0"
    },
    "RetryAttempts": 0
  },
  "StatusCode": 200,
  "LogResult":
    "U1RBULQgUmVxdWVzdElkOiAzYmRmOWEzMC03NzZkLTRhOTAtOTRhNi00Y2NjYzBmYzdiNDEgVmVyc2lvbjoJExBVEVTV
    "ExecutedVersion": "$LATEST",
    "Payload": "{\"statusCode\":200,\"body\": \"{\\\"message\\\":\\\"hello world\\\"}\"}"
}%
```

當您指定`json`輸出時，整個回應會傳回給 `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": "$LATEST",
      "x-amz-log-result":
        "U1RBULQgUmVxdWVzdElkOiBkMzBkMjgwZi04MTg4LTQzNzItYmM5NC1jZTBmMTYwM2I2YmIgVmVyc2lvbjogJExBVEVTV
        "x-amzn-trace-id":
        "root=1-649091fc-771473c7778689627a6122b7;sampld=0;lineage=2301f8dc:0"
    },
    "RetryAttempts": 0
  },
  "StatusCode": 200,
  "LogResult":
    "U1RBULQgUmVxdWVzdElkOiBkMzBkMjgwZi04MTg4LTQzNzItYmM5NC1jZTBmMTYwM2I2YmIgVmVyc2lvbjogJExBVEVTV
    "ExecutedVersion": "$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 函數執行期輸出（例如，日誌）會傳回 stderr。Lambda 函數承載會傳回 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會利用適用於 Python (Boto3) 的 AWS SDK 與雲端中的資源互動。您可以使用 `--parameter` 選項來自訂Boto3參數。如需您可以自訂的支援參數清單，請參閱 [--parameter](#)。

範例

叫用 Lambda 函數來驗證參數值和驗證許可：

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --  
parameter InvocationType="DryRun"
```

在單一命令中多次使用 `--parameter` 選項，以提供多個參數：

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --  
parameter InvocationType="Event" --parameter LogType="None"
```

其他選項

如需 `sam remote invoke` 選項的完整清單，請參閱 [sam remote invoke](#)。

設定您的專案組態檔案

若要 `sam remote invoke` 在組態檔案中設定，請在 資料表 `remote_invoke` 中使用。以下是 `samconfig.toml` 設定 `sam remote invoke` 命令預設值的檔案範例。

```
...  
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 運算部落格中的 [使用 AWS SAM 遠端 測試 AWS Lambda 函數](#)。

Kinesis Data Streams 範例

基本範例

從 檔案將資料記錄傳送至 Kinesis Data Streams 應用程式。Kinesis Data Streams 應用程式是透過提供資源 ID 的 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，在命令列提供事件：

```
$ 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"
}%
```

在命令列提供 JSON 字串做為事件：

```
$ sam remote invoke KinesisStream --stack-name kinesis-example --event '{"method":
"GET", "body": ""}'
```

Putting record to Kinesis data stream KinesisStream

```
{
  "ShardId": "shardId-000000000000",
  "SequenceNumber": "49646251411914806775980904492868617924990209230536441858"
}%
```

將空事件傳送至 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"
}%
```

以 AWS SAM CLI JSON 格式傳回回應：

```
$ 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": "Q3yBcgTwtPaQTV26IKclbECmZikUY0zKY+CzcxA84ZHgCkc5T2N/ITWg6RP0QcWw8Gn0tNPcEJBEHyVVqboJAPgCritqsvCu",
      "date": "Thu, 09 Nov 2023 18:13:10 GMT",
      "content-type": "application/x-amz-json-1.1",
      "content-length": "110"
    },
    "RetryAttempts": 0
  }
}
```

```
}%
```

將 JSON 輸出傳回 stdout：

```
$ sam remote invoke KinesisStream --stack-name kinesis-example --event '{"hello":  
"world"}' --output json 1 > stdout.log
```

Putting record to Kinesis data stream KinesisStream

```
$ cat stdout.log  
{  
  "ShardId": "shardId-000000000000",  
  "SequenceNumber": "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

透過提供 ARN 做為資源 ID 來叫用 Lambda 函數：

```
$ sam remote invoke arn:aws:lambda:us-west-2:012345678910:function:sam-app-  
HelloWorldFunction-ohRFEn2RuAvp
```

透過提供邏輯 ID 做為資源 ID 來叫用 Lambda 函數：

您還必須使用 `--stack-name` 選項提供 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，則不必指定其邏輯 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 ID 來叫用 Lambda 函數：

實體 ID 會在您使用 部署時建立 CloudFormation。

```
$ sam remote invoke sam-app-HelloWorldFunction-TZvxQRFNv0k4
```

叫用子堆疊的 Lambda 函數：

在此範例中，我們的應用程式包含下列目錄結構：

```
lambda-example
### childstack
#   ### function
#   #   ### __init__.py
#   #   ### app.py
#   #   ### requirements.txt
#   ### template.yaml
### events
#   ### event.json
### samconfig.toml
### template.yaml
```

若要叫用的 Lambda 函數 `childstack`，我們會執行下列動作：

```
$ 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 事件函數

在此範例中，我們使用 AWS SAM CLI來初始化新的無伺服器應用程式，其中包含設定為串流其回應的 Lambda 函數。我們將應用程式部署到 AWS 雲端，並使用 `sam remote invoke` 與雲端中的函數互動。

首先執行 `sam init` 命令來建立新的無伺服器應用程式。我們會選取 Lambda 回應串流快速入門範本，並將應用程式命名為 `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`

會使用下列結構 AWS SAM CLI 建立我們的專案：

```
lambda-streaming-nodejs-app
### README.md
### __tests__
#   ### unit
#       ### index.test.js
### package.json
### samconfig.toml
### src
#   ### index.js
### template.yaml
```

以下是 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` 檔案範例。使用 `FunctionUrlConfig` 屬性設定 Lambda 函數的回應串流。

```
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
```

一般而言，您可以使用 `sam build` 和 `sam deploy --guided` 來建置和部署生產應用程式。在此範例中，我們將假設開發環境，並使用 `sam sync` 命令來建置和部署我們的應用程式。

Note

建議開發環境使用 `sam sync` 命令。如需詳細資訊，請參閱 [使用 sam sync 同步至 的簡介 AWS 雲端](#)。

在執行 `aws sam sync` 之前，我們會驗證我們的專案是否已在 `samconfig.toml` 檔案中正確設定。最重要的是，我們會驗證 `stack_name` 和 `watch` 的值。在我們的組態檔案中指定了這些值，我們就不必在命令列提供這些值。

```
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，我們會執行 `aws sam sync` 來建置和部署應用程式。由於 `--watch` 選項是在我們的組態檔案中設定，因此 AWS SAM CLI 會建置我們的應用程式、部署我們的應用程式，並留意變更。

```
$ aws 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_pt15g6_80000gr/T/tmpavrzdhgp.

Execute the following command to deploy the packaged template

```
sam deploy --template-file /var/folders/45/5ct135bx3fn2551_pt15g6_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-samclisam-s3-demo-
bucket-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 ResourceStatusReason	ResourceType	LogicalResourceId	
CREATE_IN_PROGRESS Transformation	AWS::CloudFormation::Stack	lambda-streaming-	
succeeded	ack	nodejs-app	
CREATE_IN_PROGRESS	AWS::IAM::Role	StreamingFunctionRole	-
CREATE_IN_PROGRESS	AWS::CloudFormation::Stack	AwsSamAutoDependencyLayerNestedStack	-
CREATE_IN_PROGRESS creation	AWS::IAM::Role	StreamingFunctionRole	Resource
Initiated CREATE_IN_PROGRESS creation	AWS::CloudFormation::Stack	AwsSamAutoDependencyLayerNestedStack	Resource
Initiated CREATE_COMPLETE	AWS::IAM::Role	StreamingFunctionRole	-
CREATE_COMPLETE	AWS::CloudFormation::Stack	AwsSamAutoDependencyLayerNestedStack	-
CREATE_IN_PROGRESS	AWS::Lambda::Function	StreamingFunction	-
CREATE_IN_PROGRESS creation	AWS::Lambda::Function	StreamingFunction	Resource
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-gUmh0833A0vZ

Key                      StreamingFunctionURL

Description              Streaming Lambda Function URL

Value                   https://wxgkcc2dyntgtrwhf2dgdcvy1u0rnnof.lambda-url.us-
west-2.on.aws/
-----

Stack creation succeeded. Sync infra completed.

Infra sync completed.
```

現在我們的函式已部署到雲端，我們可以使用 `sam remote invoke` 與函式互動。AWS SAM CLI 會自動偵測我們的函數已設定為回應串流，並立即開始即時輸出函數的串流回應。

```
$ 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
```

當我們修改函數程式碼時，會 AWS SAM CLI 立即偵測並立即部署變更。以下是變更函數程式碼後輸出 AWS SAM CLI 的範例：

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.
```

我們現在可以 `sam remote invoke` 再次使用 與雲端中的 函式互動，並測試我們的變更。

SQS 範例

基本範例

透過提供 ARN 做為資源 ID 來叫用 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

透過提供其實體 ID 做為資源 ID 來叫用狀態機器：

首先，我們使用 `sam list resources` 來取得實體 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"
  }
]
```

接著，我們使用實體 ID 做為資源 ID 來調用狀態機器。我們透過 `--event` 選項在命令列傳入事件：

```
$ 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"%
```

透過傳遞空事件來叫用狀態機器：

```
$ sam remote invoke HelloWorldStateMachine --stack-name state-machine-example
```

```
Invoking Step Function HelloWorldStateMachine
```

```
"Hello World"%
```

相關連結

如需 `sam remote invoke` 與 使用 相關的文件 AWS SAM CLI，請參閱下列內容：

- [sam remote invoke](#)
- [AWS SAM CLI 故障診斷](#)

使用 自動化本機整合測試 AWS SAM

雖然您可以使用 [使用 進行測試的簡介](#) `sam local invoke` 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，您必須先針對本機 Lambda 事件函數執行測試，然後再部署到 AWS 雲端。[使用 進行測試的簡介](#) `sam local start-lambda` 命令會啟動模擬 Lambda 調用端點的本機端點。您可以從自動化測試叫用它。由於此端點模擬 Lambda 叫用端點，因此您可以撰寫測試一次，然後針對本機 Lambda 函數或已部署的 Lambda 函數執行測試（無需任何修改）。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。

這是程序的運作方式：

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，啟動本機 Lambda 端點：

```
sam local start-lambda
```

此命令會在 `http://127.0.0.1:3001` 模擬時啟動本機端點 AWS Lambda。您可以針對此本機 Lambda 端點執行自動化測試。當您使用 AWS CLI 或 SDK 叫用此端點時，它會在本機執行請求中指定的 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 來使用測試資料叫用 Lambda 函數、等待回應，並確認回應是您所預期的。若要在本機執行整合測試，您應該設定 AWS SDK 傳送 Lambda 調用 API 呼叫，以調用您在上一個步驟中啟動的本機 Lambda 端點。

以下是 Python 範例（其他語言的 AWS SDKs 具有類似的組態）：

```
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"
```

您可以使用此程式碼，透過將 `running_locally` 設定為 `False` 來測試已部署的 Lambda 函數。這會將 AWS SDK 設定為在 AWS 雲端中連線至 AWS Lambda。

使用 產生範例事件承載 AWS SAM

若要測試 Lambda 事件函數，您可以產生和自訂範例事件承載，以模擬 Lambda 函數在由其他服務觸發時將接收的資料 AWS。這包括 API Gateway、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。相較於手動建立 AWS 服務事件範例以測試函數，此方法也會節省時間。

如需可產生範例事件承載的服務完整清單，請使用下列命令：

```
sam local generate-event --help
```

如需可用於特定服務的選項清單，請使用下列命令：

```
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
```

測試和偵錯耐用函數

在本機測試和偵錯耐用函數的運作方式與一般 Lambda 函數類似，並自動支援檢查點和重播。本指南涵蓋常見的測試案例和故障診斷技術。

本機測試工作流程

在本機測試耐用函數時，工作流程與一般 Lambda 函數不同：

耐用的函數測試工作流程

1. 建置您的應用程式：

```
$ sam build
```

2. 叫用您的耐久函數：

```
$ sam local invoke MyDurableFunction --durable-execution-name test
```

3. 如有需要，請檢查執行歷史記錄：

```
$ sam local execution history execution-id
```

常見測試案例

測試檢查點行為

若要測試您的函數是否正確檢查點狀態：

```
# Example Python durable function
```

```
def handler(event, context):  
    # This will create a checkpoint  
    context.wait(10) # Wait 10 seconds  
  
    # Function resumes here after wait  
    return {"message": "Function resumed after wait"}
```

當您在本機叫用此函數時，會自動處理等待期間。

測試回呼案例

對於等待外部回呼的函數：

1. 啟動等待回呼的耐久函數
2. 在另一個終端機中，解析回呼：

```
$ sam local callback succeed callback-id
```

3. 觀察函數繼續執行

疑難排解

耐用函數未正確執行

問題：函數無法做為持久的函數。

解決方案：

- 確認 DurableConfig 已在您的 SAM 範本中設定
- 確保您的函數程式碼使用耐用的函數 SDK 方法（例如 `context.wait()`）
- 檢查您是否使用支援的執行期 (TypeScript、JavaScript、Python)

無法擷取執行歷史記錄

問題：`local execution history` 命令不會傳回任何結果。

解決方案：

- 驗證執行 ID 是否正確

- 檢查是否已調用該函數至少一次

回呼命令無法運作

問題：回呼命令無法解決待處理的操作。

解決方案：

- 驗證回呼 ID 是否正確
- 確保函數實際正在等待回呼
- 確認您使用的是正確的回呼命令語法

偵錯秘訣

- 使用執行歷史記錄 - 檢閱執行歷史記錄，以了解您耐用函數的流程
- 遞增測試 - 在新增複雜邏輯之前，從簡單的等待操作開始
- 使用詳細記錄 - 啟用詳細記錄以追蹤執行流程

進一步了解

如需測試和偵錯的詳細資訊，請參閱：

- [使用 進行測試的簡介 `sam local invoke`](#) - 本機調用文件
- [sam local execution history](#) - 執行歷史記錄

使用 偵錯無伺服器應用程式 AWS SAM

測試應用程式之後，您就可以偵錯發現的任何問題。使用 AWS SAM 命令列界面 (CLI)，您可以在本機測試和偵錯無伺服器應用程式，然後再將其上傳至 AWS 雲端。偵錯您的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 執行逐步偵錯，這是一種一次執程式碼一行或指令的方法。當您在內以偵錯模式本機叫用 Lambda 事件函數時 AWS SAM CLI，您可以將偵錯工具連接至該函數。使用除錯器，您可以逐步逐行瀏覽程式碼、查看不同變數的值，以及修正問題的方式與任何其他應用程式相同。您可以在進行封裝和部署應用程式的步驟之前，驗證應用程式的行為是否如預期、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，當您在本機執行和偵錯應用程式時，會下載層套件並在本機主機上快取。如需詳細資訊，請參閱[如何在本機快取層](#)。

主題

- [使用 本機偵錯函數 AWS SAM](#)
- [使用 偵錯時傳遞多個執行時間引數 AWS SAM](#)
- [使用 CloudFormation Linter 驗證您的 AWS SAM 應用程式](#)

使用 本機偵錯函數 AWS SAM

您可以使用 AWS SAM 搭配各種 AWS 工具組和偵錯工具，在本機測試和偵錯無伺服器應用程式。Lambda 事件函數的逐步偵錯可讓您在本地環境中一次識別和修正應用程式中的問題一行或指令。

您可以執行本機逐步偵錯的一些方式包括設定中斷點、檢查變數，以及一次一行執行函數程式碼。本機逐步偵錯可讓您在雲端找到可能遇到的問題並進行故障診斷，藉此加強回饋迴圈。

您可以使用 AWS Toolkits 進行偵錯，也可以 AWS SAM 在偵錯模式下執行。如需詳細資訊，請參閱本節中的主題。

使用 AWS 工具組

AWS 工具組是整合的開發環境 (IDE) 外掛程式，可讓您一次一行執行許多常見的偵錯任務，例如設定中斷點、檢查變數和執行函數程式碼。AWS 工具組可讓您更輕鬆地開發、偵錯和部署使用 建置的無

伺服器應用程式 AWS SAM。它們提供建置、測試、偵錯、部署和叫用整合到您的 IDE 的 Lambda 函數的體驗。

如需可與 搭配使用之 AWS Toolkit 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](#)

有多種 AWS 工具組可搭配不同的 IDEs 和執行時間組合使用。下表列出支援 AWS SAM 應用程式逐步偵錯的常見 IDE/執行時間組合：

IDE	執行時期	AWS 工具組	步驟式偵錯的說明
Visual Studio 程式碼	<ul style="list-style-type: none"> • Node.js • Python • .NET • Java • Go 	AWS Toolkit for Visual Studio Code	AWS Toolkit for Visual Studio Code 《使用者指南》中的 使用 AWS 無伺服器應用程式
AWS Cloud9	<ul style="list-style-type: none"> • Node.js • Python 	AWS Cloud9，啟用 AWS Toolkit ¹	使用AWS Cloud9 AWS 《使用者指南》 中的 Toolkit 使用無 AWS 伺服器應用程式。
WebStorm	Node.js	AWS Toolkit for JetBrains ²	在 中執行（叫用）或偵錯本機函數 AWS Toolkit for JetBrains
PyCharm	Python	AWS Toolkit for JetBrains ²	在 中執行（叫用）或偵錯本機函數 AWS Toolkit for JetBrains
Rider	.NET	AWS Toolkit for JetBrains ²	在 中執行（叫用）或偵錯本機函數 AWS Toolkit for JetBrains

IDE	執行時期	AWS 工具組	步驟式偵錯的說明
IntelliJ	Java	AWS Toolkit for JetBrains ²	在 中執行（叫用） 或偵錯本機函數 AWS Toolkit for JetBrains
GoLand	Go	AWS Toolkit for JetBrains ²	在 中執行（叫用） 或偵錯本機函數 AWS Toolkit for JetBrains

備註：

- 若要使用 AWS Cloud9 逐步偵錯 AWS SAM 應用程式，必須啟用 AWS Toolkit。如需詳細資訊，請參閱AWS Cloud9 《使用者指南》中的[啟用 AWS 工具組](#)。
- 若要使用 AWS Toolkit for JetBrains 逐步偵錯 AWS SAM 應用程式，您必須先依照在 [中安裝](#) 中的指示[進行安裝 AWS Toolkit for JetBrains](#)和設定AWS Toolkit for JetBrains。

在偵錯模式下於 AWS SAM 本機執行

除了與 AWS Toolkits 整合之外，您也可以 AWS SAM 在偵錯模式下執行，以連接到第三方偵錯工具，例如 [ptvsd](#) 或 [delve](#)。

若要 AWS SAM 在偵錯模式下執行，請使用 命令[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 執行個體會公開所有 Lambda 函數。不過，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 SAM CLI 繫結至連接埠之前呼叫 API，以允許偵錯工具連線。

使用 偵錯時傳遞多個執行時間引數 AWS SAM

您可以選擇將其他執行時間引數傳遞給 AWS SAM，以更有效地檢查問題和疑難排解變數。這樣做可為偵錯程序提供額外的控制和彈性，這可協助您自訂執行期組態和環境。

若要在偵錯函數時傳遞其他執行期引數，請使用環境變數 `DEBUGGER_ARGS`。這會將引數字串直接傳遞至 AWS SAM CLI 用來啟動函數的執行命令。

例如，如果您想要在 Python 函數的執行時間載入 iKpdb 之類的偵錯工具，您可以將以下內容傳遞為 `DEBUGGER_ARGS`：`-m ikpdb --ikpdb-port=5858 --ikpdb-working-directory=/var/task/ --ikpdb-client-working-directory=/myApp --ikpdb-address=0.0.0.0`。這會在執行時間使用您指定的其他引數載入 iKpdb。

在這種情況下，您的完整 AWS SAM CLI 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
```

您可以將除錯器引數傳遞至所有執行時間的函數。

使用 CloudFormation Linter 驗證您的 AWS SAM 應用程式

AWS CloudFormation Linter (cfn-lint) 是一種開放原始碼工具，可用來對 CloudFormation 範本執行詳細的驗證。Cfn-lint 包含由 CloudFormation 資源規格引導的規則。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、警告或資訊建議的詳細訊息。或者，建立自己的自訂規則進行驗證。若要進一步了解 cfn-lint，請參閱 CloudFormation GitHub 儲存庫中的 [cfn-lint](#)。

您可以使用 cfn-lint 透過 AWS SAM 命令列界面 AWS Serverless Application Model (AWS SAM) 驗證您的 (AWS SAM CLI) 範本，方法是 `sam validate` 使用 `--lint` 選項執行。

```
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，您可以定義組態檔案。若要進一步了解，請參閱 cfn-lint CloudFormation GitHub 儲存庫中的[組態檔案](#)。當您執行 `sam validate --lint`，將會套用組態檔案中定義的 cfn-lint 行為。

範例

在 AWS SAM 範本上執行 cfn-lint 驗證

```
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](#)。

使用 部署您的應用程式和資源 AWS SAM

部署您的應用程式會在 AWS 雲端佈建和設定您的 AWS 資源，讓您的應用程式在 雲端中執行。AWS SAM 會使用您在執行 `sam build` 命令時建立的 [CloudFormation](#) AWS SAM 建置成品做為部署無伺服器應用程式的標準輸入。

使用 AWS SAM，您可以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執行的步驟，用於發行無伺服器應用程式的新版本。

本節中的主題提供自動化和手動部署的指引。若要手動部署應用程式，您可以使用 AWS SAM CLI 命令。若要自動化部署，請參閱本節中的主題。它們專門提供使用管道和 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) 使用者身分驗證，以及在部署時上傳本機檔案。

主題

- [使用 部署簡介 AWS SAM](#)
- [使用 部署應用程式的選項 AWS SAM](#)
- [使用 CI/CD 系統和管道搭配 部署 AWS SAM](#)
- [使用 `sam sync` 同步至 的簡介 AWS 雲端](#)

使用 部署簡介 AWS SAM

使用 AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam deploy` 命令將您的無伺服器應用程式部署到 AWS 雲端。

- 如需 的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)。
- 如需 `sam deploy` 命令選項的清單，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

- [先決條件](#)
- [使用 `sam` 部署來部署應用程式](#)

- [最佳實務](#)
- [sam 部署的選項](#)
- [疑難排解](#)
- [範例](#)
- [進一步了解](#)

先決條件

若要使用 `sam deploy`，請完成下列步驟來 AWS SAM CLI 安裝：

- [AWS SAM 先決條件](#).
- [安裝 AWS SAM CLI](#).

使用之前 `sam deploy`，我們建議您對以下內容有基本的了解：

- [設定 AWS SAM CLI](#).
- [在中建立您的應用程式 AWS SAM](#).
- [使用建置簡介 AWS SAM](#).

使用 sam 部署來部署應用程式

當您第一次部署無伺服器應用程式時，請使用 `--guided` 選項。AWS SAM CLI 將引導您完成互動式流程，以設定應用程式的部署設定。

使用互動式流程部署應用程式

1. 前往專案的根目錄。這與 AWS SAM 範本的位置相同。

```
$ cd sam-app
```

2. 執行以下命令：

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

3. 在互動式流程中，AWS SAM CLI 會提示您設定應用程式部署設定的選項。

括號 ([]) 表示預設值。將您的答案保留空白以選取預設值。預設值是從下列組態檔案取得：

- `~/.aws/config` – 您的一般 AWS 帳戶設定。
- `~/.aws/credentials` – AWS 您的帳戶登入資料。
- `<project>/samconfig.toml` – 專案的組態檔案。

回答 AWS SAM CLI 提示以提供值。例如，您可以針對是、否或字串值輸入 **n**。

會將您的回應 AWS SAM CLI 寫入專案的 `samconfig.toml` 檔案。對於後續部署，您可以使用 `sam deploy` 來使用這些設定值進行部署。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。

以下是輸出範例：

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

Configuring SAM deploy
=====

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

    Setting default arguments for 'sam deploy'
    =====
    Stack Name [sam-app]: 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. 接著，AWS SAM CLI會將您的應用程式部署到 AWS 雲端。在部署期間，進度會顯示在命令提示中。以下是部署的主要階段：
- 對於 AWS Lambda 事件函數封裝為 .zip 檔案封存的應用程式，AWS SAM CLI會壓縮套件並將其上傳至 Amazon Simple Storage Service (Amazon S3) 儲存貯體。如有必要，AWS SAM CLI 會建立新的儲存貯體。
 - 對於使用 Lambda 函數套件做為容器映像的應用程式，AWS SAM CLI會將映像上傳至 Amazon Elastic Container Registry (Amazon ECR)。如有必要，AWS SAM CLI會建立新的儲存庫。
 - AWS SAM CLI 會建立 AWS CloudFormation 變更集，並將您的應用程式部署到 CloudFormation 做為堆疊。
 - 會使用 Lambda AWS SAM CLI函數的新CodeUri值來修改已部署的 AWS SAM 範本。

以下是部署輸出的範例 AWS SAM CLI：

```
Looking for resources needed for deployment:

Managed S3 bucket: aws-sam-cli-managed-default-samclisam-s3-demo-
bucket-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-samclisam-s3-
demo-bucket-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 ResourceStatusReason	ResourceType	LogicalResourceId	
CREATE_IN_PROGRESS	AWS::IAM::Role	HelloWorldFunctionRole	-
CREATE_IN_PROGRESS creation	AWS::IAM::Role	HelloWorldFunctionRole	Resource
Initiated			
CREATE_COMPLETE	AWS::IAM::Role	HelloWorldFunctionRole	-
CREATE_IN_PROGRESS	AWS::Lambda::Function	HelloWorldFunction	-
CREATE_IN_PROGRESS creation	AWS::Lambda::Function	HelloWorldFunction	Resource
Initiated			
CREATE_COMPLETE	AWS::Lambda::Function	HelloWorldFunction	-
CREATE_IN_PROGRESS	AWS::ApiGateway::RestA	ServerlessRestApi	-
	pi		
CREATE_IN_PROGRESS creation	AWS::ApiGateway::RestA	ServerlessRestApi	Resource

Initiated	pi			
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-11Z0GSCG28H0M
```

```
Key                HelloWorldApi
```

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

```
Value              https://njzfhdm1s0.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. 使用 URL 直接開啟 CloudFormation 主控台 [https : //console.aws.amazon.com/cloudformation](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並選取它。

在部署之前驗證變更

您可以設定 AWS SAM CLI 以顯示 CloudFormation 變更集，並在部署之前要求確認。

在部署之前確認變更

1. 在期間 `sam deploy --guided`，輸入 **Y** 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，您可以使用下列項目修改 `samconfig.toml` 檔案：

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

2. 在部署期間，AWS SAM CLI 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。您可以透過修改 AWS SAM 範本並在部署期間設定參數值來執行此操作。

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`。以下是輸出範例：

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

Configuring SAM deploy
```



```

=====

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

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

```

為您的 Lambda 事件函數設定程式碼簽署

您可以在部署時為 Lambda 函數設定程式碼簽署。您可以透過修改 AWS SAM 範本並在部署期間設定程式碼簽署來執行此操作。

設定程式碼簽署

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 SAM CLI 將提示您設定程式碼簽署。以下是輸出範例：

```

#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 `sam deploy`，AWS SAM CLI 會部署您的應用程式在 `.aws-sam` 目錄中的建置成品。當您變更應用程式的原始檔案時，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，您可以使用 `sam deploy` 搭配設定的 設定進行部署。

sam 部署的選項

以下是 常用的選項 `sam deploy`。如需所有選項的清單，請參閱 [sam deploy](#)。

使用引導式互動式流程來部署您的應用程式

使用 `--guided` 選項，透過互動式流程來設定應用程式的部署設定。以下是範例：

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

您應用程式的部署設定會儲存在專案的 `samconfig.toml` 檔案中。如需詳細資訊，請參閱 [設定專案設定](#)。

疑難排解

若要對 進行故障診斷 AWS SAM CLI，請參閱 [AWS SAM CLI 故障診斷](#)。

範例

部署 Hello World 應用程式，其中包含封裝為 `.zip` 檔案封存檔的 Lambda 事件函數

如需範例，請參閱 Hello World 應用程式教學 [步驟 3：將您的應用程式部署到 AWS 雲端](#) 中的。

部署 Hello World 應用程式，其中包含封裝為容器映像的 Lambda 事件函數

首先，我們使用 `sam init` 來建立 Hello World 應用程式。在互動式流程中，我們會選擇 Python3.9 執行時間和 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，我們會cd移至專案的根目錄，並執行 `sam build`。使用在 AWS SAM CLI本機建置我們的 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-
manylinux2_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，我們會執行 `sam deploy --guided` 來部署應用程式。會 AWS SAM CLI 引導我們設定部署設定。然後，AWS SAM CLI 會將我們的應用程式部署到 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-samclisam-s3-demo-
bucket-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-samclisam-s3-demo-
bucket-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 oWorldPermissionProd	AWS::Lambda::Permissio n	N/A
+ Add	HelloWorldFunctionRole	AWS::IAM::Role	N/A
+ Add	HelloWorldFunction	AWS::Lambda::Function	N/A
+ Add	ServerlessRestApiDeplo yment47fc2d5f9d	AWS::ApiGateway::Deplo yment	N/A
+ 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
Initiated                                     User

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

```

                                tage
CREATE_COMPLETE      AWS::Lambda::Permissio 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-
JFML1J0KHJ71	
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

若要進一步了解如何使用 AWS SAM CLI `sam deploy` 命令，請參閱以下內容：

- [完成 AWS SAM 研討會：單元 3 - 手動部署](#) – 了解如何使用 建置、封裝和部署無伺服器應用程式 AWS SAM CLI。

使用 部署應用程式的選項 AWS SAM

使用 AWS SAM，您可以手動部署應用程式並自動化部署。使用 AWS SAM CLI 手動部署您的應用程式。若要自動化部署，請使用管道和持續整合和持續部署 (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 AWS SAM CLI 手動部署](#)
- [使用 CI/CD 系統和管道部署](#)
- [逐步部署](#)
- [使用 對部署進行故障診斷 AWS SAM CLI](#)
- [進一步了解](#)

如何使用 AWS SAM CLI 手動部署

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](#) 命令部署應用程式。

若要讓 使用提示 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` 命令會壓縮應用程式成品、將它們上傳至 Amazon Simple Storage Service (Amazon S3)（適用於 .zip 檔案封存）或 Amazon Elastic Container Registry (Amazon ECR)（適用於容器映像）。命令接著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 系統和管道部署

AWS SAM 可協助您使用管道和持續整合和持續部署 (CI/CD) 系統來自動化部署。AWS SAM 可用來建立管道，並簡化無伺服器應用程式的 CI/CD 任務。多個 CI/CD 系統支援 AWS SAM 建置容器映像，AWS SAM 也為封裝部署最佳實務 AWS 的多個 CI/CD 系統提供一組預設管道範本。

如需詳細資訊，請參閱[使用 CI/CD 系統和管道搭配 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。如需詳細資訊，請參閱 AWS CodeDeploy 《使用者指南》中的[在 CodeDeploy 中使用部署組態](#)。

如需設定 AWS SAM 應用程式以逐步部署的資訊，請參閱[使用 逐步部署無伺服器應用程式 AWS SAM](#)。

使用 對部署進行故障診斷 AWS SAM CLI

AWS SAM CLI 錯誤：「未滿足安全限制條件」

執行時 `sam deploy --guided`，系統會提示您輸入問題 `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在未經授權的情況下設定 Amazon API Gateway API。透過**N**回應此提示，您表示這不行。

若要修正此問題，您有下列選項：

- 使用授權設定您的應用程式。如需設定授權的資訊，請參閱[使用 AWS SAM 範本控制 API 存取](#)。
- 使用 回應此問題，**Y**表示您可以部署已設定 API Gateway API 的應用程式，無需授權。

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 The Complete AWS SAM Workshop 中的以下內容：

- [單元 3 - 手動部署](#) – 了解如何使用 建置、封裝和部署無伺服器應用程式 AWS SAM CLI。
- [單元 4 - CI/CD](#) – 了解如何透過建立持續整合和交付 (CI/CD) 管道來自動化建置、套件和部署階段。

使用 CI/CD 系統和管道搭配 部署 AWS SAM

AWS SAM 協助組織為其偏好的 CI/CD 系統建立管道，以便能夠以最少的努力實現 CI/CD 的優勢，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 透過建置容器映像的協助，簡化無伺服器應用程式的 CI/CD 任務。AWS SAM 提供的映像包含 AWS SAM CLI 多個支援 AWS Lambda 執行時間的 和 建置工具。這可讓您更輕鬆地使用 建置和封裝無伺服器應用程式 AWS SAM CLI。這些映像也減輕了團隊為 CI/CD 系統建立和管理自己映像的需求。如需 AWS SAM 建置容器映像的詳細資訊，請參閱[映像儲存庫 AWS SAM](#)。

多個 CI/CD 系統支援 AWS SAM 建置容器映像。您應該使用的 CI/CD 系統取決於幾個因素。這些包括您的應用程式是使用單一執行時間還是多個執行時間，還是您要在容器映像內或直接在主機上建置應用程式，可以是虛擬機器 (VM) 或裸機主機。

AWS SAM 也為封裝部署最佳實務 AWS 的多個 CI/CD 系統提供一組預設管道範本。這些預設管道範本使用標準 JSON/YAML 管道組態格式，而內建的最佳實務有助於執行多帳戶和多區域部署，並確認管道無法對基礎設施進行意外變更。

您有兩個主要選項可使用 AWS SAM 來部署無伺服器應用程式：1) 修改現有管道組態以使用 AWS SAM CLI 命令，或 2) 產生範例 CI/CD 管道組態，您可以將其做為自有應用程式的起點。

主題

- [什麼是管道？](#)
- [如何在部署時 AWS SAM 上傳本機檔案](#)
- [使用 產生入門 CI/CD 管道 AWS SAM](#)
- [如何使用 自訂入門管道 AWS SAM](#)
- [自動化 AWS SAM 應用程式的部署](#)
- [如何搭配 AWS SAM 管道使用 OIDC 身分驗證](#)

什麼是管道？

管道是自動執行的步驟序列，用於發行應用程式的新版本。透過 AWS SAM，您可以使用許多常見的 CI/CD 系統來部署應用程式，包括 [AWS CodePipeline](#)、[Jenkins](#)、[GitLab CI/CD](#) 和 [GitHub 動作](#)。

管道範本包含 AWS 部署最佳實務，可協助多帳戶和多區域部署。開發和生產等 AWS 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道，而不會對基礎設施進行意外變更。

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。

如何在部署時 AWS SAM 上傳本機檔案

當您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 AWS 的服務，例如 Amazon Simple Storage Service (Amazon S3)。這包括 AWS SAM 範本參考的本機檔案。為了符合此需求，當您使用 `sam deploy` 或 `sam package` 命令時，會 AWS SAM CLI 執行下列動作：

1. 自動將本機檔案上傳至可存取 AWS 的服務。
2. 自動更新您的應用程式範本，以參考新的檔案路徑。

主題

- [示範：使用 AWS SAM CLI 上傳 Lambda 事件函數程式碼](#)
- [支援的使用案例](#)
- [進一步了解](#)

示範：使用 AWS SAM CLI 上傳 Lambda 事件函數程式碼

在此示範中，我們使用 Lambda 事件函數的 .zip 套件類型來初始化範例 Hello World 應用程式。我們使用 AWS SAM CLI 自動將 Lambda 函數程式碼上傳至 Amazon S3，並在應用程式範本中參考其新路徑。

首先，我們會執行 `sam init` 來初始化 Hello World 應用程式。

```
$ 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 函數程式碼會在專案的 `hello_world` 子目錄中組織。

```

demo
### README.md
### hello_world
#   ### __init__.py
#   ### app.py
#   ### requirements.txt
### template.yaml
### tests

```

在我們的 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，我們會執行 `sam build` 來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，我們會執行 `sam deploy --guided` 來部署應用程式。

```
$ 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-samclisam-s3-demo-
bucket-1a4x26zbcdkqr
Capabilities               : ["CAPABILITY_IAM"]
Parameter overrides       : {}
Signing Profiles           : {}

Initiating deployment
=====
...
Waiting for changeset to be created..

```

CloudFormation stack changeset

Operation	LogicalResourceId	ResourceType	Replacement
+ Add	HelloWorldFunctionHelloWorldPermissionProd	AWS::Lambda::Permission	N/A
+ 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 ResourceStatusReason	ResourceType	LogicalResourceId	
CREATE_IN_PROGRESS	AWS::IAM::Role	HelloWorldFunctionRole	-
CREATE_IN_PROGRESS creation	AWS::IAM::Role	HelloWorldFunctionRole	Resource 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，AWS SAM CLI會自動將 Lambda 函數程式碼上傳至 Amazon S3，並更新我們的範本。我們在 CloudFormation 主控台中修改的範本會反映 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-samclisam-s3-demo-bucket-1a4x26zbcdkqr/
demo/da3c598813f1c2151579b73ad788cac8
      Handler: app.lambda_handler
      ...

```

支援的使用案例

對於 AWS SAM CLI多種檔案類型、CloudFormation 資源類型和 CloudFormation 巨集，可以自動促進此程序。

檔案類型

支援應用程式檔案和 Docker 映像。

CloudFormation 資源類型

以下是支援的資源類型及其屬性清單：

資源	Properties
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
AWS::Glue::Job	Command.ScriptLocation
AWS::Lambda::Function	

資源	Properties
	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

CloudFormation 巨集

支援使用AWS::Include轉換巨集參考的檔案。

進一步了解

若要進一步了解AWS::Include轉換，請參閱AWS CloudFormation 《使用者指南》中的 [AWS :: Include 轉換](#)。

若要查看在 AWS SAM 範本中使用AWS::Include轉換的範例，請參閱 Serverless Land 的 [API Gateway HTTP API to SQS](#) 模式。

使用 產生入門 CI/CD 管道 AWS SAM

當您準備好自動化部署時，您可以使用其中一個 AWS SAM 入門管道範本，為您選擇使用的 CI/CD 系統產生部署管道。您的部署管道是您設定和使用來自自動化無伺服器應用程式的部署。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。

透過入門管道範本，您可以使用 [sam pipeline init](#) 命令在幾分鐘內產生管道。

入門管道範本使用 CI/CD 系統熟悉的 JSON/YAML 語法，並納入最佳實務，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許可量。目前，CLI AWS SAM 支援為 [AWS CodePipeline](#)、[Jenkins](#)、[GitLab CI/CD](#)、[GitHub Actions](#) 和 [Bitbucket Pipelines](#) 產生入門 CI/CD 管道組態。[GitHub](#) <https://support.atlassian.com/bitbucket-cloud/docs/get-started-with-bitbucket-pipelines/>

以下是產生入門管道組態所需的高階任務：

1. 建立基礎設施資源 – 您的管道需要特定 AWS 資源，例如具有必要許可的 IAM 使用者和角色、Amazon S3 儲存貯體，以及選用的 Amazon ECR 儲存庫。
2. 將您的 Git 儲存庫與您的 CI/CD 系統連線 – 您的 CI/CD 系統需要知道哪些 Git 儲存庫會觸發管道執行。請注意，視您使用的 Git 儲存庫和 CI/CD 系統組合而定，可能不需要此步驟。
3. 產生管道組態 – 此步驟會產生入門管道組態，其中包含兩個部署階段。
4. 將您的管道組態遞交至 Git 儲存庫 – 此步驟是必要的，以確保您的 CI/CD 系統了解您的管道組態，並在遞交變更時執行。

在您產生入門管道組態並將其遞交至 Git 儲存庫之後，每當有人對該儲存庫遞交程式碼變更時，管道都會觸發自動執行。

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：

- 如果您使用的是 AWS CodePipeline，請參閱 [在 AWS CodePipeline 中產生的入門管道 AWS SAM](#)。
- 如果您使用的是 Jenkins、GitLab CI/CD、GitHub Actions 或 Bitbucket Pipelines，請參閱 [使用 AWS SAM 為 Jenkins、GitLab CI/CD、GitHub Actions、Bitbucket Pipelines 產生入門管道](#)。

在 AWS CodePipeline 中產生的入門管道 AWS SAM

若要產生 的入門管道組態 AWS CodePipeline，請依此順序執行下列任務：

1. 建立基礎設施資源
2. 產生管道組態
3. 將您的管道組態遞交至 Git
4. 將您的 Git 儲存庫與您的 CI/CD 系統連線

 Note

下列程序使用兩個 AWS SAM CLI 命令 [sam pipeline bootstrap](#) 和 [sam pipeline init](#)。有兩個命令的原因是處理使用案例，其中管理員（即需要許可來設定基礎設施 AWS 資源的使用者，例如 IAM 使用者和角色）擁有更多許可，開發人員（即只需要許可來設定個別管道，但不需要必要基礎設施 AWS 資源的使用者）。

步驟 1：建立基礎設施資源

使用的管道 AWS SAM 需要特定 AWS 資源，例如具有必要許可的 IAM 使用者和角色、Amazon S3 儲存貯體，以及選用的 Amazon ECR 儲存庫。您必須針對管道的每個部署階段擁有一組基礎設施資源。

您可以執行下列命令來協助進行此設定：

```
sam pipeline bootstrap
```

 Note

針對管道的每個部署階段執行先前的命令。

步驟 2：產生管道組態

若要產生管道組態，請執行下列命令：

```
sam pipeline init
```

步驟 3：將您的管道組態遞交至 Git 儲存庫

此步驟是必要的，以確保您的 CI/CD 系統知道您的管道組態，並在遞交變更時執行。

步驟 4：將 Git 儲存庫與您的 CI/CD 系統連線

對於 AWS CodePipeline，您現在可以執行下列命令來建立連線：

```
sam deploy -t codepipeline.yaml --stack-name <pipeline-stack-name> --  
capabilities=CAPABILITY_IAM --region <region-X>
```

如果您使用的是 GitHub 或 Bitbucket，則先前執行 `sam deploy` 命令後，請依照開發人員工具主控台使用者指南中的更新待連線主題中的完成連線步驟，完成連線。<https://docs.aws.amazon.com/dtconsole/latest/userguide/connections-update.html> 此外，請存放 `CodeStarConnectionArn` 來自 `sam deploy` 命令輸出的複本，因為如果您想要 AWS CodePipeline 搭配以外的其他分支使用，則需要它 `main`。

設定其他分支

根據預設，AWS CodePipeline 會使用 `main` 分支 AWS SAM。如果您想要使用以外的分支 `main`，您必須再次執行 `sam deploy` 命令。請注意，視您使用的 Git 儲存庫而定，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 管道的實作範例，請參閱完成 AWS SAM 研討會中的[使用 CI/CD AWS CodePipeline](#)。

使用 AWS SAM 為 Jenkins、GitLab CI/CD、GitHub Actions、Bitbucket Pipelines 產生入門管道

若要產生 Jenkins、GitLab CI/CD、GitHub Actions 或 Bitbucket Pipelines 的入門管道組態，請依此順序執行下列任務：

1. 建立基礎設施資源
2. 將您的 Git 儲存庫與您的 CI/CD 系統連線

3. 建立登入資料物件
4. 產生管道組態
5. 將您的管道組態遞交至 Git 儲存庫

Note

下列程序使用兩個 AWS SAM CLI 命令 [sam pipeline bootstrap](#) 和 [sam pipeline init](#)。有兩個命令的原因是處理使用案例，其中管理員（即需要許可來設定基礎設施 AWS 資源的使用者，例如 IAM 使用者和角色）擁有更多許可，開發人員（即只需要許可來設定個別管道，但不需要必要基礎設施 AWS 資源的使用者）。

步驟 1：建立基礎設施資源

使用的管道 AWS SAM 需要特定 AWS 資源，例如具有必要許可的 IAM 使用者和角色、Amazon S3 儲存貯體，以及選用的 Amazon ECR 儲存庫。您必須針對管道的每個部署階段擁有一組基礎設施資源。

您可以執行下列命令來協助進行此設定：

```
sam pipeline bootstrap
```

Note

針對管道的每個部署階段執行先前的命令。

您必須為管道的每個部署階段擷取管道使用者的 AWS 登入資料（金鑰 ID 和私密金鑰），因為後續步驟需要這些登入資料。

步驟 2：將 Git 儲存庫與您的 CI/CD 系統連線

必須將 Git 儲存庫連線至您的 CI/CD 系統，以便 CI/CD 系統能夠存取您的應用程式原始碼以進行建置和部署。

Note

如果您使用下列其中一個組合，則可以略過此步驟，因為會自動為您完成連線：

1. 使用 GitHub 儲存庫的 GitHub 動作
2. 使用 GitLab 儲存庫的 GitLab CI/CD
3. 具有 Bitbucket 儲存庫的 Bitbucket 管道

若要將 Git 儲存庫與您的 CI/CD 系統連線，請執行下列其中一項操作：

- 如果您使用的是 Jenkins，請參閱「新增分支來源」的 [Jenkins 文件](#)。
- 如果您使用的是 GitLab CI/CD 和 GitLab 以外的 GitLab 儲存庫，請參閱「連接外部儲存庫」的 [GitLab 文件](#)。

步驟 3：建立登入資料物件

每個 CI/CD 系統都有自己的方法來管理 CI/CD 系統存取 Git 儲存庫所需的登入資料。

若要建立必要的登入資料物件，請執行下列其中一項操作：

- 如果您使用的是 Jenkins，請建立儲存金鑰 ID 和私密金鑰的單一「憑證」。請遵循[使用部落格建置 Jenkins 管道 AWS SAM](#)一節中的指示。下一個步驟需要 "Credential id"。
- 如果您使用的是 GitLab CI/CD，請建立兩個「受保護的變數」，每個金鑰 ID 和私密金鑰各一個。遵循 [GitLab 文件](#) 中的指示 – 在下一個步驟中，您將需要兩個「可變金鑰」。
- 如果您使用的是 GitHub 動作，請建立兩個「加密的秘密」，每個金鑰和私密金鑰各一個。遵循 [GitHub 文件](#) 中的指示 - 下一個步驟需要兩個「秘密名稱」。
- 如果您使用的是 Bitbucket 管道，請建立兩個「安全變數」，每個金鑰 ID 和私密金鑰各一個。遵循[變數和秘密](#)中的指示 - 下一個步驟需要兩個「秘密名稱」。

步驟 4：產生管道組態

若要產生管道組態，請執行下列命令。您需要輸入您在上一個步驟中建立的登入資料物件：

```
sam pipeline init
```

步驟 5：將您的管道組態遞交至 Git 儲存庫

此步驟是必要的，以確保您的 CI/CD 系統知道您的管道組態，並在遞交變更時執行。

進一步了解

如需使用 設定 CI/CD 管道的實作範例GitHub Actions，請參閱 完成 AWS SAM 研討會中的[使用 CI/CDGitHub](#)。

如何使用 自訂入門管道 AWS SAM

身為 CI/CD 管理員，您可能想要自訂入門管道範本和相關聯的引導式提示，組織中的開發人員可以使用這些提示來建立管道組態。

建立入門範本時，AWS SAM CLI會使用 Cookiecutter 範本。如需 Cookie 切線器範本的詳細資訊，請參閱 [Cookiecutter](#)。

您也可以自訂使用 `sam pipeline init` 命令建立管道組態時，AWS SAM CLI向使用者顯示的提示。若要自訂使用者提示，請執行下列動作：

1. 建立 **questions.json** 檔案 – `questions.json` 檔案必須位於專案儲存庫的根目錄中。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 [questions.json.schema](#)。若要檢視範例 `questions.json` 檔案，請參閱 [questions.json](#)。
2. 使用 Cookiecutter 名稱對應問題金鑰 – `questions.json` 檔案中的每個物件都需要符合 Cookiecutter 範本中名稱的金鑰。此金鑰比對是 將 AWS SAM CLI使用者提示回應映射至 Cookie 切換器範本的方式。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](#) 一節。
3. 建立 **metadata.json** 檔案 – 宣告管道在 `metadata.json` 檔案中的階段數。階段數目會指示 `sam pipeline init` 命令要提示多少階段的資訊，或在 `--bootstrap` 選項的情況下，要為其建立基礎設施資源的階段數量。若要檢視宣告具有兩個階段之管道的範例 `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 Cookiecutter 範本、`questions.json` 檔案和 `metadata.json` 檔案：

- Jenkins 範例：[兩階段 Jenkins 管道範本](#)
- CodePipeline 範例：[兩階段 CodePipeline 管道範本](#)

範例檔案

以下一組檔案顯示 `questions.json` 檔案中的問題如何與 Cookiecutter 範本檔案中的項目相關聯。請注意，這些範例是檔案程式碼片段，而不是完整檔案。若要查看完整檔案的範例，請參閱本主題前面的 [範例專案](#) 一節。

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，自動化 AWS SAM 應用程式部署的方式取決於您使用的 CI/CD 系統。因此，本節中的範例說明如何設定各種 CI/CD 系統，以在建置容器映像中自動化 AWS SAM 建置無伺服器應用程式。這些建置容器映像可讓您更輕鬆地使用 建置和封裝無伺服器應用程式 AWS SAM CLI。

您現有 CI/CD 管道使用 部署無伺服器應用程式的程序 AWS SAM，會因您使用的 CI/CD 系統而略有不同。

下列主題提供設定 CI/CD 系統以在建置容器映像中 AWS SAM 建置無伺服器應用程式的範例：

主題

- [使用 AWS CodePipeline 搭配 部署 AWS SAM](#)
- [使用 Bitbucket 管道搭配 部署 AWS SAM](#)
- [使用 Jenkins 透過 部署 AWS SAM](#)
- [使用 GitLab CI/CD 搭配 部署 AWS SAM](#)
- [使用 GitHub 動作搭配 部署 AWS SAM](#)

使用 AWS CodePipeline 搭配 部署 AWS SAM

若要設定[AWS CodePipeline](#)管道以自動化 AWS SAM 應用程式的建置和部署，您的 CloudFormation 範本和buildspec.yml檔案必須包含執行下列動作的行：

1. 參考可用映像中具有必要執行時間的建置容器映像。下列範例使用public.ecr.aws/sam/build-nodejs20.x建置容器映像。
2. 設定管道階段以執行必要的 AWS SAM 命令列界面 (CLI) 命令。下列範例執行兩個 AWS SAM CLI 命令：sam build和 sam deploy (具有必要選項)。

此範例假設您已使用 宣告 AWS SAM 範本檔案中的所有函數和層runtime: nodejs20.x。

CloudFormation 範本程式碼片段：

```
CodeBuildProject:
```

```
Type: AWS::CodeBuild::Project
Properties:
  Environment:
    ComputeType: BUILD_GENERAL1_SMALL
    Image: public.ecr.aws/sam/build-nodejs20.x
    Type: LINUX_CONTAINER
  ...
```

buildspec.yml 程式碼片段：

```
version: 0.2
phases:
  build:
    commands:
      - sam build
      - sam deploy --no-confirm-changeset --no-fail-on-empty-changeset
```

如需不同執行時間的可用 Amazon Elastic Container Registry (Amazon ECR) 組建容器映像清單，請參閱 [映像儲存庫 AWS SAM](#)。

使用 Bitbucket 管道搭配 部署 AWS SAM

若要設定 [Bitbucket Pipeline](#) 以自動化 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. 參考可用映像中具有必要執行時間的建置容器映像。下列範例使用 `public.ecr.aws/sam/build-nodejs20.x` 建置容器映像。
2. 設定管道階段以執行必要的 AWS SAM 命令列界面 (CLI) 命令。下列範例執行兩個 AWS SAM CLI 命令：`sam build` 和 `sam deploy` (具有必要選項)。

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
```

如需不同執行時間的可用 Amazon Elastic Container Registry (Amazon ECR) 組建容器映像清單，請參閱 [的影像儲存庫 AWS SAM](#)。

使用 Jenkins 透過 部署 AWS SAM

若要設定 [Jenkins](#) 管道以自動化 AWS SAM 應用程式的建置和部署，您的 Jenkinsfile 必須包含執行下列動作的行：

1. 參考可用映像中具有必要執行時間的建置容器映像。下列範例使用 `public.ecr.aws/sam/build-nodejs20.x` 建置容器映像。
2. 設定管道階段以執行必要的 AWS SAM 命令列界面 (CLI) 命令。下列範例執行兩個 AWS SAM CLI 命令：`sam build` 和 `sam deploy` (具有必要選項)。

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'
      }
    }
  }
}
```

如需不同執行時間的可用 Amazon Elastic Container Registry (Amazon ECR) 組建容器映像清單，請參閱 [的影像儲存庫 AWS SAM](#)。

使用 GitLab CI/CD 搭配 部署 AWS SAM

若要設定 [GitLab](#) 管道以自動化 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. 參考可用映像中具有必要執行時間的建置容器映像。下列範例使用 `public.ecr.aws/sam/build-nodejs20.x` 建置容器映像。

2. 設定管道階段以執行必要的 AWS SAM 命令列界面 (CLI) 命令。下列範例執行兩個 AWS SAM CLI 命令： `sam build` 和 `sam deploy` (具有必要選項)。

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
```

如需不同執行時間的可用 Amazon Elastic Container Registry (Amazon ECR) 組建容器映像清單，請參閱 [的影像儲存庫 AWS SAM](#)。

使用 GitHub 動作搭配 部署 AWS SAM

若要設定 [GitHub](#) 管道以自動化 AWS SAM 應用程式的建置和部署，您必須先在主機上安裝 AWS SAM 命令列界面 (CLI)。您可以在 [GitHub 工作流程中使用 GitHub 動作](#) 來協助進行此設定。GitHub

下列範例 GitHub 工作流程會使用一系列 GitHub 動作設定 Ubuntu 主機，然後執行 AWS SAM CLI 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
```


如需不同執行時間的可用 Amazon Elastic Container Registry (Amazon ECR) 組建容器映像清單，請參閱 [映像儲存庫 AWS SAM](#)。

如何搭配 AWS SAM 管道使用 OIDC 身分驗證

AWS Serverless Application Model (AWS SAM) 支援 Bitbucket、GitHub Actions 和 GitLab 持續整合和持續交付 (CI/CD) 平台的 OpenID Connect (OIDC) 使用者身分驗證。透過此支援，您可以使用任何這些平台的授權 CI/CD 使用者帳戶來管理您的無伺服器應用程式管道。否則，您將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 AWS SAM 管道設定 OIDC

在 `sam pipeline bootstrap` 組態程序期間，執行下列動作以使用 AWS SAM 管道設定 OIDC。

1. 當系統提示您選擇身分提供者時，請選取 OIDC。
2. 接著，選取支援的 OIDC 提供者。
3. 輸入以 `https://` 開頭的 OIDC 提供者 URL。

Note

AWS SAM 會在產生 `AWS::IAM::OIDCProvider` 資源類型時參考此 URL。

4. 接著，依照提示輸入存取所選平台所需的 CI/CD 平台資訊。這些詳細資訊因平台而異，可能包括：
 - OIDC 用戶端 ID。
 - 程式碼儲存庫名稱或通用唯一識別碼 (UUID)。
 - 與儲存庫相關聯的群組或組織名稱。
 - 程式碼儲存庫所屬的 GitHub 組織。
 - GitHub 儲存庫名稱。
 - 部署將從中發生的分支。
5. AWS SAM 會顯示輸入的 OIDC 組態摘要。輸入要編輯之設定的號碼，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，請按 Y 繼續。

AWS SAM 會使用提供的組態產生 `AWS::IAM::OIDCProvider` AWS CloudFormation 資源，以擔任管道執行角色。若要進一步了解此 CloudFormation 資源類型，請參閱 AWS CloudFormation 《使用者指南》中的 [AWS::IAM::OIDCProvider](#)。

 Note

如果身分提供者 (IdP) 資源已存在於您的 中 AWS 帳戶，AWS SAM 會參考它，而不是建立新的資源。

範例

以下是使用 AWS SAM 管道設定 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

如需搭配 AWS SAM 管道使用 OIDC 的詳細資訊，請參閱 [sam pipeline bootstrap](#)。

使用 sam sync 同步至 的簡介 AWS 雲端

Command AWS Serverless Application Model Line Interface (AWS SAM CLI) `sam sync` 命令提供快速將本機應用程式變更同步至 的選項 AWS 雲端。開發應用程式 `sam sync` 時使用 來：

1. 自動偵測本機變更並將其同步至 AWS 雲端。
2. 自訂要同步到 的本機變更 AWS 雲端。
3. 在雲端中準備您的應用程式以進行測試和驗證。

使用 `sam sync`，您可以建立快速開發工作流程，以縮短將本機變更同步至雲端進行測試和驗證所需的時間。

Note

建議開發環境使用 `sam sync` 命令。對於生產環境，我們建議使用或 `sam deploy` 設定持續整合和交付 (CI/CD) 管道。如需詳細資訊，請參閱 [使用 部署您的應用程式和資源 AWS SAM](#)。

`sam sync` 命令是 的一部分 AWS SAM Accelerate。AWS SAM Accelerate 提供工具，可讓您用來加速在 中開發和測試無伺服器應用程式的體驗 AWS 雲端。

主題

- [自動偵測本機變更並將其同步至 AWS 雲端](#)
- [自訂要同步到 的本機變更 AWS 雲端](#)
- [在雲端中準備您的應用程式以進行測試和驗證](#)
- [sam 同步命令的選項](#)
- [疑難排解](#)
- [範例](#)
- [進一步了解](#)

自動偵測本機變更並將其同步至 AWS 雲端

`sam sync` 使用 `--watch` 選項執行，以開始將您的應用程式同步至 AWS 雲端。這會執行下列動作：

1. 建置您的應用程式 – 此程序類似於使用 `sam build` 命令。
2. 部署您的應用程式 – AWS SAM CLI 使用 AWS CloudFormation 使用預設設定將應用程式部署到。使用下列預設值：
 - a. AWS 登入資料和一般組態設定可在您的 `.aws` 使用者資料夾中找到。
 - b. 在您的應用程式 `samconfig.toml` 檔案中找到應用程式部署設定。

如果找不到預設值，AWS SAM CLI 會通知您並結束同步程序。

3. 注意本機變更 – AWS SAM CLI 會保持執行狀態，並注意應用程式的本機變更。這是 `--watch` 選項提供的內容。

此選項預設為開啟。如需預設值，請參閱您應用程式的 `samconfig.toml` 檔案。以下是範例 檔案：

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

4. 同步本機變更至 AWS 雲端- 當您進行本機變更時，AWS 雲端 會透過可用的最快速方法 AWS SAM CLI 偵測並同步這些變更至。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服務 AWS APIs，AWS SAM CLI將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 APIs，AWS SAM CLI將執行 CloudFormation 部署。這會在 中更新您的整個應用程式 AWS 雲端。雖然速度不那麼快，但它確實讓您無法手動啟動部署。

由於 `sam sync` 命令會在 中自動更新您的應用程式 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
```

自訂要同步到 的本機變更 AWS 雲端

提供選項來自訂要同步到 的本機變更 AWS 雲端。這可以加快在雲端中查看本機變更進行測試和驗證所需的時間。

例如，提供僅同步程式碼變更 `--code` 的選項，例如 AWS Lambda 事件函數程式碼。在開發期間，如果您特別專注於 Lambda 程式碼，這將使您快速變更到雲端進行測試和驗證。以下是範例：

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

若要僅同步特定 Lambda 函數或 layer 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
```

在雲端中準備您的應用程式以進行測試和驗證

`sam sync` 命令會自動尋找可在 中更新應用程式的最快方法 AWS 雲端。這可以加速您的開發和雲端測試工作流程。透過利用 AWS 服務 APIs，您可以快速開發、同步和測試支援的資源。如需實作範例，請參閱完成 AWS SAM 研討會中的 [單元 6 - AWS SAM 加速](#)。

sam 同步命令的選項

以下是您可以用來修改 `sam sync` 命令的一些主要選項。如需所有選項的清單，請參閱 [sam sync](#)。

執行一次性 CloudFormation 部署

使用 `--no-watch` 選項來關閉自動同步。以下是範例：

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

AWS SAM CLI 將執行一次性 CloudFormation 部署。此命令會將 `sam build` 和 `sam deploy` 命令執行的動作分組在一起。

略過初始 CloudFormation 部署

您可以自訂每次 `sam sync` 執行時是否需要 CloudFormation 部署。

- 提供 `--no-skip-deploy-sync` 以在每次執行 `sam sync` 時要求 CloudFormation 部署。這可確保您的本機基礎設施同步到 CloudFormation，防止偏離。使用此選項可為您的開發和測試工作流程增加額外的時間。
- 提供 `--skip-deploy-sync` 讓 CloudFormation 部署成為選用。AWS SAM CLI 會將您的本機 AWS SAM 範本與部署的 CloudFormation 範本進行比較，如果未偵測到變更，會略過初始 CloudFormation 部署。略過 CloudFormation 部署可節省同步本機變更至 AWS 雲端的時間。

如果未偵測到變更，AWS SAM CLI 仍會在下列情況下執行 CloudFormation 部署：

- 如果自上次 CloudFormation 部署以來已超過 7 天。
- 如果偵測到大量的 Lambda 函數程式碼變更，讓 CloudFormation 部署成為更新應用程式的最快方法。

以下是範例：

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

從巢狀堆疊同步資源

從巢狀堆疊同步資源

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 在巢狀堆疊中提供資源 `--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 [在 中使用巢狀應用程式重複使用程式碼和資源 AWS SAM](#)。

指定要更新的特定 CloudFormation 堆疊

若要指定要更新的特定 CloudFormation 堆疊，請提供 `--stack-name` 選項。以下是範例：

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

在來源資料夾中建置您的專案，以加快建置時間

對於支援的執行時間和建置方法，您可以使用 `--build-in-source` 選項直接在來源資料夾中建置專案。根據預設，會在暫存目錄中 AWS SAM CLI 建置，這涉及透過原始程式碼和專案檔案進行複製。使用 `--build-in-source` 時，AWS SAM 會直接在您的來源資料夾中 CLI 建置，而不需要將檔案複製到暫時目錄，即可加速建置程序。

如需支援的執行時間和建置方法清單，請參閱 [--build-in-source](#)。

指定不會啟動同步的檔案和資料夾

使用 `--watch-exclude` 選項指定更新時不會啟動同步的任何檔案或資料夾。如需有關此選項的詳細資訊，請參閱 [--watch-exclude](#)。

以下是排除與 `HelloWorldFunction` 函數相關聯 `package-lock.json` 檔案的範例：

```
$ sam sync --watch --watch-exclude HelloWorldFunction=package-lock.json
```

執行此命令時，AWS SAM CLI 會啟動同步程序。這包含下列項目：

- 執行 `sam build` 來建置您的 函數，並準備您的應用程式以進行部署。
- 執行 `sam deploy` 以部署您的應用程式。
- 請注意應用程式的變更。

當我們修改 `package-lock.json` 檔案時，AWS SAM CLI 不會啟動同步。更新另一個檔案時，AWS SAM CLI 會啟動同步，其中包含 `package-lock.json` 檔案。

以下是指定子堆疊 Lambda 函數的範例：

```
$ sam sync --watch --watch-exclude ChildStackA/MyFunction=database.sqlite3
```

疑難排解

若要對 進行故障診斷 AWS SAM CLI，請參閱 [AWS SAM CLI 故障診斷](#)。

範例

使用 sam 同步更新 Hello World 應用程式

在此範例中，我們從初始化範例 Hello World 應用程式開始。若要進一步了解此應用程式，請參閱 [教學課程：使用 部署 Hello World 應用程式 AWS SAM](#)。

執行 `aws 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-samclisam-s3-demo-
bucket-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-
                   BUFVM02PJIYF

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HelloWorldFunction程式碼。AWS SAM CLI 會偵測此變更，並將我們的應用程式同步至 AWS 雲端。由於 AWS Lambda 支援 AWS 服務 APIs，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.

```

接著，我們在應用程式的 AWS SAM 範本中修改 API 端點。我們將 /hello 變更為 /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 AWS SAM CLI 會自動執行 CloudFormation 部署。以下是輸出範例：

```

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-samclisam-s3-demo-
bucket-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-BUFVM02PJYF
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選項的說明，請參閱 [sam sync](#)。

使用 監控無伺服器應用程式 AWS SAM

部署無伺服器應用程式之後，您可以對其進行監控，以提供有關其操作的洞察並偵測異常，這有助於進行故障診斷。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 CloudWatch 在偵測到異常時通知您的資訊。它還提供使用日誌的資訊，包括錯誤反白和檢視、篩選、擷取和結尾日誌的提示。

主題

- [使用 CloudWatch Application Insights 監控無 AWS SAM 伺服器應用程式](#)
- [在 中使用 日誌 AWS SAM](#)

使用 CloudWatch Application Insights 監控無 AWS SAM 伺服器應用程式

Amazon CloudWatch Application Insights 可協助您監控應用程式中 AWS 的資源，以協助識別潛在問題。它可以分析 AWS 資源資料是否有問題的跡象，並建置自動化儀表板以視覺化它們。您可以設定 CloudWatch Application Insights 搭配您的 AWS Serverless Application Model (AWS SAM) 應用程式使用。若要進一步了解 CloudWatch Application Insights，請參閱《[Amazon CloudWatch 使用者指南](#)》中的 [Amazon CloudWatch Application Insights](#)。Amazon CloudWatch

主題

- [使用 設定 CloudWatch Application Insights AWS SAM](#)
- [後續步驟](#)

使用 設定 CloudWatch Application Insights AWS SAM

透過 AWS SAM 命令列界面 (AWS SAM CLI) 或透過 AWS SAM 範本為您的 AWS SAM 應用程式設定 CloudWatch Application Insights。

透過 設定 AWS SAM CLI

使用 初始化應用程式時 `sam init`，請透過互動式流程或使用 `--application-insights` 選項來啟用 CloudWatch Application Insights。

若要透過 AWS SAM CLI 互動式流程啟用 CloudWatch Application Insights，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` 選項啟用 CloudWatch Application Insights，請執行下列動作。

```
sam init --application-insights
```

若要進一步了解如何使用 `sam init` 命令，請參閱 [sam init](#)。

透過 AWS SAM 範本設定

在 AWS SAM 範本 `AWS::ApplicationInsights::Application` 中定義 `AWS::ResourceGroups::Group` 和資源，以啟用 CloudWatch Application Insights。

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  ApplicationResourceGroup:
    Type: AWS::ResourceGroups::Group
    Properties:
      Name:
        Fn::Join:
          - ''
          - - ApplicationInsights-SAM-
            - Ref: AWS::StackName
      ResourceQuery:
        Type: CLOUDFORMATION_STACK_1_0
  ApplicationInsightsMonitoring:
    Type: AWS::ApplicationInsights::Application
    Properties:
      ResourceGroupName:
        Fn::Join:
          - ''
          - - ApplicationInsights-SAM-
            - Ref: AWS::StackName
      AutoConfigurationEnabled: 'true'
    DependsOn: ApplicationResourceGroup
```

- `AWS::ResourceGroups::Group` – 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任務。在這裡，您可以建立要與 CloudWatch Application Insights 搭配

使用的資源群組。如需此資源類型的詳細資訊，請參閱AWS CloudFormation 《使用者指南[AWS::ResourceGroups::Group](#)》中的。

- [AWS::ApplicationInsights::Application](#) – 設定資源群組的 CloudWatch Application Insights。如需此資源類型的詳細資訊，請參閱AWS CloudFormation 《使用者指南[AWS::ApplicationInsights::Application](#)》中的。

這兩個資源都會在應用程式部署 CloudFormation 時自動傳遞至。您可以使用 AWS SAM 範本中的 CloudFormation 語法進一步設定 CloudWatch Application Insights。如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的[使用 CloudFormation 範本](#)。

使用 `sam init --application-insights` 命令時，這兩個資源都會在您的 AWS SAM 範本中自動產生。以下是產生範本的範例。

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: >
  sam-app-test

  Sample SAM Template for sam-app-test

# More info about Globals: https://github.com/awslabs/serverless-application-model/blob/master/docs/globals.rst
Globals:
  Function:
    Timeout: 3
    MemorySize: 128

Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function # More info about Function Resource:
    https://github.com/awslabs/serverless-application-model/blob/master/versions/2016-10-31.md#awsserverlessfunction
    Properties:
      CodeUri: hello_world/
      Handler: app.lambda_handler
      Runtime: python3.9
      Architectures:
        - x86_64
      Events:
        HelloWorld:
```

```
Type: Api # More info about API Event Source: https://github.com/aws-labs/
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/aws-labs/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 Application Insights 之後，請使用 `sam build` 建置您的應用程式 `sam deploy` 和部署您的應用程式。所有 CloudWatch Application Insights 支援的資源都會設定為監控。

- 如需支援的資源清單，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 Application Insights，請參閱《Amazon [CloudWatch 使用者指南](#)》中的 [存取 CloudWatch Application Insights](#)。Amazon CloudWatch

在 中使用 日誌 AWS SAM

為了簡化故障診斷，AWS SAM CLI 有一個名為 `sam logs` 的命令。此命令可讓您從命令列擷取 Lambda 事件函數產生的日誌。

Note

`sam logs` 命令適用於所有 AWS Lambda 函數，而不只是您部署使用的函數 AWS SAM。

依 CloudFormation 堆疊擷取日誌

當您的函數是 CloudFormation 堆疊的一部分時，您可以使用函數的邏輯 ID 來擷取日誌：

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

依 Lambda 事件函數名稱擷取日誌

或者，您可以使用函數的名稱來擷取日誌：

```
sam logs -n mystack-HelloWorldFunction-1FJ8PD
```

自訂日誌

新增 `--tail` 選項以等待新的日誌，並在它們到達時看到它們。這在部署期間或當您疑難排解生產問題時很有用。

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

檢視特定時間範圍的日誌

您可以使用 `-s` 和 `-e` 選項來檢視特定時間範圍的日誌：

```
sam logs -n HelloWorldFunction --stack-name mystack -s '10min ago' -e '2min ago'
```

篩選日誌

使用 `--filter` 選項快速尋找符合日誌事件中詞彙、片語或值的日誌：

```
sam logs -n HelloWorldFunction --stack-name mystack --filter "error"
```

在輸出中，AWS SAM CLI 會強調「錯誤」一詞的所有出現，讓您可以輕鬆地在日誌輸出中尋找篩選條件關鍵字。

反白顯示錯誤

當您的 Lambda 函數當機或逾時時，AWS SAM CLI 會以紅色反白顯示逾時訊息。這可協助您輕鬆尋找日誌輸出巨型串流內逾時的特定執行。

JSON 美型列印

如果您的日誌訊息列印 JSON 字串，AWS SAM CLI 會自動列印 JSON，以協助您以視覺化方式剖析和了解 JSON。

AWS SAM 參考

本節包含 AWS SAM 參考資料。這包括 AWS SAM CLI 參考資料，例如命令的 AWS SAM CLI 參考資訊，以及其他資訊 AWS SAM CLI，例如組態、版本控制和故障診斷資訊。此外，本節包含 AWS SAM 規格和 AWS SAM 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 AWS SAM 範本

此 AWS SAM 規格是 Apache 2.0 授權下的開放原始碼規格。AWS SAM 規格的目前版本在 [AWS SAM 範本](#)。AWS SAM specification 中提供簡化的速記語法，您可用來定義無伺服器應用程式的函數、事件、APIs、組態和許可。

您可以透過 AWS SAM 應用程式專案目錄與 AWS SAM 規格互動，這些目錄是執行 `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。AWS SAM 範本是 AWS CloudFormation 範本的延伸。如需範本的完整參考 CloudFormation，請參閱 AWS CloudFormation 《使用者指南》中的 [範本參考](#)。

AWS SAM CLI 命令參考

AWS Serverless Application Model 命令列界面 (AWS SAM CLI) 是一種命令列工具，您可以搭配 AWS SAM 範本和支援的第三方整合來建置和執行無伺服器應用程式。

您可以使用 AWS SAM CLI 命令來開發、測試無伺服器應用程式並將其部署到 AWS 雲端。以下是一些命令的範例 AWS SAM CLI：

- `sam init` – 如果您是第一次 AWS SAM CLI 使用，您可以執行 `sam init` 命令，無需任何參數即可建立 Hello World 應用程式。命令會以您選擇的語言產生預先設定的 AWS SAM 範本和範例應用程式程式碼。
- `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，然後再將其部署到 AWS 雲端。
- `sam logs` – 使用此命令來擷取 Lambda 事件函數產生的日誌。這可協助您在將應用程式部署到之後進行測試和偵錯 AWS 雲端。
- `sam package` – 使用此命令將您的應用程式程式碼和相依性綁定到部署套件中。您需要部署套件，才能將應用程式上傳至 AWS 雲端。
- `sam deploy` – 使用此命令將您的無伺服器應用程式部署到 AWS 雲端。它會建立 AWS 資源，並設定 AWS SAM 範本中定義的許可和其他組態。

如需安裝 的指示 AWS SAM CLI，請參閱 [安裝 AWS SAM CLI](#)。

AWS SAM 政策範本

使用 時 AWS SAM，您可以從政策範本清單中選擇，將函數的許可範圍限定 AWS Lambda 為應用程式使用的資源。如需可用政策範本的清單，請參閱 [政策範本資料表](#)。如需政策範本和 的一般資訊 AWS SAM，請參閱 [AWS SAM 政策範本](#)。

主題

- [AWS SAM 範本](#)
- [AWS SAM CLI 命令參考](#)
- [AWS SAM CLI 組態檔案](#)
- [AWS SAM 連接器參考](#)
- [AWS SAM 政策範本](#)
- [的影像儲存庫 AWS SAM](#)
- [中的遙測 AWS SAM CLI](#)
- [在 AWS SAM 範本中設定和管理資源存取](#)

AWS SAM CLI 命令參考

本節包含 命令的 AWS SAM CLI參考資訊。這包括用量的詳細資訊、每個命令可用的不同選項的完整清單，以及其他資訊。適用時，其他資訊會包含引數、環境變數和事件等詳細資訊。如需詳細資訊，請參閱每個命令。如需安裝 的指示 AWS SAM CLI，請參閱 [安裝 AWS SAM CLI](#)。

主題

- [sam build](#)
- [sam delete](#)
- [sam deploy](#)
- [sam init](#)
- [sam list](#)
- [sam local callback](#)

- [sam local execution](#)
- [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 callback](#)
- [sam remote execution](#)
- [sam remote invoke](#)
- [sam remote test-event](#)
- [sam sync](#)
- [sam traces](#)
- [sam validate](#)

sam build

此頁面提供 AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam build` 命令的參考資訊。

- 如需 的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)
- 如需使用 AWS SAM CLI `sam build` 命令的文件，請參閱 [使用 建置 簡介 AWS SAM](#)。

`sam build` 命令會準備應用程式以進行開發人員工作流程中的後續步驟，例如本機測試或部署到 AWS 雲端。

Usage

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


引數

Resource ID (資源 ID)

選用。指示 AWS SAM 在 [AWS SAM 範本](#) 中建立宣告的單一資源。指定資源的建置成品將是唯一可用於工作流程中後續命令的建置成品，即 `sam package` 和 `sam deploy`。

選項

`--base-dir, -s DIRECTORY`

解決函數或 layer 的原始程式碼相對於此目錄的相對路徑。如果您想要變更原始程式碼資料夾的相對路徑解析方式，請使用此選項。根據預設，相對路徑會根據範本的位置進行解析 AWS SAM。

除了您要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 Beta 版功能。

`--build-dir, -b DIRECTORY`

存放建置成品的目錄路徑。此選項會移除此目錄及其所有內容。

`--build-image TEXT`

您要為組建提取的容器映像的 URI。根據預設，會從 Amazon ECR Public AWS SAM 提取容器映像。使用此選項從另一個位置提取映像。

您可以多次指定此選項。此選項的每個執行個體都可以採用字串或鍵值對。如果您指定字串，它是要用於應用程式中所有資源的容器映像的 URI。例如 `sam build --use-container --build-image amazon/aws-sam-cli-build-image-python3.8`。如果您指定金鑰/值對，則金鑰是資源名稱，而值是用於該資源的容器映像的 URI。例如 `sam build --use-container`

`--build-image Function1=amazon/aws-sam-cli-build-image-python3.8`。使用鍵/值對，您可以為不同的資源指定不同的容器映像。

此選項僅適用於指定 `--use-container` 選項時，否則將導致錯誤。

`--build-in-source` | `--no-build-in-source`

提供 `--build-in-source` 以直接在來源資料夾中建置您的專案。

`--build-in-source` 選項支援下列執行時間和建置方法：

- 執行時間 – [sam init --runtime](#) 選項支援的任何 Node.js 執行時間。
- 組建方法 – Makefile、esbuild。

`--build-in-source` 選項與下列選項不相容：

- `--hook-name`
- `--use-container`

預設：`--no-build-in-source`

`--cached` | `--no-cached`

啟用或停用快取組建。使用此選項可重複使用未從先前 builds 變更的建置成品。會 AWS SAM 評估您是否變更專案目錄中的任何檔案。根據預設，組建不會快取。如果叫用 `--no-cached` 選項，則會覆寫 `samconfig.toml` 中的 `cached = true` 設定。

 Note

AWS SAM 不會評估您是否變更了專案所依賴的第三方模組，而您尚未提供特定版本。例如，如果您的 Python 函數包含項目 `requirements.txt` 的檔案 `requests=1.x`，而最新的請求模組版本從 1.1 變更為 1.2，則 AWS SAM 在您執行非快取建置之前，不會提取最新版本。

`--cache-dir`

`--cached` 指定 時存放快取成品的目錄。預設快取目錄為 `.aws-sam/cache`。

`--config-env` *TEXT*

在要使用的組態檔案中指定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 SAM CLI 組態檔案](#)。

`--config-file` *PATH*

組態檔案的路徑和檔案名稱，其中包含要使用的預設參數值。專案目錄根目錄中的預設值為「samconfig.toml」。如需關於組態檔案的詳細資訊，請參閱 [AWS SAM CLI 組態檔案](#)。

`--container-env-var`, `-e` *TEXT*

要傳遞至建置容器的環境變數。您可以多次指定此選項。此選項的每個執行個體都會接受金鑰值對，其中金鑰是資源和環境變數，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`。

此選項僅適用於指定 `--use-container` 選項時，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 [容器環境變數檔案](#)。

此選項僅適用於指定 `--use-container` 選項時，否則將導致錯誤。

`--debug`

開啟偵錯記錄以列印產生的偵錯訊息 AWS SAM CLI，並顯示時間戳記。

`--docker-network` *TEXT*

指定 Lambda Docker 容器應連線之現有 Docker 網路的名稱或 ID，以及預設橋接網路。如果未指定，Lambda 容器只會連線到預設橋接 Docker 網路。

`--exclude`, `-x`

要從 () 排除的資源名稱 sam build。例如，如果您的範本包含 Function1、和 Function2，Function3 而且您執行 `sam build --exclude Function2, Function3` 則只會建置 Function1 和。

`--help`

顯示此訊息並結束。

`--hook-name` *TEXT*

用於擴展 AWS SAM CLI 功能的勾點名稱。

接受的值：terraform。

`--manifest` , `-m` *PATH*

要使用的自訂相依性資訊清單檔案路徑 (例如 `package.json`) , 而非預設值。

`--mount-symlinks`

確保 AWS SAM CLI 一律掛載檔案中存在的符號連結來建置或叫用。這僅適用於頂層目錄上的符號連結 (即直接在函數根目錄上的符號連結) 。根據預設, 不會掛載符號連結, 除了在 NodeJS `node_modules` 中使用 `build-in-source` 時需要的符號連結。

`--no-use-container`

可讓您使用 IDE 工具組來設定預設行為的選項。您也可以使用 在本機機器中 `sam build --no-use-container` 執行組建, 而不是使用 Docker 容器。

`--output` [*text* | *json*]

命令輸出的格式:

- `text` – 列印一般、人類可讀取的輸出。這是預設值。
- `json` – 列印結構化、機器可讀取的輸出。當您想要以程式設計方式使用輸出時, 例如在 CI/CD 管道、IDE 延伸模組和 AI 輔助工具中, 請使用此格式。

當您指定時 `--output json`, 會在 AWS SAM CLI 命令完成後列印單一 JSON 物件。成功時, 此物件會包含組建狀態、組建目錄、範本檔案和所建置資源清單等欄位。失敗時, 它包含一個 `error` 物件, 描述出了什麼問題。

`--parallel`

啟用平行建置。使用此選項來平行建置 AWS SAM 範本的函數和層。根據預設, 函數和層會依序建置。

`--parameter-overrides`

(選用) 包含 CloudFormation 參數覆寫編碼為鍵值對的字串。使用與 AWS Command Line Interface () 相同的格式 AWS CLI。例如: `'ParameterKey=KeyPairName, ParameterValue=MyKey ParameterKey=InstanceType, ParameterValue=t1.micro'`。此選項與 `--hook-name` 不相容。

`--profile` *TEXT*

從您的登入資料檔案中取得 AWS 登入資料的特定設定檔。

`--region` *TEXT*

AWS 區域 要部署到的。例如 `us-east-1`。

--save-params

將您在命令列提供的參數儲存至 AWS SAM 組態檔案。

--skip-prepare-infra

如果未進行任何基礎設施變更，請略過準備階段。使用 搭配 --hook-name 選項。

--skip-pull-image

指定命令是否應該略過提取 Lambda 執行時間的最新 Docker 映像。

--template-file, --template, -t *PATH*

AWS SAM 範本檔案 的路徑和檔案名稱[default: template.[yaml|yml]]。此選項與 不相容--hook-name。

--terraform-project-root-path

最上層目錄的相對或絕對路徑，其中包含您的Terraform組態檔案或函數原始程式碼。如果這些檔案位於包含Terraform根模組的目錄之外，請使用此選項指定其絕對或相對路徑。此選項需要--hook-name將 設定為 terraform。

--use-buildkit | --no-use-buildkit

使用容器映像建置的BuildKit外掛程式啟用或停用 。這將使用 Finch或 Docker CLIs。根據預設，建置會使用 SDK 中的舊版建置器。如果叫用 --no-use-buildkit選項，則會覆寫 samconfig.toml 中的use-buildkit = true設定。

--use-container, -u

如果您的函數依賴於具有原生編譯相依性的套件，請使用此選項在類似 Lambda 的 Docker 容器中建置函數。

範例

如需使用 sam build 子命令的詳細範例和深入演練，請參閱 [使用 建置 簡介 AWS SAM](#)。

sam delete

此頁面提供 AWS Serverless Application Model Command Line Interface (AWS SAM CLI) sam delete命令的參考資訊。

如需的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)

`sam delete` 命令會刪除 CloudFormation 堆疊、封裝並部署至 Amazon S3 和 Amazon ECR 的成品，以及 AWS SAM 範本檔案，藉此刪除 AWS SAM 應用程式。

此命令也會檢查是否已部署 Amazon ECR 配套堆疊，如果是，則提示使用者刪除該堆疊和 Amazon ECR 儲存庫。如果指定 `--no-prompts`，則依預設會刪除配套堆疊和 Amazon ECR 儲存庫。

Usage

```
$ sam delete <options>
```

選項

`--config-env` *TEXT*

在要使用的組態檔案中指定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 SAM CLI 組態檔案](#)。

`--config-file` *PATH*

組態檔案的路徑和檔案名稱，其中包含要使用的預設參數值。預設值位於專案目錄的根 `samconfig.toml` 目錄中。如需關於組態檔案的詳細資訊，請參閱 [AWS SAM CLI 組態檔案](#)。

`--debug`

開啟偵錯記錄以列印產生的偵錯訊息 AWS SAM CLI，並顯示時間戳記。

`--help`

顯示此訊息並結束。

`--no-prompts`

指定此選項，讓以非互動式模式 AWS SAM 運作。必須使用 `--stack-name` 選項或在組態 `toml` 檔案中提供堆疊名稱。

`--profile` *TEXT*

從您的登入資料檔案中取得 AWS 登入資料的特定設定檔。

`--region` *TEXT*

AWS 要部署的區域。例如 `us-east-1`。

`--s3-bucket`

您要刪除的 Amazon S3 儲存貯體路徑。

`--s3-prefix`

您要刪除的 Amazon S3 儲存貯體字首。

`--save-params`

將您在命令列提供的參數儲存至 AWS SAM 組態檔案。

`--stack-name` *TEXT*

您要刪除的 CloudFormation 堆疊名稱。

範例

下列命令會刪除堆疊 MY-STACK。

```
$ sam delete --stack-name MY-STACK
```

下列命令會刪除堆疊MY-STACK和 S3 儲存貯體 sam-s3-demo-bucket：

```
$ sam delete \  
  --stack-name MyStack \  
  --s3-bucket MySAMBucket
```

sam deploy

此頁面提供 AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam deploy`命令的參考資訊。

- 如需 的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)
- 如需使用 AWS SAM CLIsam deploy命令的文件，請參閱 [使用 部署簡介 AWS SAM](#)。

`sam deploy` 命令 AWS 雲端 會使用 將應用程式部署到 AWS CloudFormation。

Usage

```
$ <environment variables> sam deploy <options>
```

環境變數

SAM_CLI_POLL_DELAY

設定 Shell 中值為 秒SAM_CLI_POLL_DELAY的環境變數，以設定 AWS SAM CLI 檢查 CloudFormation 堆疊狀態的頻率，這在查看調節時非常有用 CloudFormation。此 env 變數用於輪詢在執行時進行的 describe_stack API 呼叫sam deploy。

以下是此變數的範例：

```
$ SAM_CLI_POLL_DELAY=5 sam deploy
```

選項

--capabilities *LIST*

您必須指定才能允許 CloudFormation 建立特定堆疊的功能清單。某些堆疊範本可能包含會影響許可的資源，AWS 帳戶例如，透過建立新的 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可其功能。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。如果您有具有自訂名稱的 IAM 資源，則必須指定 CAPABILITY_NAMED_IAM。如果您未指定此選項，則操作會傳回InsufficientCapabilities錯誤。

當您部署包含巢狀應用程式的應用程式時，您必須使用 CAPABILITY_AUTO_EXPAND來確認應用程式包含巢狀應用程式。如需詳細資訊，請參閱[部署巢狀應用程式](#)。

--config-env *TEXT*

在要使用的組態檔案中指定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 SAM CLI 組態檔案](#)。

--config-file *PATH*

組態檔案的路徑和檔案名稱，其中包含要使用的預設參數值。預設值位於專案目錄的根samconfig.toml目錄中。如需關於組態檔案的詳細資訊，請參閱[AWS SAM CLI 組態檔案](#)。

--confirm-changeset | --no-confirm-changeset

提示 AWS SAM CLI確認 是否部署計算的變更集。

--debug

開啟偵錯記錄以列印 產生的偵錯訊息 AWS SAM CLI，並顯示時間戳記。

`--disable-rollback | --no-disable-rollback`

指定在部署期間發生錯誤時是否要轉返 CloudFormation 堆疊。根據預設，如果部署期間發生錯誤，您的 CloudFormation 堆疊會回復到最後的穩定狀態。如果您指定 `--disable-rollback` 並在部署期間發生錯誤，則在發生錯誤之前建立或更新的資源不會復原。

`--express | --no-express`

指定是否使用 CloudFormation 快速模式。作用中時，一旦套用資源組態，堆疊操作就會完成。資源會繼續在背景準備就緒。部署時間最多可縮短 4 倍。

當您也指定 AWS SAM CLI 時，CloudFormation 會將轉返行為傳遞至 DeploymentConfig 參數 `--disable-rollback`。

預設值為 `--no-express`。

`--fail-on-empty-changeset | --no-fail-on-empty-changeset`

指定如果堆疊沒有變更，是否傳回非零結束碼。預設行為是傳回非零結束代碼。

`--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。比對成品會遭到覆寫。

`--guided, -g`

指定此選項，讓 AWS SAM CLI 使用提示來引導您完成部署。

`--help`

顯示此訊息並結束。

`--image-repositories TEXT`

函數映射至其 Amazon ECR 儲存庫 URI。依邏輯 ID 參考函數。以下是範例：

```
$ sam deploy --image-repositories Function1=123456789012.dkr.ecr.us-east-1.amazonaws.com/my-repo
```

您可以在單一命令中多次指定此選項。

`--image-repository TEXT`

此命令上傳函數映像的 Amazon ECR 儲存庫名稱。使用 Image 套件類型宣告的函數需要此選項。

`--kms-key-id` *TEXT*

用來加密 Amazon S3 儲存貯體中靜態成品的 AWS Key Management Service (AWS KMS) 金鑰 ID。如果您未指定此選項，則 AWS SAM 會使用 Amazon S3-managed 加密金鑰。

`--metadata`

要連接到範本中參考之所有成品的中繼資料映射。

`--no-execute-changeset`

指出是否套用變更集。如果您想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 CloudFormation 建立變更集，然後結束而不套用變更集。若要套用變更集，請在沒有此選項的情況下執行相同的命令。

`--no-progressbar`

將成品上傳至 Amazon S3 時，請勿顯示進度列。

`--notification-arns` *LIST*

與堆疊建立 CloudFormation 關聯的 Amazon Simple Notification Service (Amazon SNS) 主題 ARNs 清單。

`--on-failure` [ROLLBACK | DELETE | DO_NOTHING]

指定堆疊無法建立時要採取的動作。

下列選項可供使用：

- ROLLBACK – 將堆疊復原至先前的已知良好狀態。
- DELETE – 如果存在堆疊，則將堆疊轉返至先前的已知良好狀態。否則，會刪除堆疊。
- DO_NOTHING – 既不會轉返，也不會刪除堆疊。效果與 `--disable-rollback` 的效果相同。

預設行為是 ROLLBACK。

 Note

您可以指定 `--disable-rollback` 選項或 `--on-failure` 選項，但不能同時指定兩者。

`--output` [*text* | *json*]

命令輸出的格式：

- `text` – 列印一般、人類可讀取的輸出。這是預設值。
- `json` – 列印結構化、機器可讀取的輸出。當您想要以程式設計方式使用輸出時，例如在 CI/CD 管道、IDE 延伸模組和 AI 輔助工具中，請使用此格式。

部署會在一段時間內執行。當您指定 `--output json`，AWS SAM CLI 會隨著部署進度，列印每行一個 JSON 物件的串流。每個物件都包含一個 `type` 欄位，可識別其包含的資訊類型，例如變更集、部署事件、堆疊輸出和最終結果。在失敗時，最後一個物件包含一個 `error` 物件，描述出了什麼問題。

 Note

`--output json` 選項與需要互動式提示的選項不相容，例如 `--guided` 和 `--confirm-changeset`。

`--parameter-overrides` *LIST*

包含 CloudFormation 參數的字串會覆寫編碼為鍵值對的字串。每個覆寫都會使用格式 `ParameterKey=name,ParameterValue=value`。多個覆寫會以空格分隔。以下是兩個範例：

```
$ sam deploy --parameter-overrides ParameterKey=value1,ParameterValue=value2
```

```
$ sam deploy --parameter-overrides ParameterKey=value1,ParameterValue=value2  
ParameterKey=hello,ParameterValue=world ParameterKey=apple,ParameterValue=banana
```

`--profile` *TEXT*

從您的登入資料檔案中取得 AWS 登入資料的特定設定檔。

`--region` *TEXT*

AWS 區域 要部署到的。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。此選項僅適用於具有 `PackageType: Image` 指定的函數和層。如果您指定 `--guided` 選項，則 AWS SAM CLI 會忽略 `--resolve-image-repos`。

 Note

如果 使用此選項 AWS SAM 自動為函數或 layer 建立任何 Amazon ECR 儲存庫，而您稍後將這些函數或 layer 從 AWS SAM 範本中刪除，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，用於封裝和部署非引導式部署。如果您指定 --guided 選項，則 AWS SAM CLI 會忽略 --resolve-s3。如果您同時指定 --s3-bucket 和 --resolve-s3 選項，則會發生錯誤。

--role-arn *TEXT*

套用變更集時 CloudFormation 擔任之 IAM 角色的 Amazon Resource Name (ARN)。

--s3-bucket *TEXT*

此命令上傳 CloudFormation 範本的 Amazon S3 儲存貯體名稱。如果您的範本大於 51,200 個位元組，則需要 --s3-bucket 選項或 --resolve-s3 選項。如果您同時指定 --s3-bucket 和 --resolve-s3 選項，則會發生錯誤。

--s3-prefix *TEXT*

新增至上傳到 Amazon S3 儲存貯體之成品名稱的字首。字首名稱是 Amazon S3 儲存貯體的路徑名稱（資料夾名稱）。

--save-params

將您在命令列提供的參數儲存至 AWS SAM 組態檔案。

--signing-profiles *LIST*

用來簽署部署套件的簽署設定檔清單。此選項會取得索引鍵/值對的清單，其中索引鍵是要簽署的函數或 layer 的名稱，而值是簽署描述檔，選用的描述檔擁有者以 分隔:。例如 FunctionNameToSign=SigningProfileName1
LayerNameToSign=SigningProfileName2:SigningProfileOwner。

--stack-name *TEXT*

（必要）您要部署的 CloudFormation 堆疊名稱。如果您指定現有的堆疊，則命令會更新堆疊。如果您指定新的堆疊，則命令會建立它。

--tags *LIST*

要與建立或更新的堆疊建立關聯的標籤清單。CloudFormation 也會將這些標籤傳播到堆疊中支援該標籤的資源。

--template-file, --template, -t *PATH*

範本 AWS SAM 所在的路徑和檔案名稱。

 Note

如果您指定此選項，則 AWS SAM 只會部署範本及其指向的本機資源。

--use-json

CloudFormation 範本的輸出 JSON。預設輸出為 YAML。

範例

如需使用 `sam deploy` 子命令的詳細範例和深入演練，請參閱 [使用 部署簡介 AWS SAM](#)。

sam init

此頁面提供 AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam init` 命令的參考資訊。

- 如需 的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)
- 如需使用 AWS SAM CLI `sam init` 命令的文件，請參閱 [在 中建立您的應用程式 AWS SAM](#)。

`sam init` 命令提供選項來初始化新的無伺服器應用程式。

Usage

```
$ sam init <options>
```

選項

`--app-template` *TEXT*

您要使用的受管應用程式範本的識別符。如果您不確定，請呼叫 `sam init` 沒有互動式工作流程的選項。

如果 `--no-interactive` 指定 `--location` 且未提供 `--app-template`，則需要此參數。

此參數僅適用於 0. AWS SAM CLI 3.0 版和更新版本。使用較早版本指定此參數會導致錯誤。

`--application-insights` | `--no-application-insights`

為您的應用程式啟用 Amazon CloudWatch Application Insights 監控。如需詳細資訊，請參閱 [使用 CloudWatch Application Insights 監控無 AWS SAM 伺服器應用程式](#)。

預設選項為 `--no-application-insights`。

`--architecture`, `-a` [*x86_64* | *arm64*]

應用程式 Lambda 事件函數的指示集架構。指定 *x86_64* 或 *arm64*。

`--base-image` [*amazon/dotnet8-base* | *amazon/dotnet6-base* | *amazon/java25-base* | *amazon/java21-base* | *amazon/java17-base* | *amazon/java17.al2023-base* | *amazon/java11-base* | *amazon/java11.al2023-base* | *amazon/java8.al2023-base* | *amazon/nodejs24.x-base* | *amazon/nodejs22.x-base* | *amazon/nodejs20.x-base* | *amazon/nodejs18.x-base* | *amazon/nodejs16.x-base* | *amazon/python3.14-base* | *amazon/python3.13-base* | *amazon/python3.12-base* | *amazon/python3.11-base* | *amazon/python3.10-base* | *amazon/python3.9-base* | *amazon/python3.8-base* | *amazon/ruby4.0-base* | *amazon/ruby3.4-base* | *amazon/ruby3.3-base* | *amazon/ruby3.2-base*]

您應用程式的基礎映像。此選項僅適用於套件類型為 `Image`。

如果指定 `--no-interactive`、`--package-type` 指定為 `Image` 且未指定 `--location` 指定 `Image`，則此參數為必要。

`--config-env` *TEXT*

在要使用的組態檔案中指定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 SAM CLI 組態檔案](#)。

`--config-file` *PATH*

組態檔案的路徑和檔案名稱，其中包含要使用的預設參數值。預設值為專案目錄根目錄中的 "samconfig.toml"。如需關於組態檔案的詳細資訊，請參閱 [AWS SAM CLI 組態檔案](#)。

`--debug`

開啟偵錯記錄以列印 產生的偵錯訊息 AWS SAM CLI，並顯示時間戳記。

`--dependency-manager, -d` [*gradle* | *mod* | *maven* | *bundler* | *npm* | *cli-package* | *pip*]

Lambda 執行時間的相依性管理員。

`--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*

範本或應用程式位置 (Git、Mercurial、HTTP/HTTPS、.zip 檔案、路徑)。

如果指定 `--no-interactive` 且 `--app-template` 未提供 `--runtime`、和 `--name`，則需要此參數。

對於 Git 儲存庫，您必須使用儲存庫根的位置。

對於本機路徑，範本必須是 .zip 檔案或 [Cookiecutter](#) 格式。

`--name, -n` *TEXT*

要產生為目錄的專案名稱。

如果 `--no-interactive` 指定 `--location` 且未提供，則需要此參數。

`--no-input`

停用 Cookiecutter 提示，並接受範本組態中定義的 `vcf` 預設值。

`--no-interactive`

停用啟動參數的互動式提示，如果缺少任何必要值，則失敗。

`--output [ text | json ]`

命令輸出的格式：

- `text` – 列印一般、人類可讀取的輸出。這是預設值。
- `json` – 列印結構化、機器可讀取的輸出。當您想要以程式設計方式使用輸出時，例如在 CI/CD 管道、IDE 延伸模組和 AI 輔助工具中，請使用此格式。

當您指定 `--output json`，會在 AWS SAM CLI 命令完成後列印單一 JSON 物件。成功時，此物件會包含專案目錄和範本檔案等欄位。您可以將輸出鏈結至後續命令，例如 `sam build`。失敗時，它包含一個 `error` 物件，描述出了什麼問題。

 Note

`--output json` 選項與互動式工作流程不相容。若要使用它，請提供所有必要的選項來指定完全非互動式的執行。

`--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。

`--runtime, -r [dotnet8 | dotnet6 | java25 | java21 | java17 | java17.al2023 | java11 | java11.al2023 | java8.al2023 | nodejs24.x | nodejs22.x | nodejs20.x | nodejs18.x | nodejs16.x | python3.14 | python3.13 | python3.12 | python3.11 | python3.10 | python3.9 | python3.8 | ruby4.0 | ruby3.4 | ruby3.3 | ruby3.2]`

應用程式的 Lambda 執行時間。此選項僅適用於套件類型為 `Zip`。

如果指定 `--no-interactive`、`--package-type` 指定為 `Image` 且未 `--location` 指定 `Zip`，則此參數為必要。

`--save-params`

將您在命令列提供的參數儲存至 AWS SAM 組態檔案。

`--tracing | --no-tracing`

為您的 Lambda 函數啟用 AWS X-Ray 追蹤。

範例

如需使用 `sam init` 子命令的詳細範例和深入演練，請參閱 [在 中建立您的應用程式 AWS SAM](#)。

sam list

此頁面提供 AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam list` 命令的參考資訊。

如需 的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)

`sam list` 命令會輸出有關無伺服器應用程式中資源和無伺服器應用程式狀態的重要資訊。在部署 `sam list` 前後使用 在本機和雲端開發期間提供協助。

Usage

```
$ sam list <options> <subcommand>
```

選項

--help, -h

顯示此訊息並結束。

子命令

endpoints

顯示 CloudFormation 堆疊中的雲端和本機端點清單。如需詳細資訊，請參閱 [sam list endpoints](#)。

resources

顯示部署 AWS CloudFormation 時在 中建立的 AWS Serverless Application Model (AWS SAM) 範本中的資源。如需詳細資訊，請參閱 [sam list resources](#)。

stack-outputs

顯示來自 AWS SAM 或 CloudFormation 範本的 CloudFormation 堆疊輸出。如需詳細資訊，請參閱 [sam list stack-outputs](#)。

sam list endpoints

此頁面提供 AWS Serverless Application Model 命令列界面 (AWS SAM CLI) `sam list endpoints` 子命令的參考資訊。

如需 的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)

`sam list endpoints` 子命令會顯示您 CloudFormation 堆疊中的雲端和本機端點清單。您可以透過 `sam local` 和 `sam sync` 命令與這些資源互動。

AWS Lambda 此命令支援 和 Amazon API Gateway 資源類型。

Note

為您的 Amazon API Gateway 資源設定時，支援自訂網域。此命令將輸出自訂網域，而不是預設端點。

Usage

```
$ sam list endpoints <options>
```

選項

`--config-env` *TEXT*

在要使用的組態檔案中指定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 SAM CLI 組態檔案](#)。

`--config-file` *TEXT*

組態檔案的路徑和檔案名稱，其中包含要使用的預設參數值。

預設值：samconfig.toml 在目前的工作目錄中。

如需關於組態檔案的詳細資訊，請參閱 [AWS SAM CLI 組態檔案](#)。

`--debug`

開啟偵錯記錄，以列印由具有時間戳記的 AWS SAM CLI 產生的偵錯訊息。

`--help, -h`

顯示此訊息並結束。

`--output [json|table]`

指定輸出結果的格式。

預設值：`table`

`--profile TEXT`

從登入資料檔案選取特定設定檔以取得 AWS 登入資料。

`--region TEXT`

設定服務 AWS 的區域。例如 `us-east-1`。

`--save-params`

將您在命令列提供的參數儲存至 AWS SAM 組態檔案。

`--stack-name TEXT`

部署堆疊的名稱 CloudFormation。您可以在應用程式的 `samconfig.toml` 檔案或指定的組態檔案中找到堆疊名稱。

未指定此選項時，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

從名為 `test-stack` 的 CloudFormation 堆疊中，以 `json` 格式顯示已部署資源端點的輸出。

```
$ 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

此頁面提供 AWS Serverless Application Model 命令列界面 (AWS SAM CLI) `sam list resources` 子命令的參考資訊。

如需 的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)

`sam list resources` 子命令會顯示部署時轉換 AWS CloudFormation 在 AWS SAM 中建立的 AWS Serverless Application Model (AWS SAM) 範本中的資源。

在部署之前 `sam list resources` 搭配 AWS SAM 範本使用，以查看要建立的資源。提供 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，會執行範本的本機轉換。此清單中包含將隨條件部署的資源，例如特定區域內的資源。

Usage

```
$ sam list resources <options>
```

選項

`--config-env TEXT`

在要使用的組態檔案中指定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 SAM CLI 組態檔案](#)。

--config-file *TEXT*

組態檔案的路徑和檔案名稱，其中包含要使用的預設參數值。

預設值：samconfig.toml在目前的工作目錄中。

如需關於組態檔案的詳細資訊，請參閱 [AWS SAM CLI 組態檔案](#)。

--debug

開啟偵錯記錄，以列印由具有時間戳記的 AWS SAM CLI產生的偵錯訊息。

--help, -h

顯示此訊息並結束。

--output [json|table]

指定輸出結果的格式。

預設值：table

--profile *TEXT*

從登入資料檔案選取特定設定檔以取得 AWS 登入資料。

--region *TEXT*

設定服務 AWS 的區域。例如 us-east-1。

--save-params

將您在命令列提供的參數儲存至 AWS SAM 組態檔案。

--stack-name *TEXT*

部署堆疊的名稱 CloudFormation。您可以在應用程式的 samconfig.toml 檔案或指定的組態檔案中找到堆疊名稱。

提供時，範本中的資源邏輯 IDs 將映射至其對應的實體 IDs CloudFormation。若要進一步了解實體 IDs，請參閱AWS CloudFormation 《使用者指南》中的[資源欄位](#)。

未指定此選項時，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

以資料表格式顯示 AWS SAM 範本中本機資源的輸出，以及從名為 `test-stack` 的 CloudFormation 堆疊中部署的資源。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

此頁面提供 AWS Serverless Application Model 命令列界面 (AWS SAM CLI) `sam list stack-outputs` 子命令的參考資訊。

如需的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)

`sam list stack-outputs` 子命令會顯示來自 AWS Serverless Application Model (AWS SAM) 或 CloudFormation 範本的 AWS CloudFormation 堆疊輸出。如需的詳細資訊 `Outputs`，請參閱 AWS CloudFormation 《使用者指南》中的 [輸出](#)。

Usage

```
$ sam list stack-outputs <options>
```

選項

`--config-env` *TEXT*

在要使用的組態檔案中指定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 SAM CLI 組態檔案](#)。

`--config-file` *TEXT*

組態檔案的路徑和檔案名稱，其中包含要使用的預設參數值。

預設值：samconfig.toml在目前的工作目錄中。

如需關於組態檔案的詳細資訊，請參閱 [AWS SAM CLI 組態檔案](#)。

`--debug`

開啟偵錯記錄，以列印由具有時間戳記的 AWS SAM CLI產生的偵錯訊息。

`--help`, `-h`

顯示此訊息並結束。

`--output` [json|table]

指定輸出結果的格式。

預設值：table

`--profile` *TEXT*

從登入資料檔案選取特定設定檔以取得 AWS 登入資料。

`--region` *TEXT*

設定服務 AWS 的區域。例如 us-east-1。

`--save-params`

將您在命令列提供的參數儲存至 AWS SAM 組態檔案。

`--stack-name` *TEXT*

部署堆疊的名稱 CloudFormation。您可以在應用程式的 `samconfig.toml` 檔案或指定的組態檔案中找到堆疊名稱。

此選項為必要。

範例

以資料表格式顯示 CloudFormation 堆疊中名為 `test-stack` 之資源的輸出。

```
$ sam list stack-outputs --stack-name test-stack --output table
```

OutputKey Description	OutputValue	
HelloWorldFunctionIamRole Implicit IAM Role created for Hello function	arn:aws:iam:: <i>account-number</i> :role/sam- app-test-list-HelloWorldFunctionRole- SRJDMJ6F7F41	World
HelloWorldApi Gateway endpoint URL for Prod for Hello World function	https://uj80uoe2o2.execute-api.us- east-1.amazonaws.com/Prod/hello/	API stage
HelloWorldFunction World Lambda Function ARN	arn:aws:lambda:us- east-1: <i>account-number</i> :function:sam-app- test-list- HelloWorldFunction-H85Y7yIV7ZLq	Hello

sam local callback

此頁面提供 AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam local callback` 命令的參考資訊。

如需 的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)

`sam local callback` 命令可讓您在開發和測試期間將回呼傳送至本機耐久函數執行。

Usage

```
$ sam local callback <subcommand> <options>
```

選項

--help, -h

顯示此訊息並結束。

子命令

succeed

將成功回呼傳送至持久的函數執行。如需詳細資訊，請參閱[sam local callback succeed](#)。

fail

將失敗回呼傳送至持久的函數執行。如需詳細資訊，請參閱[sam local callback fail](#)。

heartbeat

將活動訊號回呼傳送至持久的函數執行。如需詳細資訊，請參閱[sam local callback heartbeat](#)。

sam local callback succeed

將成功回呼傳送至持久的函數執行。

Usage

```
$ sam local callback succeed CALLBACK_ID [OPTIONS]
```

必要的引數

CALLBACK_ID

要傳送成功回應的回呼的唯一識別符。

選項

-r, --result TEXT

成功結果承載做為字串。

--region TEXT

設定服務的 AWS 區域 (例如 us-east-1)。

--profile TEXT

從登入資料檔案選取特定設定檔以取得 AWS 登入資料。

--config-env TEXT

在組態檔案中指定預設參數值的環境名稱。預設 : default

--config-file TEXT

包含預設參數值的組態檔案。預設 : samconfig.toml

--save-params

將透過命令列提供的參數儲存至組態檔案。

--beta-features / --no-beta-features

啟用/停用 Beta 版功能。

--debug

開啟偵錯記錄以列印 CLI AWS SAM 產生的偵錯訊息，並顯示時間戳記。

-h, --help

顯示此訊息並結束。

範例

傳送沒有結果的成功回呼：

```
$ sam local callback succeed my-callback-id
```

傳送成功回呼與結果：

```
$ sam local callback succeed my-callback-id --result 'Task completed successfully'
```

使用簡短選項傳送成功回呼：

```
$ sam local callback succeed my-callback-id -r 'Success result'
```

`sam local callback fail`

將失敗回呼傳送至持久的函數執行。

Usage

```
$ sam local callback fail CALLBACK_ID [OPTIONS]
```

必要的引數

`CALLBACK_ID`

要傳送失敗回應的回呼的唯一識別符。

選項

`--error-data TEXT`

其他錯誤資料。

`--stack-trace TEXT`

堆疊追蹤項目。此選項可指定多次。

`--error-type TEXT`

錯誤類型。

`--error-message TEXT`

詳細的錯誤訊息。

`--region TEXT`

設定服務的 AWS 區域 (例如 `us-east-1`)。

`--profile TEXT`

從登入資料檔案選取特定設定檔以取得 AWS 登入資料。

--config-env TEXT

在組態檔案中指定預設參數值的環境名稱。預設：default

--config-file TEXT

包含預設參數值的組態檔案。預設：samconfig.toml

--save-params

將透過命令列提供的參數儲存至組態檔案。

--beta-features / --no-beta-features

啟用/停用 Beta 版功能。

--debug

開啟偵錯記錄以列印 CLI AWS SAM 產生的偵錯訊息，並顯示時間戳記。

-h, --help

顯示此訊息並結束。

範例

傳送沒有參數的失敗回呼：

```
$ sam local callback fail my-callback-id
```

傳送失敗回呼並顯示錯誤訊息：

```
$ sam local callback fail my-callback-id --error-message 'Task failed'
```

使用所有參數傳送失敗回呼：

```
$ sam local callback fail my-callback-id --error-message 'Task failed' --error-type  
'ValidationError' --stack-trace 'at line 42' --error-data '{"code": 500}'
```

sam local callback heartbeat

將活動訊號回呼傳送至持久的函數執行。

Usage

```
$ sam local callback heartbeat CALLBACK_ID [OPTIONS]
```

必要的引數

CALLBACK_ID

要向其傳送活動訊號的回呼的唯一識別符。

選項

`--region TEXT`

設定服務的 AWS 區域 (例如 us-east-1)。

`--profile TEXT`

從登入資料檔案選取特定設定檔以取得 AWS 登入資料。

`--config-env TEXT`

在組態檔案中指定預設參數值的環境名稱。預設 : default

`--config-file TEXT`

包含預設參數值的組態檔案。預設 : samconfig.toml

`--save-params`

將透過命令列提供的參數儲存至組態檔案。

`--beta-features / --no-beta-features`

啟用/停用 Beta 版功能。

`--debug`

開啟偵錯記錄以列印 CLI AWS SAM 產生的偵錯訊息，並顯示時間戳記。

`-h, --help`

顯示此訊息並結束。

範例

傳送活動訊號回呼：

```
$ sam local callback heartbeat my-callback-id
```

sam local execution

此頁面提供 AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam local execution` 命令的參考資訊。

如需 的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)

`sam local execution` 命令可讓您在開發和測試期間管理和檢查本機耐久函數執行。

Note

這些命令可能不需要存取 AWS 登入資料。

Usage

```
$ sam local execution <subcommand> <options>
```

選項

`--help`, `-h`

顯示此訊息並結束。

子命令

get

取得耐久函數執行的詳細資訊。如需詳細資訊，請參閱[sam local execution get](#)。

history

取得持久函數執行的執行歷史記錄。如需詳細資訊，請參閱[sam local execution history](#)。

stop

停止耐用的函數執行。如需詳細資訊，請參閱[sam local execution stop](#)。

sam local execution get

擷取特定耐用函數執行的詳細資訊。

Note

此命令可能不需要存取 AWS 登入資料。

Usage

```
$ sam local execution get DURABLE_EXECUTION_ARN [OPTIONS]
```

必要的引數

DURABLE_EXECUTION_ARN

要擷取詳細資訊之耐久函數執行的 Amazon Resource Name (ARN)。

選項

`--format [summary|json]`

輸出格式。預設：summary

`--region TEXT`

設定服務的 AWS 區域 (例如 us-east-1)。

`--profile TEXT`

從登入資料檔案選取特定設定檔以取得 AWS 登入資料。

`--config-env TEXT`

在組態檔案中指定預設參數值的環境名稱。預設：default

`--config-file TEXT`

包含預設參數值的組態檔案。預設：samconfig.toml

`--save-params`

將透過命令列提供的參數儲存至組態檔案。

`--beta-features / --no-beta-features`

啟用/停用 Beta 版功能。

`--debug`

開啟偵錯記錄以列印 CLI AWS SAM 產生的偵錯訊息，並顯示時間戳記。

`-h, --help`

顯示此訊息並結束。

範例

取得預設摘要格式的執行詳細資訊：

```
$ sam local execution get arn:aws:lambda:us-east-1:123456789012:function:MyFunction:
$LATEST/durable-execution/my-execution-name/my-execution-id
```

取得 JSON 格式的執行詳細資訊：

```
$ sam local execution get arn:aws:lambda:us-east-1:123456789012:function:MyFunction:
$LATEST/durable-execution/my-execution-name/my-execution-id --format json
```

sam local execution history

取得持久函數執行的執行歷史記錄。

Usage

```
$ sam local execution history EXECUTION_ID [OPTIONS]
```

必要的引數

EXECUTION_ID

要擷取歷史記錄之耐久函數執行的唯一識別符。

選項

`--format [table|json]`

輸出格式。預設：table

--region TEXT

設定服務的 AWS 區域 (例如 us-east-1)。

--profile TEXT

從登入資料檔案選取特定設定檔以取得 AWS 登入資料。

--config-env TEXT

在組態檔案中指定預設參數值的環境名稱。預設 : default

--config-file TEXT

包含預設參數值的組態檔案。預設 : samconfig.toml

--save-params

將透過命令列提供的參數儲存至組態檔案。

--beta-features / --no-beta-features

啟用/停用 Beta 版功能。

--debug

開啟偵錯記錄以列印 CLI AWS SAM 產生的偵錯訊息，並顯示時間戳記。

-h, --help

顯示此訊息並結束。

範例

取得資料表格式的執行歷史記錄：

```
$ sam local execution history my-execution-id
```

取得 JSON 格式的執行歷史記錄：

```
$ sam local execution history my-execution-id --format json
```

sam local execution stop

停止執行中的耐用函數執行。

 Note

此命令可能不需要存取 AWS 登入資料。

Usage

```
$ sam local execution stop DURABLE_EXECUTION_ARN [OPTIONS]
```

必要的引數

DURABLE_EXECUTION_ARN

要停止之耐久函數執行的 Amazon Resource Name (ARN)。

選項

--error-message TEXT

要與停止的執行建立關聯的錯誤訊息。

--error-type TEXT

要與停止的執行建立關聯的錯誤類型。

--error-data TEXT

要與停止的執行建立關聯的錯誤資料。

--stack-trace TEXT

堆疊追蹤項目。此選項可指定多次。

--region TEXT

設定服務的 AWS 區域 (例如 us-east-1)。

--profile TEXT

從登入資料檔案選取特定設定檔以取得 AWS 登入資料。

--config-env TEXT

在組態檔案中指定預設參數值的環境名稱。預設：default

--config-file TEXT

包含預設參數值的組態檔案。預設：samconfig.toml

--save-params

將透過命令列提供的參數儲存至組態檔案。

--beta-features / --no-beta-features

啟用/停用 Beta 版功能。

--debug

開啟偵錯記錄以列印 CLI AWS SAM 產生的偵錯訊息，並顯示時間戳記。

-h, --help

顯示此訊息並結束。

範例

在沒有錯誤詳細資訊的情況下停止執行：

```
$ sam local execution stop arn:aws:lambda:us-east-1:123456789012:function:MyFunction:
$LATEST/durable-execution/my-execution-name/my-execution-id
```

使用錯誤訊息和類型停止執行：

```
$ sam local execution stop arn:aws:lambda:us-east-1:123456789012:function:MyFunction:
$LATEST/durable-execution/my-execution-name/my-execution-id --error-message "Execution
cancelled" --error-type "UserCancellation"
```

使用完整錯誤詳細資訊和堆疊追蹤停止執行：

```
$ sam local execution stop arn:aws:lambda:us-east-1:123456789012:function:MyFunction:
$LATEST/durable-execution/my-execution-name/my-execution-id --error-message "Task
failed" --error-type "TaskFailure" --error-data '{"reason":"timeout"}' --stack-trace
"at function1()" --stack-trace "at function2()"
```

sam local generate-event

此頁面提供 AWS Serverless Application Model 命令列界面 (AWS SAM CLI) sam local generate-event 子命令的參考資訊。

- 如需的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)
- 如需使用 AWS SAM CLI 的 `sam local generate-event` 命令的文件，請參閱 [使用 進行測試的簡介 sam local generate-event](#)。

`sam local generate-event` 子命令會為支援的 產生事件承載範例 AWS 服務。

Usage

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

選項

`--config-env TEXT`

在要使用的組態檔案中指定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 SAM CLI 組態檔案](#)。

`--config-file PATH`

組態檔案的路徑和檔案名稱，其中包含要使用的預設參數值。預設值位於專案目錄的根 `samconfig.toml` 目錄中。如需關於組態檔案的詳細資訊，請參閱 [AWS SAM CLI 組態檔案](#)。

`--help`

顯示此訊息並結束。

服務

若要查看支援的 服務清單，請執行下列動作：

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

事件

若要查看可為每個服務產生的支援事件清單，請執行下列動作：

```
$ 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，請執行下列動作：

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

範例

如需使用 `sam local generate-event` 子命令的範例，請參閱 [產生範例事件](#)。

sam local invoke

此頁面提供 AWS Serverless Application Model 命令列界面 (AWS SAM CLI) `sam local invoke` 子命令的參考資訊。

- 如需的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)
- 如需使用 AWS SAM CLI `sam local invoke` 子命令的文件，請參閱 [使用 進行測試的簡介 sam local invoke](#)。

`sam local invoke` 子命令會在本機啟動 AWS Lambda 事件函數的一次性調用。

Note

不建議在不受信任的程式碼中使用 SAM CLI 的本機調用功能。若要與本機環境完全隔離，請直接在 Lambda 服務中執行程式碼。

Note

對於耐用的函數，`sam local invoke` 支援使用自動檢查點和重播的狀態執行。容器會在持久的函數執行期間持續執行，以處理狀態持久性和恢復。

Usage

```
$ 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

Resource ID (資源 ID)

要叫用的 Lambda 函數 ID。

此為選用引數。如果您的應用程式包含單一 Lambda 函數，CLI AWS SAM 會叫用它。如果您的應用程式包含多個函數，請提供要叫用的函數 ID。

有效值：資源的邏輯 ID 或資源 ARN。

選項

`--add-host` *LIST*

將主機名稱傳遞至 Docker 容器主機檔案的 IP 地址映射。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 Beta 版功能。

`--config-env` *TEXT*

在要使用的組態檔案中指定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 SAM CLI 組態檔案](#)。

`--config-file` *PATH*

組態檔案的路徑和檔案名稱，其中包含要使用的預設參數值。專案目錄根目錄中的預設值為「samconfig.toml」。如需關於組態檔案的詳細資訊，請參閱 [AWS SAM CLI 組態檔案](#)。

`--container-env-vars`

(選用) 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。如果您想要在 macOS 上的 Docker 容器中執行 AWS SAM CLI，您可以指定 host.docker.internal。如果您想要在與不同的主機上執行容器 AWS SAM CLI，您可以指定遠端主機的 IP 地址。

`--container-host-interface` *TEXT*

容器連接埠應繫結之主機網路介面的 IP 地址。預設值為 127.0.0.1。使用 0.0.0.0 繫結至所有介面。

`--container-dns` *TEXT*

指定 Docker 容器的自訂 DNS 伺服器。此參數可以多次傳遞，以指定多個 DNS 伺服器。

Example

範例：`--container-dns 8.8.8.8`

`--debug`

開啟偵錯記錄以列印產生的偵錯訊息 AWS SAM CLI，並顯示時間戳記。

`--debug-args` *TEXT*

要傳遞給除錯器的其他引數。

`--debug-port, -d` *TEXT*

指定時，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*

掛載至 Lambda 容器之偵錯工具的主機路徑。

`--docker-network` *TEXT*

Lambda Docker 容器應連線的現有 Docker 網路的名稱或 ID，以及預設橋接網路。如果未指定，Lambda 容器只會連線到預設橋接器 Docker 網路。

`--docker-volume-basedir, -v` *TEXT*

AWS SAM 檔案存在的基礎目錄位置。如果 Docker 在遠端機器上執行，您必須掛載 AWS SAM Docker 機器上檔案所在的路徑，並修改此值以符合遠端機器。

`--durable-execution-name` *TEXT*

持久性執行的名稱（僅適用於持久性函數）

`--env-vars, -n` *PATH*

包含 Lambda 函數環境變數值的 JSON 或 .env 檔案。系統會自動偵測檔案格式。如需環境變數檔案的詳細資訊，請參閱 [環境變數檔案](#)。

`--event, -e PATH`

包含事件資料的 JSON 檔案，會在叫用時傳遞至 Lambda 函數。如果您未指定此選項，則不會假設任何事件。若要從輸入 JSONstdin，您必須傳入值 '-'。如需不同 AWS 服務的事件訊息格式詳細資訊，請參閱《AWS Lambda 開發人員指南》中的[使用其他服務](#)。

`--force-image-build`

指定 AWS SAM CLI 是否應重建用於使用 layer 叫用 Lambda 函數的映像。

`--help`

顯示此訊息並結束。

`--hook-name TEXT`

用於擴展 AWS SAM CLI 功能的勾點名稱。

接受的值：terraform。

`--invoke-image TEXT`

您要用於本機函數調用之容器映像的 URI。

對於具有 `Lambda 函數PackageType: Zip`，這會覆寫執行時間基礎映像。根據預設，會從 Amazon ECR Public AWS SAM 提取容器映像（列於 [的影像儲存庫 AWS SAM](#)）。使用此選項從另一個位置提取映像。

對於具有 `Lambda 函數PackageType: Image`，這會覆寫建置範本 `ImageUri` 中的。使用此選項可在本機叫用不同的應用程式映像，而不修改您的範本。

您可以多次指定此選項。此選項的每個執行個體都可以採用字串或鍵值對。如果您指定字串，則容器映像的 URI 會用於應用程式中的所有函數。例如 `sam local invoke --invoke-image public.ecr.aws/sam/emu-python3.13`。如果您指定金鑰/值對，則金鑰是資源名稱，而值是用於該資源的容器映像的 URI。例如 `sam local invoke --invoke-image public.ecr.aws/sam/emu-python3.13 --invoke-image Function1=amazon/aws-sam-cli-emulation-image-python3.13`。使用鍵/值對，您可以為不同的資源指定不同的容器映像。

`--layer-cache-basedir DIRECTORY`

指定下載範本所用層的基礎目錄位置。

`--log-file, -l TEXT`

要傳送執行期日誌的日誌檔案。

--mount-symlinks

確保 AWS SAM CLI 一律掛載檔案中存在的符號連結來建置或叫用。這僅適用於頂層目錄上的符號連結（即直接在函數根目錄上的符號連結）。根據預設，不會掛載符號連結，除了在 NodeJS `node_modules` 中使用 `build-in-source` 時需要的符號連結。

--no-event

使用空白事件叫用 函數。

--no-memory-limit

在本機調用期間移除容器中的記憶體限制，即使 AWS SAM 範本中已設定記憶體也一樣。

--parameter-overrides

包含 CloudFormation 參數的字串會覆寫編碼為鍵值對的字串。使用與 AWS Command Line Interface () 相同的格式 AWS CLI。格式 AWS SAM CLI 是明確的索引鍵和值關鍵字，每個覆寫都會以空格分隔。以下是兩個範例：

- `--parameter-overrides ParameterKey=hello,ParameterValue=world`
- `--parameter-overrides ParameterKey=hello,ParameterValue=world
ParameterKey=example1,ParameterValue=example2
ParameterKey=apple,ParameterValue=banana`

--profile *TEXT*

從您的登入資料檔案中取得 AWS 登入資料的特定設定檔。

--region *TEXT*

AWS 要部署的區域。例如 `us-east-1`。

--runtime *TEXT*

使用指定的執行時間在本機叫用 Lambda 函數。這會覆寫 `template.yml` 檔案中定義的執行時間。這也允許使用不同的執行時間測試 Lambda 函數，而無需修改原始函數組態。

--save-params

將您在命令列提供的參數儲存至 AWS SAM 組態檔案。

--shutdown

在調用完成後模擬關機事件，以測試關機行為的延伸處理。

--skip-prepare-infra

如果未進行任何基礎設施變更，請略過準備階段。使用 搭配 --hook-name 選項。

--skip-pull-image

根據預設，會 AWS SAM CLI 檢查 Lambda 最新的遠端執行期環境，並自動更新本機映像以保持同步。

指定此選項可略過提取 Lambda 執行時間環境的最新 Docker 映像。

--template, -t *PATH*

AWS SAM 範本檔案。

此選項與 不相容 --hook-name。

Note

如果您指定此選項，只會 AWS SAM 載入範本及其指向的本機資源。

--tenant-id *TEXT*

多租用戶 Lambda 函數的租用戶 ID。用來確保不同租用戶之間的運算隔離。叫用使用租戶隔離模式設定的函數時為必要。

--terraform-plan-file

搭配 使用 時，本機 Terraform 計劃檔案的 AWS SAM CLI 相對或絕對路徑 Terraform Cloud。此選項需要 --hook-name 將 設定為 terraform。

範例

下列範例使用產生的事件進行本機測試，方法是使用 s3.json 事件在本機叫用 Lambda 函數

```
$ sam local invoke --event events/s3.json S3JsonLoggerFunction
```

下列範例 HelloWorldFunction 使用 Python 3.11 執行時間測試函數

```
$ sam local invoke --runtime python3.11 HelloWorldFunction
```

下列範例會測試 HelloWorldFunction 具有耐久執行名稱的函數

```
$ sam local invoke HelloWorldFunction --durable-execution-name my-execution
```

sam local start-api

此頁面提供 AWS Serverless Application Model 命令列界面 (AWS SAM CLI) `sam local start-api` 子命令的參考資訊。

- 如需 的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)
- 如需使用 AWS SAM CLI `sam local start-api` 子命令的文件，請參閱 [使用 進行測試的簡介 sam local start-api](#)。

`sam local start-api` 子命令會在本機執行您的 AWS Lambda 事件函數，以透過本機 HTTP 伺服器主機進行測試。

Usage

```
$ sam local start-api <options>
```

選項

`--add-host` *LIST*

將主機名稱傳遞至 Docker 容器主機檔案的 IP 地址映射。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 Beta 版功能。

`--config-env` *TEXT*

在要使用的組態檔案中指定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 SAM CLI 組態檔案](#)。

`--config-file` *PATH*

組態檔案的路徑和檔案名稱，其中包含要使用的預設參數值。預設值為專案目錄根目錄中的 "samconfig.toml"。如需關於組態檔案的詳細資訊，請參閱 [AWS SAM CLI 組態檔案](#)。

--container-env-vars

選用。在本機偵錯時，將環境變數傳遞至映像容器。

--container-host *TEXT*

本機模擬 Lambda 容器的主機。預設值為 localhost。如果您想要在 macOS 上的 Docker 容器中執行 AWS SAM CLI，您可以指定 host.docker.internal。如果您想要在與不同的主機上執行容器 AWS SAM CLI，您可以指定遠端主機的 IP 地址。

--container-host-interface *TEXT*

容器連接埠應繫結之主機網路介面的 IP 地址。預設值為 127.0.0.1。使用 0.0.0.0 繫結至所有介面。

--container-dns *TEXT*

指定 Docker 容器的自訂 DNS 伺服器。此參數可以多次傳遞，以指定多個 DNS 伺服器。

Example

範例：**--container-dns 8.8.8.8**

--debug

開啟偵錯記錄以列印 AWS SAM CLI 產生的偵錯訊息，並顯示時間戳記。

--debug-args *TEXT*

要傳遞給除錯器的其他引數。

--debug-function

選用。指定指定時要套用偵錯選項的 Lambda --warm-containers 函數。此參數適用於 --debug-port、--debugger-path 和 --debug-args。

--debug-port, -d *TEXT*

指定時，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*

要掛載到 Lambda 容器的偵錯工具的主機路徑。

--docker-network *TEXT*

Lambda Docker 容器應連線的現有 Docker 網路的名稱或 ID，以及預設橋接網路。如果未指定，Lambda 容器只會連線到預設橋接 Docker 網路。

`--docker-volume-basedir, -v TEXT`

AWS SAM 檔案存在的基礎目錄位置。如果 Docker 在遠端機器上執行，您必須掛載 AWS SAM Docker 機器上檔案所在的路徑，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 或 .env 檔案。系統會自動偵測檔案格式。

`--force-image-build`

指定 是否 AWS SAM CLI應重建用於使用 layer 叫用函數的映像。

`--help`

顯示此訊息並結束。

`--hook-name TEXT`

用於擴展 AWS SAM CLI功能的勾點名稱。

接受的值：terraform。

`--host TEXT`

要繫結的本機主機名稱或 IP 地址（預設：'127.0.0.1'）。

`--invoke-image TEXT`

您要用於 Lambda 函數的容器映像的 URI。

對於具有 的 Lambda 函數PackageType: Zip，這會覆寫執行時間基礎映像。根據預設，會從 Amazon ECR Public AWS SAM 提取容器映像。使用此選項從另一個位置提取映像。

對於具有 的 Lambda 函數PackageType: Image，這會覆寫建置範本ImageUri中的 。使用此選項可在本機叫用不同的應用程式映像，而不修改您的範本。

您可以多次指定此選項。此選項的每個執行個體都可以採用字串或鍵值對。如果您指定字串，則容器映像的 URI 會用於應用程式中的所有函數。例如 `sam local start-api --invoke-image public.ecr.aws/sam/emu-python3.13`。如果您指定金鑰/值對，則金鑰是資源名稱，而值是用於該資源的容器映像的 URI。例如 `sam local start-api --invoke-image public.ecr.aws/sam/emu-python3.13 --invoke-image Function1=amazon/aws-sam-cli-emulation-image-python3.13`。使用鍵/值對，您可以為不同的資源指定不同的容器映像。

`--layer-cache-basedir DIRECTORY`

指定下載範本所用 Layers 的位置基數。

`--log-file, -l TEXT`

要傳送執行期日誌的日誌檔案。

`--no-memory-limit`

在本機調用期間移除容器中的記憶體限制，即使 AWS SAM 範本中已設定記憶體也一樣。

`--no-watch`

與 `--warm-containers` 搭配使用時，停用檔案監看。指定時，AWS SAM CLI 不會監控原始程式碼或範本的變更，而且暖容器不會在程式碼修改時自動重新啟動。若要取得本機變更，請停止並重新執行命令。

當檔案監看造成高 CPU 或 I/O 額外負荷時，例如大型單一儲存、具有防毒軟體掃描的環境，或多個 Lambda 函數共用的專案，請使用此選項 `CodeUri`。

如果 `--warm-containers` 未指定，則會忽略此選項。

`--parameter-overrides`

包含 CloudFormation 參數的字串會覆寫編碼為鍵值對的字串。使用與 AWS Command Line Interface () 相同的格式 AWS CLI。格式 AWS SAM CLI 是明確的索引鍵和值關鍵字，每個覆寫都會以空格分隔。以下是兩個範例：

- `--parameter-overrides ParameterKey=hello,ParameterValue=world`
- `--parameter-overrides ParameterKey=hello,ParameterValue=world
ParameterKey=example1,ParameterValue=example2
ParameterKey=apple,ParameterValue=banana`

`--port, -p INTEGER`

要接聽的本機連接埠號碼（預設值：'3000'）。

`--profile TEXT`

從您的登入資料檔案中取得 AWS 登入資料的特定設定檔。

`--region TEXT`

AWS 要部署的區域。例如 `us-east-1`。

`--save-params`

將您在命令列提供的參數儲存至 AWS SAM 組態檔案。

--shutdown

在調用完成後模擬關機事件，以測試關機行為的延伸處理。

--skip-prepare-infra

如果未進行任何基礎設施變更，請略過準備階段。使用 搭配 `--hook-name` 選項。

--skip-pull-image

指定 CLI 是否應該略過提取 Lambda 執行時間的最新 Docker 映像。

--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 `--ssl-key-file` 選項。

--ssl-key-file *PATH*

SSL 金鑰檔案的路徑（預設：無）。使用此選項時，也必須使用 `--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）檔案會顯示在 /。

--template, -t *PATH*

AWS SAM 範本檔案。

 Note

如果您指定此選項，只會 AWS SAM 載入範本及其指向的本機資源。

--terraform-plan-file

搭配 使用 時，本機 Terraform 計劃檔案的 AWS SAM CLI 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 SAM CLI 管理每個函數的容器。

有兩個可用選項：

EAGER：所有函數的容器會在啟動時載入，並在叫用之間保留。

LAZY：只有在第一次叫用每個函數時，才會載入容器。這些容器會持續進行其他調用。

範例

下列範例會啟動本機伺服器，讓您透過 API 測試應用程式。若要讓此命令正常運作，必須安裝應用程式，且 Docker 必須正在執行。

```
$ sam local start-api --port 3000
```

sam local start-lambda

此頁面提供 AWS Serverless Application Model 命令列界面 (AWS SAM CLI) `sam local start-lambda` 子命令的參考資訊。

- 如需的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)
- 如需使用 AWS SAM CLI `sam local start-lambda` 子命令的文件，請參閱 [使用 進行測試的簡介 sam local start-lambda](#)。

`sam local start-lambda` 子命令會啟動要模擬的本機端點 AWS Lambda。

Usage

```
$ sam local start-lambda <options>
```

選項

`--add-host` *LIST*

將主機名稱傳遞至 Docker 容器主機檔案的 IP 地址映射。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 Beta 版功能。

`--config-env` *TEXT*

在要使用的組態檔案中指定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 SAM CLI 組態檔案](#)。

`--config-file` *PATH*

組態檔案的路徑和檔案名稱，其中包含要使用的預設參數值。預設值為專案目錄根目錄中的 "samconfig.toml"。如需關於組態檔案的詳細資訊，請參閱 [AWS SAM CLI 組態檔案](#)。

`--container-env-vars`

選用。在本機偵錯時，將環境變數傳遞至映像容器。

`--container-host` *TEXT*

本機模擬 Lambda 容器的主機。預設值為 localhost。如果您想要在 macOS 上的 Docker 容器中執行 AWS SAM CLI，您可以指定 host.docker.internal。如果您想要在與不同的主機上執行容器 AWS SAM CLI，您可以指定遠端主機 IP 地址。

`--container-host-interface` *TEXT*

容器連接埠應繫結之主機網路介面的 IP 地址。預設值為 127.0.0.1。使用 0.0.0.0 繫結至所有介面。

`--container-dns` *TEXT*

指定 Docker 容器的自訂 DNS 伺服器。此參數可以多次傳遞，以指定多個 DNS 伺服器。

Example

範例：`--container-dns 8.8.8.8`

`--debug`

開啟偵錯記錄以列印 AWS SAM CLI 產生的偵錯訊息，並顯示時間戳記。

`--debug-args` *TEXT*

要傳遞給除錯器的其他引數。

`--debug-function`

選用。指定指定時要套用偵錯選項的 Lambda `--warm-containers` 事件函數。此參數適用於 `--debug-port`、`--debugger-path` 和 `--debug-args`。

`--debug-port, -d` *TEXT*

指定時，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*

要掛載至 Lambda 容器之偵錯工具的主機路徑。

`--docker-network` *TEXT*

Lambda Docker 容器應連線的現有 Docker 網路的名稱或 ID，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橋接器 Docker 網路。

`--docker-volume-basedir, -v` *TEXT*

AWS SAM 檔案存在的基礎目錄位置。如果 Docker 在遠端機器上執行，您必須掛載 AWS SAM Docker 機器上檔案所在的路徑，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 或 .env 檔案。系統會自動偵測檔案格式。

`--force-image-build`

指定 是否 CLI 應重建用於使用 layer 叫用函數的映像。

`--help`

顯示此訊息並結束。

`--hook-name` *TEXT*

用於擴展 AWS SAM CLI 功能的勾點名稱。

接受的值：terraform。

`--host` *TEXT*

要繫結的本機主機名稱或 IP 地址（預設：'127.0.0.1'）。

`--invoke-image` *TEXT*

您要用於本機函數調用之容器映像的 URI。

對於具有 的 Lambda 函數 `PackageType: Zip`，這會覆寫執行時間基礎映像。根據預設，會從 Amazon ECR Public AWS SAM 提取容器映像。使用此選項從另一個位置提取映像。

對於具有 的 Lambda 函數 `PackageType: Image`，這會覆寫建置範本 `ImageUri` 中的。使用此選項可在本機叫用不同的應用程式映像，而不修改您的範本。

您可以多次指定此選項。此選項的每個執行個體都可以採用字串或鍵值對。如果您指定字串，則容器映像的 URI 會用於應用程式中的所有函數。例如 `sam local start-lambda --invoke-image public.ecr.aws/sam/emu-python3.13`。如果您指定金鑰/值對，則金鑰是資源名稱，而值是用於該資源的容器映像的 URI。例如 `sam local start-lambda --invoke-image public.ecr.aws/sam/emu-python3.13 --invoke-image Function1=amazon/aws-`

`sam-cli-emulation-image-python3.13`。使用鍵/值對，您可以為不同的資源指定不同的容器映像。

`--layer-cache-basedir` *DIRECTORY*

指定下載範本所用圖層的位置基數。

`--log-file, -l` *TEXT*

要傳送執行期日誌的日誌檔案。

`--no-memory-limit`

在本機調用期間移除容器中的記憶體限制，即使 AWS SAM 範本中已設定記憶體也一樣。

`--no-watch`

與 `--warm-containers` 搭配使用時，停用檔案監看。指定時，AWS SAM CLI 不會監控原始程式碼或範本的變更，而且暖容器不會在程式碼修改時自動重新啟動。若要取得本機變更，請停止並重新執行命令。

當檔案監看造成高 CPU 或 I/O 額外負荷時，例如大型單一儲存、具有防毒軟體掃描的環境，或多個 Lambda 函數共用的專案，請使用此選項 `CodeUri`。

如果 `--warm-containers` 未指定，則會忽略此選項。

`--parameter-overrides`

包含 CloudFormation 參數的字串會覆寫編碼為鍵值對的字串。使用與 AWS Command Line Interface () 相同的格式 AWS CLI。格式 AWS SAM CLI 是明確的索引鍵和值關鍵字，每個覆寫都會以空格分隔。以下是兩個範例：

- `--parameter-overrides ParameterKey=hello,ParameterValue=world`
- `--parameter-overrides ParameterKey=hello,ParameterValue=world
ParameterKey=example1,ParameterValue=example2
ParameterKey=apple,ParameterValue=banana`

`--port, -p` *INTEGER*

要接聽的本機連接埠號碼（預設值：'3001'）。

`--profile` *TEXT*

從您的登入資料檔案中取得 AWS 登入資料的特定設定檔。

`--region` *TEXT*

AWS 要部署的區域。例如 `us-east-1`。

--save-params

將您在命令列提供的參數儲存至 AWS SAM 組態檔案。

--shutdown

在調用完成後模擬關機事件，以測試關機行為的延伸處理。

--skip-prepare-infra

如果未進行任何基礎設施變更，請略過準備階段。使用 搭配 --hook-name 選項。

--skip-pull-image

指定 是否CLI應該略過提取 Lambda 執行時間的最新 Docker 映像。

--template, -t *PATH*

AWS SAM 範本檔案。

Note

如果您指定此選項，只會 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

搭配 使用 時，本機Terraform計劃檔案的 AWS SAM CLI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 SAM CLI管理每個函數的容器。

有兩個可用選項：

- EAGER：所有函數的容器會在啟動時載入，並在叫用之間保留。
- LAZY：只有在第一次叫用每個函數時，才會載入容器。這些容器會持續進行其他調用。

範例

如需使用 sam local start-lambda 子命令的詳細範例和深入演練，請參閱 [使用 進行測試的簡介 sam local start-lambda](#)。

sam logs

此頁面提供 AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam logs` 命令的參考資訊。

如需 的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)

`sam logs` 命令會擷取 AWS Lambda 事件函數產生的日誌。

Usage

```
$ sam logs <options>
```

選項

`--config-env` *TEXT*

在要使用的組態檔案中指定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 SAM CLI 組態檔案](#)。

`--config-file` *PATH*

組態檔案的路徑和檔案名稱，其中包含要使用的預設參數值。預設值為專案目錄根目錄中的 "samconfig.toml"。如需關於組態檔案的詳細資訊，請參閱 [AWS SAM CLI 組態檔案](#)。

`--cw-log-group` *LIST*

包含您指定的 CloudWatch Logs 日誌群組中的日誌。如果您指定此選項與 `--log-group-name`，除了具名資源的日誌之外，還 AWS SAM 包含來自指定日誌群組的日誌。

`--debug`

開啟偵錯記錄以列印 AWS SAM CLI 產生的偵錯訊息，並顯示時間戳記。

`---end-time, e` *TEXT*

擷取截至目前為止的日誌。時間可以是「5 分鐘前」、「明天」等相對值，或「2018-01-01 10:10:10」等格式化時間戳記。

`--filter` *TEXT*

可讓您指定表達式，以快速尋找符合日誌事件中詞彙、片語或值的日誌。這可以是簡單的關鍵字（例如「錯誤」）或 Amazon CloudWatch Logs 支援的模式。如需語法，請參閱 [Amazon CloudWatch Logs 文件](#)。

`--help`

顯示此訊息並結束。

`--include-traces`

在日誌輸出中包含 X-Ray 追蹤。

`--name, -n TEXT`

要擷取日誌的資源名稱。如果此資源是 CloudFormation 堆疊的一部分，這可以是 CloudFormation/ AWS SAM template 中函數資源的邏輯 ID。您可以再次重複 參數來提供多個名稱。如果資源位於巢狀堆疊中，則可以在巢狀堆疊名稱前面加上名稱，以從該資源提取日誌 (NestedStackLogicalId/ ResourceLogicalId)。如果未提供資源名稱，則會掃描指定的堆疊，並提取所有支援資源的日誌資訊。如果您未指定此選項，會為您指定的堆疊中的所有資源 AWS SAM 擷取日誌。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`

指定日誌的輸出格式。若要列印格式化日誌，請指定 `text`。若要將日誌列印為 JSON，請指定 `json`。

`--profile TEXT`

從您的登入資料檔案中取得 AWS 登入資料的特定設定檔。

`--region TEXT`

AWS 要部署的區域。例如 `us-east-1`。

`--save-params`

將您在命令列提供的參數儲存至 AWS SAM 組態檔案。

`--stack-name` *TEXT*

資源所屬的 CloudFormation 堆疊名稱。

`--start-time, -s` *TEXT*

擷取目前開始的日誌。時間可以是「5 分鐘前」、「昨天」等相對值，或「2018-01-01 10:10:10」等格式化時間戳記。預設為 '10 分鐘前'。

`--tail, -t`

尾隨日誌輸出。這會忽略結束時間引數，並在日誌可用時繼續擷取日誌。

範例

當您的函數是 CloudFormation 堆疊的一部分時，您可以在指定堆疊名稱時使用函數的邏輯 ID 來擷取日誌。

```
$ sam logs -n HelloWorldFunction --stack-name myStack
```

使用 `-s` (`--start-time`) 和 `-e` (`--end-time`) 選項檢視特定時間範圍的日誌。

```
$ sam logs -n HelloWorldFunction --stack-name myStack -s '10min ago' -e '2min ago'
```

您也可以新增 `--tail` 選項來等待新的日誌，並在它們送達時看到它們。

```
$ sam logs -n HelloWorldFunction --stack-name myStack --tail
```

使用 `--filter` 選項快速尋找符合日誌事件中詞彙、片語或值的日誌。

```
$ sam logs -n HelloWorldFunction --stack-name myStack --filter "error"
```

檢視子堆疊中資源的日誌。

```
$ sam logs --stack-name myStack -n childStack/HelloWorldFunction
```

您應用程式中所有支援資源的自訂日誌。

```
$ sam logs --stack-name sam-app --tail
```

擷取應用程式中特定 Lambda 函數和 API Gateway API 的日誌。

```
$ sam logs --stack-name sam-app --name HelloWorldFunction --name HelloWorldRestApi
```

擷取應用程式中所有支援資源的日誌，以及來自指定日誌群組的日誌。

```
$ sam logs --cw-log-group /aws/lambda/myfunction-123 --cw-log-group /aws/lambda/myfunction-456
```

sam package

AWS Serverless Application Model 命令列界面 (AWS SAM CLI) 會封裝 AWS SAM 應用程式。

此命令會建立程式碼和相依性 .zip 的檔案，並將檔案上傳至 Amazon Simple Storage Service (Amazon S3)。會針對存放在 Amazon S3 中的所有檔案 AWS SAM 啟用加密。然後，它會傳回 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。

根據預設，當您使用此命令時，AWS SAM CLI 會假設您目前的工作目錄是專案的根目錄。AWS SAM CLI 第一個嘗試尋找使用 [sam build](#) 命令建置、位於 .aws-sam 子資料夾中且名為 的範本檔案 template.yaml。接著，AWS SAM CLI 會嘗試 template.yaml 在目前的工作目錄中尋找名為 template.yaml 或 的範本檔案。如果您指定 --template 選項，則會覆寫 AWS SAM CLI 的預設行為，並將僅封裝該 AWS SAM 範本及其指向的本機資源。

Note

[sam deploy](#) 現在會隱含執行 的功能 sam package。您可以直接使用 [sam deploy](#) 命令來封裝和部署您的應用程式。

Usage

```
$ sam package <arguments> <options>
```

引數

Resource ID (資源 ID)

要封裝的 Lambda 事件函數 ID。

此為選用引數。如果您的應用程式包含單一 Lambda 函數，CLI AWS SAM 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 來封裝單一函數。

有效值：資源的邏輯 ID 或資源 ARN。

選項

`--config-env` *TEXT*

在要使用的組態檔案中指定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 SAM CLI 組態檔案](#)。

`--config-file` *PATH*

組態檔案的路徑和檔案名稱，其中包含要使用的預設參數值。預設值為專案目錄根目錄中的 "samconfig.toml"。如需關於組態檔案的詳細資訊，請參閱 [AWS SAM CLI 組態檔案](#)。

`--debug`

開啟偵錯記錄以列印 AWS SAM CLI 產生的偵錯訊息，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) 儲存庫的 URI，此命令會上傳函數的映像。使用 Image 套件類型宣告的函數需要。

`--kms-key-id` *TEXT*

用來加密 Amazon S3 儲存貯體中靜態成品的 AWS Key Management Service (AWS KMS) 金鑰 ID。如果未指定此選項，則 AWS SAM 會使用 Amazon S3-managed 加密金鑰。

`--metadata`

(選用) 要連接到範本中參考之所有成品的中繼資料映射。

`--no-progressbar`

將成品上傳至 Amazon S3 時，請勿顯示進度列。

`--output-template-file` *PATH*

命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*

從您的登入資料檔案中取得 AWS 登入資料的特定設定檔。

`--region` *TEXT*

AWS 要部署的區域。例如 us-east-1。

`--resolve-s3`

自動建立用於封裝的 Amazon S3 儲存貯體。如果您同時指定 `--s3-bucket` 和 `--resolve-s3` 選項，則會產生錯誤。

`--s3-bucket` *TEXT*

此命令上傳成品的 Amazon S3 儲存貯體名稱。如果您的成品大於 51,200 個位元組，則需要 `--s3-bucket` 或 `--resolve-s3` 選項。如果您同時指定 `--s3-bucket` 和 `--resolve-s3` 選項，則會產生錯誤。

`--s3-prefix` *TEXT*

新增至上傳到 Amazon S3 儲存貯體之成品名稱的字首。字首名稱是 Amazon S3 儲存貯體的路徑名稱（資料夾名稱）。這僅適用於使用 Zip 套件類型宣告的函數。

`--save-params`

將您在命令列提供的參數儲存至 AWS SAM 組態檔案。

`--signing-profiles` *LIST*

（選用）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/值對的清單，其中索引鍵是要簽署的函數或 layer 的名稱，而值是簽署描述檔，選用的描述檔擁有者以 分隔:。例如 `FunctionNameToSign=SigningProfileName1`
`LayerNameToSign=SigningProfileName2:SigningProfileOwner`。

`--template-file`, `--template`, `-t` *PATH*

範本 AWS SAM 所在的路徑和檔案名稱。

 Note

如果您指定此選項，只會 AWS SAM 封裝範本及其指向的本機資源。

--use-json

CloudFormation 範本的輸出 JSON。預設會使用 YAML。

範例

下列範例會為 Lambda 函數和 CodeDeploy 應用程式建立並封裝成品。成品會上傳至 Amazon S3 儲存貯體。命令的輸出是名為 package.yml 的新檔案。

```
$ sam package \
  --template-file template.yml \
  --output-template-file package.yml \
  --s3-bucket amzn-s3-demo-bucket
```

sam pipeline bootstrap

此頁面提供 AWS Serverless Application Model 命令列界面 (AWS SAM CLI) sam local pipeline bootstrap 子命令的參考資訊。

如需 的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)

sam pipeline bootstrap 子命令會產生必要的 AWS 基礎設施資源，以連線至您的 CI/CD 系統。在執行 sam pipeline init 命令之前，必須針對管道中的每個部署階段執行此步驟。

此子命令會設定下列 AWS 基礎設施資源：

- 可透過下列方式設定管道許可的選項：
- 具有要與 CI/CD 系統共用之存取金鑰 ID 和私密金鑰存取憑證的管道 IAM 使用者。

Note

建議您定期輪換存取金鑰。如需詳細資訊，請參閱《IAM 使用者指南》中的[針對需要長期憑證的使用案例定期輪換存取金鑰](#)。

- 透過 OIDC 支援的 CI/CD 平台。如需搭配 AWS SAM 管道使用 OIDC 的簡介，請前往 [如何搭配 AWS SAM 管道使用 OIDC 身分驗證](#)。
- 由 擔任 CloudFormation 以部署 AWS SAM 應用程式的 CloudFormation 執行 IAM 角色。
- 保存 AWS SAM 成品的 Amazon S3 儲存貯體。

- 或者，保留容器映像 Lambda 部署套件的 Amazon ECR 映像儲存庫（如果您有套件類型 的資源Image）。

Usage

```
$ sam pipeline bootstrap <options>
```

選項

`--bitbucket-repo-uuid` *TEXT*

Bitbucket 儲存庫的 UUID。此選項專用於使用 Bitbucket OIDC 取得許可。

Note

您可以在 [https://bitbucket.org/ : //workspace/repository/admin/addon/admin/pipelines/openid-connect](https://bitbucket.org/://workspace/repository/admin/addon/admin/pipelines/openid-connect) 找到此值

`--bucket` *TEXT*

存放 AWS SAM 成品之 Amazon S3 儲存貯體的 ARN。

`--cicd-provider` *TEXT*

AWS SAM 管道的 CI/CD 平台。

`--cloudformation-execution-role` *TEXT*

部署應用程式堆疊 CloudFormation 時，要由 擔任之 IAM 角色的 ARN。只有在您想要使用自己的角色時才提供。否則，命令會建立新的角色。

`--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 SAM CLI 組態檔案](#)。

`--config-file` *PATH*

組態檔案的路徑和檔案名稱，其中包含要使用的預設參數值。預設值位於專案目錄的根samconfig.toml目錄中。如需關於組態檔案的詳細資訊，請參閱 [AWS SAM CLI 組態檔案](#)。

`--confirm-changeset | --no-confirm-changeset`

提示 確認 資源的部署。

`--create-image-repository | --no-create-image-repository`

指定是否在沒有提供的情況下建立 Amazon ECR 映像儲存庫。Amazon ECR 儲存庫會保留 Lambda 事件函數的容器映像，或套件類型為 的層Image。預設值為 `--no-create-image-repository`。

`--debug`

開啟偵錯記錄並列印 產生的偵錯訊息 AWS SAM CLI，並顯示時間戳記。

`--deployment-branch TEXT`

部署將從中發生的分支名稱。此選項專用於使用 GitHub Actions OIDC 取得許可。

`--github-org TEXT`

儲存庫所屬的 GitHub 組織。如果沒有組織，請輸入儲存庫擁有者的使用者名稱。此選項專用於使用 GitHub Actions OIDC 取得許可。

`--github-repo TEXT`

部署將從中發生的 GitHub 儲存庫名稱。此選項專用於使用 GitHub Actions OIDC 取得許可。

`--gitlab-group TEXT`

儲存庫所屬的 GitLab 群組。此選項專用於將 GitLab OIDC 用於許可。

`--gitlab-project TEXT`

GitLab 專案名稱。此選項專用於將 GitLab OIDC 用於許可。

`--help, -h`

顯示此訊息並結束。

`--image-repository TEXT`

Amazon ECR 映像儲存庫的 ARN，該儲存庫存放 Lambda 函數的容器映像，或套件類型為 的層Image。如果提供，則會忽略`--create-image-repository`選項。如果未提供 且 `--create-image-repository`已指定，則命令會建立一個。

`--interactive | --no-interactive`

停用引導參數的互動式提示，並在缺少任何必要的參數時失敗。預設值為 `--interactive`。對於此命令，`--stage`是唯一必要的參數。

 Note

如果 `--no-interactive` 與 `--use-oidc-provider` 一起指定，則必須包含 OIDC 供應商的所有必要參數。

`--oidc-client-id` *TEXT*

設定為與 OIDC 提供者搭配使用的用戶端 ID。

`--oidc-provider` [*github-actions* | *gitlab* | *bitbucket-pipelines*]

將用於 OIDC 許可的 CI/CD 提供者名稱。支援 GitLab、GitHub 和 Bitbucket。

`--oidc-provider-url` *TEXT*

OIDC 提供者的 URL。值必須以開頭 **https://**。

`--permissions-provider` [*oidc* | *iam*]

選擇許可提供者以擔任管道執行角色。預設值為 **iam**。

`--pipeline-execution-role` *TEXT*

管道使用者要擔任在此階段操作之 IAM 角色的 ARN。只有在您想要使用自己的角色時才提供。如果未提供，此命令將建立新的角色。

`--pipeline-user` *TEXT*

IAM 使用者的 Amazon Resource Name (ARN)，其存取金鑰 ID 和私密存取金鑰與 CI/CD 系統共用。它用於授予此 IAM 使用者存取對應 AWS 帳戶的許可。如果未提供，命令將建立 IAM 使用者以及存取金鑰 ID 和私密存取金鑰憑證。

`--profile` *TEXT*

從您的登入資料檔案中取得 AWS 登入資料的特定設定檔。

`--region` *TEXT*

AWS 要部署的區域。例如 **us-east-1**。

`--save-params`

將您在命令列提供的參數儲存至 AWS SAM 組態檔案。

`--stage` *TEXT*

對應部署階段的名稱。它用作所建立 AWS 基礎設施資源的尾碼。

疑難排解

錯誤：缺少必要的參數

當 `--no-interactive` 與一起指定 `--use-oidc-provider`，但未提供任何必要的參數時，此錯誤訊息將與遺失參數的描述一起顯示。

範例

下列範例會建立建立 CI/CD 系統所需的 AWS 資源，並開啟偵錯記錄，並列印產生的偵錯訊息 AWS SAM CLI：使用產生的事件進行本機測試，方法是使用 `s3.json` 事件在本機叫用 Lambda 函數

```
$ sam pipeline bootstrap --debug
```

sam pipeline init

此頁面提供 AWS Serverless Application Model 命令列界面 (AWS SAM CLI) `sam pipeline init` 子命令的參考資訊。

如需的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)

`sam pipeline init` 子命令會產生管道組態檔案，您的 CI/CD 系統可使用該檔案來部署無伺服器應用程式 AWS SAM。

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 `sam pipeline init --bootstrap` 以引導完成設定和組態檔案產生程序，或參考您先前使用 `sam pipeline bootstrap` 命令建立的資源。

Usage

```
$ sam pipeline init <options>
```

選項

`--bootstrap`

啟用互動式模式，引導使用者建立必要的 AWS 基礎設施資源。

`--config-env` *TEXT*

在要使用的組態檔案中指定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 SAM CLI 組態檔案](#)。

`--config-file` *TEXT*

組態檔案的路徑和檔案名稱，其中包含要使用的預設參數值。預設值位於 samconfig.toml 專案根目錄中。如需關於組態檔案的詳細資訊，請參閱 [AWS SAM CLI 組態檔案](#)。

`--debug`

開啟偵錯記錄以列印 產生的偵錯訊息 AWS SAM CLI，並顯示時間戳記。

`--help, -h`

顯示此訊息並結束。

`--save-params`

將您在命令列提供的參數儲存至 AWS SAM 組態檔案。

範例

下列範例說明如何使用 `--bootstrap` 選項，讓您演練互動式模式，逐步引導您建立必要的 AWS 基礎設施資源：

```
$ sam pipeline init --bootstrap
```

sam publish

此頁面提供 AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam publish` 命令的參考資訊。

如需 的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)

`sam publish` 命令會將 AWS SAM 應用程式發佈至 AWS Serverless Application Repository。此命令採用封裝 AWS SAM 範本，並將應用程式發佈至指定的 AWS 區域。

`sam publish` 命令預期 AWS SAM 範本包含包含發佈所需應用程式中繼資料的 Metadata 區段。在 Metadata 區段中，`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。如需 AWS SAM 範本 Metadata 區段的詳細資訊，請參閱 [使用 發佈您的應用程式 AWS SAM CLI](#)。

根據預設，`sam publish`會將應用程式建立為私有。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資訊，請參閱《AWS Serverless Application Repository 開發人員指南》中的以[AWS Serverless Application Repository 資源為基礎的政策範例](#)。

Note

目前`sam publish`不支援發佈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，您必須先將它們分別發佈到 `AWS Serverless Application Repository` 然後再發佈父應用程式。

Usage

```
$ sam publish <options>
```

選項

`--config-env` *TEXT*

在要使用的組態檔案中指定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 SAM CLI 組態檔案](#)。

`--config-file` *PATH*

組態檔案的路徑和檔案名稱，其中包含要使用的預設參數值。專案目錄根目錄中的預設值為「`samconfig.toml`」。如需關於組態檔案的詳細資訊，請參閱 [AWS SAM CLI 組態檔案](#)。

`--debug`

開啟偵錯記錄以列印 產生的偵錯訊息 AWS SAM CLI，並顯示時間戳記。

`--help`

顯示此訊息並結束。

`--profile` *TEXT*

從您的登入資料檔案中取得 AWS 登入資料的特定設定檔。

`--region` *TEXT*

AWS 要部署的區域。例如 `us-east-1`。

--save-params

將您在命令列提供的參數儲存至 AWS SAM 組態檔案。

--semantic-version *TEXT*

(選用) 使用此選項提供應用程式的語意版本，覆寫範本檔案 Metadata 區段中的 SemanticVersion 屬性。如需語意版本控制的詳細資訊，請參閱[語意版本控制規格](#)。

--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 callback

此頁面提供 AWS Serverless Application Model Command Line Interface (AWS SAM CLI) sam remote callback 命令的參考資訊。

如需 的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)

sam remote callback 命令可讓您將回呼傳送至在 AWS Lambda 中執行的遠端耐久函數執行。

Note

這些命令需要存取 AWS 登入資料。

Usage

```
$ sam remote callback <subcommand> <options>
```

選項

--help, -h

顯示此訊息並結束。

子命令

succeed

將回呼成功傳送至遠端耐用函數執行。如需詳細資訊，請參閱[sam remote callback succeed](#)。

fail

將回呼失敗傳送至遠端耐用函數執行。如需詳細資訊，請參閱[sam remote callback fail](#)。

heartbeat

將回呼活動訊號傳送至遠端耐用的函數執行。如需詳細資訊，請參閱[sam remote callback heartbeat](#)。

sam remote callback succeed

將回呼成功傳送至遠端耐用函數執行。

Note

此命令需要存取 AWS 登入資料。

Usage

```
$ sam remote callback succeed CALLBACK_ID [OPTIONS]
```

必要的引數

CALLBACK_ID

要傳送成功回應的回呼的唯一識別符。

選項

-r, --result TEXT

成功結果承載做為字串。

--region TEXT

設定服務的 AWS 區域 (例如 us-east-1)。

--profile TEXT

從登入資料檔案選取特定設定檔以取得 AWS 登入資料。

--config-env TEXT

在組態檔案中指定預設參數值的環境名稱。預設：default

--config-file TEXT

包含預設參數值的組態檔案。預設：samconfig.toml

--save-params

將透過命令列提供的參數儲存至組態檔案。

--beta-features / --no-beta-features

啟用/停用 Beta 版功能。

--debug

開啟偵錯記錄以列印 CLI AWS SAM 產生的偵錯訊息，並顯示時間戳記。

-h, --help

顯示此訊息並結束。

範例

傳送沒有結果的成功回呼：

```
$ sam remote callback succeed my-callback-id
```

傳送成功回呼，結果為：

```
$ sam remote callback succeed my-callback-id --result 'Task completed successfully'
```

sam remote callback fail

將回呼失敗傳送至遠端耐用函數執行。

Note

此命令需要存取 AWS 登入資料。

Usage

```
$ sam remote callback fail CALLBACK_ID [OPTIONS]
```

必要的引數

CALLBACK_ID

要傳送失敗回應的呼叫的唯一識別符。

選項

`--error-data TEXT`

其他錯誤資料。

`--stack-trace TEXT`

堆疊追蹤項目。此選項可指定多次。

`--error-type TEXT`

錯誤類型。

`--error-message TEXT`

詳細的錯誤訊息。

`--region TEXT`

設定服務的 AWS 區域 (例如 us-east-1)。

`--profile TEXT`

從登入資料檔案選取特定設定檔以取得 AWS 登入資料。

`--config-env TEXT`

在組態檔案中指定預設參數值的環境名稱。預設 : default

`--config-file TEXT`

包含預設參數值的組態檔案。預設 : samconfig.toml

`--save-params`

將透過命令列提供的參數儲存至組態檔案。

`--beta-features / --no-beta-features`

啟用/停用 Beta 版功能。

`--debug`

開啟偵錯記錄以列印 CLI AWS SAM 產生的偵錯訊息，並顯示時間戳記。

`-h, --help`

顯示此訊息並結束。

範例

傳送沒有參數的失敗回呼：

```
$ sam remote callback fail my-callback-id
```

傳送失敗回呼並顯示錯誤訊息：

```
$ sam remote callback fail my-callback-id --error-message 'Task failed'
```

使用所有參數傳送失敗回呼：

```
$ sam remote callback fail my-callback-id --error-message 'Task failed' --error-type  
'ValidationError' --stack-trace 'at line 42' --error-data '{"code": 500}'
```

sam remote callback heartbeat

將回呼活動訊號傳送至遠端耐用的函數執行。

Note

此命令需要存取 AWS 登入資料。

Usage

```
$ sam remote callback heartbeat CALLBACK_ID [OPTIONS]
```

必要的引數

CALLBACK_ID

要向其傳送活動訊號的回呼的唯一識別符。

選項

`--region TEXT`

設定服務的 AWS 區域 (例如 `us-east-1`)。

`--profile TEXT`

從登入資料檔案選取特定設定檔以取得 AWS 登入資料。

`--config-env TEXT`

在組態檔案中指定預設參數值的環境名稱。預設：`default`

`--config-file TEXT`

包含預設參數值的組態檔案。預設：`samconfig.toml`

`--save-params`

將透過命令列提供的參數儲存至組態檔案。

`--beta-features / --no-beta-features`

啟用/停用 Beta 版功能。

`--debug`

開啟偵錯記錄以列印 CLI AWS SAM 產生的偵錯訊息，並顯示時間戳記。

`-h, --help`

顯示此訊息並結束。

範例

傳送活動訊號回呼：

```
$ sam remote callback heartbeat my-callback-id
```

sam remote execution

此頁面提供 AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam remote execution` 命令的參考資訊。

如需 的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)

`sam remote execution` 命令可讓您管理和檢查在 AWS Lambda 中執行的遠端耐久函數執行。

Note

這些命令需要存取 AWS 登入資料。

Usage

```
$ sam remote execution <subcommand> <options>
```

選項

`--help`, `-h`

顯示此訊息並結束。

子命令

`get`

取得耐久執行的詳細資訊。如需詳細資訊，請參閱[sam remote execution get](#)。

`history`

取得持久函數執行的執行歷史記錄。如需詳細資訊，請參閱[sam remote execution history](#)。

`stop`

停止耐用的函數執行。如需詳細資訊，請參閱[sam remote execution stop](#)。

sam remote execution get

從 AWS Lambda 擷取特定耐用函數執行的詳細資訊。

執行 ARN 應採用叫用傳回的格式，或在執行日誌中找到。

Note

此命令需要存取 AWS 登入資料。

Usage

```
$ sam remote execution get DURABLE_EXECUTION_ARN [OPTIONS]
```

必要的引數

DURABLE_EXECUTION_ARN

要擷取詳細資訊之耐久函數執行的 Amazon Resource Name (ARN)。

選項

--format [summary|json]

輸出格式。預設：summary

--region TEXT

設定服務的 AWS 區域（例如 us-east-1）。

--profile TEXT

從登入資料檔案選取特定設定檔以取得 AWS 登入資料。

--config-env TEXT

在組態檔案中指定預設參數值的環境名稱。預設：default

--config-file TEXT

包含預設參數值的組態檔案。預設：samconfig.toml

--save-params

將透過命令列提供的參數儲存至組態檔案。

`--beta-features / --no-beta-features`

啟用/停用 Beta 版功能。

`--debug`

開啟偵錯記錄以列印 CLI AWS SAM 產生的偵錯訊息，並顯示時間戳記。

`-h, --help`

顯示此訊息並結束。

範例

取得預設摘要格式的執行詳細資訊：

```
$ sam remote execution get 'arn:aws:lambda:us-east-1:123456789012:function:my-function:
$LATEST/durable-execution/c63eec67-3415-4eb4-a495-116aa3a86278/1d454231-a3ad-3694-aa03-
c917c175db55'
```

取得 JSON 格式的執行詳細資訊：

```
$ sam remote execution get 'arn:aws:lambda:us-east-1:123456789012:function:my-function:
$LATEST/durable-execution/c63eec67-3415-4eb4-a495-116aa3a86278/1d454231-a3ad-3694-aa03-
c917c175db55' --format json
```

Note

您必須確保在使用 shell 命令時，如等執行 ARN 中的控制字元 \$ 會正確逸出。

sam remote execution history

從 AWS Lambda 擷取特定耐久函數執行的執行歷史記錄。

執行 ARN 應採用 StartDurableExecution 傳回的格式，或在執行日誌中找到。

Note

此命令需要存取 AWS 登入資料。

Usage

```
$ sam remote execution history DURABLE_EXECUTION_ARN [OPTIONS]
```

必要的引數

DURABLE_EXECUTION_ARN

要擷取其歷史記錄之耐久函數執行的 Amazon Resource Name (ARN)。

選項

--format [table|json]

輸出格式。預設：table

--region TEXT

設定服務的 AWS 區域 (例如 us-east-1)。

--profile TEXT

從登入資料檔案選取特定設定檔以取得 AWS 登入資料。

--config-env TEXT

在組態檔案中指定預設參數值的環境名稱。預設：default

--config-file TEXT

包含預設參數值的組態檔案。預設：samconfig.toml

--save-params

將透過命令列提供的參數儲存至組態檔案。

--beta-features / **--no-beta-features**

啟用/停用 Beta 版功能。

--debug

開啟偵錯記錄以列印 CLI AWS SAM 產生的偵錯訊息，並顯示時間戳記。

-h, --help

顯示此訊息並結束。

範例

使用預設資料表格式取得執行歷史記錄：

```
$ sam remote execution history arn:aws:lambda:us-east-1:123456789012:function:MyFunction:$LATEST/durable-execution/my-execution-name/my-execution-id
```

取得 JSON 格式的執行歷史記錄：

```
$ sam remote execution history arn:aws:lambda:us-east-1:123456789012:function:MyFunction:$LATEST/durable-execution/my-execution-name/my-execution-id --format json
```

sam remote execution stop

在 AWS Lambda 中停止執行中的耐用函數執行。

執行 ARN 應採用 StartDurableExecution 傳回的格式，或在執行日誌中找到。

Note

此命令需要存取 AWS 登入資料。

Usage

```
$ sam remote execution stop DURABLE_EXECUTION_ARN [OPTIONS]
```

必要的引數

DURABLE_EXECUTION_ARN

要停止之耐久函數執行的 Amazon Resource Name (ARN)。

選項

--error-message TEXT

要與停止的執行建立關聯的錯誤訊息。

`--error-type TEXT`

要與已停止執行建立關聯的錯誤類型。

`--error-data TEXT`

要與已停止執行建立關聯的錯誤資料。

`--stack-trace TEXT`

堆疊追蹤項目。此選項可指定多次。

`--region TEXT`

設定服務的 AWS 區域 (例如 `us-east-1`)。

`--profile TEXT`

從登入資料檔案選取特定設定檔以取得 AWS 登入資料。

`--config-env TEXT`

在組態檔案中指定預設參數值的環境名稱。預設：`default`

`--config-file TEXT`

包含預設參數值的組態檔案。預設：`samconfig.toml`

`--save-params`

將透過命令列提供的參數儲存至組態檔案。

`--beta-features / --no-beta-features`

啟用/停用 Beta 版功能。

`--debug`

開啟偵錯記錄以列印 CLI AWS SAM 產生的偵錯訊息，並顯示時間戳記。

`-h, --help`

顯示此訊息並結束。

範例

在沒有錯誤詳細資訊的情況下停止執行：

```
$ sam remote execution stop arn:aws:lambda:us-east-1:123456789012:function:MyFunction:
$LATEST/durable-execution/my-execution-name/my-execution-id
```

使用錯誤訊息和類型停止執行：

```
$ sam remote execution stop arn:aws:lambda:us-east-1:123456789012:function:MyFunction:
$LATEST/durable-execution/my-execution-name/my-execution-id --error-message "Execution
cancelled" --error-type "UserCancellation"
```

使用完整的錯誤詳細資訊和堆疊追蹤停止執行：

```
$ sam remote execution stop arn:aws:lambda:us-east-1:123456789012:function:MyFunction:
$LATEST/durable-execution/my-execution-name/my-execution-id --error-message "Task
failed" --error-type "TaskFailure" --error-data '{"reason":"timeout"}' --stack-trace
"at function1()" --stack-trace "at function2()"
```

sam remote invoke

此頁面提供 AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam remote invoke` 命令的參考資訊。

- 如需 的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)
- 如需使用 AWS SAM CLI `sam remote invoke` 命令的文件，請參閱 [在雲端使用 進行測試的簡介 sam remote invoke](#)。

`sam remote invoke` 命令會在 中叫用支援的資源 AWS 雲端。

Usage

```
$ sam remote invoke <arguments> <options>
```

引數

Resource ID (資源 ID)

要叫用之受支援資源的 ID。

此引數接受下列值：

- Amazon Resource Name (ARN) – 資源的 ARN。

 Tip

使用 `sam list stack-outputs --stack-name <stack-name>` 取得 資源的 ARN。

- 邏輯 ID – 資源的邏輯 ID。您還必須使用 `--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 CloudFormation。

 Tip

使用 `sam list resources --stack-name <stack-name>` 取得 資源的實體 ID。

當您提供 ARN 或實體 ID 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 CloudFormation 堆疊中的邏輯 ID 值。

當您不提供資源 ID 時：

如果您未提供資源 ID，但使用 `--stack-name` 選項提供堆疊名稱，CLI AWS SAM 將嘗試使用以下邏輯自動調用 CloudFormation 堆疊中的資源：

1. AWS SAM CLI 將依下列順序識別資源類型，並在堆疊中找到資源類型後移至下一個步驟：
 - a. Lambda
 - b. 步驟函數
 - c. Amazon SQS
 - d. Kinesis Data Streams
2. 如果資源類型在您的堆疊中有單一資源，AWS SAM CLI 會叫用它。如果您的堆疊中存在資源類型的多個資源，AWS SAM CLI 將傳回錯誤。

以下是 AWS SAM CLI 將執行的操作範例：

- 包含兩個 Lambda 事件函數和一個 Amazon SQS 佇列的堆疊 – AWS SAM CLI 將尋找 Lambda 資源類型，並傳回 和 錯誤，因為堆疊包含多個 Lambda 函數。
- 包含 Lambda 函數和兩個 Amazon Kinesis Data Streams 應用程式的堆疊 – AWS SAM CLI 將尋找 Lambda 函數並將其叫用，因為堆疊包含單一 Lambda 資源。

- 包含單一 Amazon SQS 佇列和兩個 Kinesis Data Streams 應用程式的堆疊 – AWS SAM CLI 將尋找 Amazon SQS 佇列並將其叫用，因為堆疊包含單一 Amazon SQS 佇列。

選項

`--beta-features` | `--no-beta-features`

允許或拒絕 Beta 版功能。

`--config-env` *TEXT*

從 AWS SAM CLI組態檔案指定要使用的環境。

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 [設定 AWS SAM CLI](#)。

預設：在專案目錄的samconfig.toml根目錄。

`--debug`

啟用除錯記錄。這會列印 產生的偵錯訊息和時間戳記 AWS SAM CLI。

`--event`, `-e` *TEXT*

要傳送至目標資源的事件。

`--event-file` *FILENAME*

檔案的路徑，其中包含要傳送至目標資源的事件。

`--help`, `-h`

顯示說明訊息並結束。

`--output` [*text* | *json*]

以特定輸出格式輸出調用的結果。

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 – 請求中繼資料會以文字結構傳回。資源回應會以叫用資源的輸出格式傳回。

--parameter

您可以傳遞給要叫用之資源的其他[Boto3](#)參數。

Amazon Kinesis Data Streams

下列其他參數可用來將記錄放入 Kinesis 資料串流：

- ExplicitHashKey=*'string'*
- PartitionKey=*'string'*
- SequenceNumberForOrdering=*'string'*
- StreamARN=*'string'*

如需每個參數的說明，請參閱 [Kinesis.Client.put_record](#)。

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透過回應串流叫用 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：

- 具有緩衝回應的 Lambda – [Lambda.Client.invoke](#)
- 具有回應串流的 Lambda – [Lambda.Client.invoke_with_response_stream](#)

Amazon Simple Queue Service (Amazon SQS)

下列其他參數可用來傳送訊息至 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 [SQS.Client.send_message](#)。

AWS Step Functions

下列其他參數可用來啟動狀態機器執行：

- name='string'
- traceHeader='string'

如需每個參數的說明，請參閱 [SFN.Client.start_execution](#)。

--profile *TEXT*

從您的登入資料檔案中取得 AWS 登入資料的特定設定檔。

--region *TEXT*

資源 AWS 區域 的。例如 us-east-1。

--stack-name *TEXT*

資源所屬的 CloudFormation 堆疊名稱。

--tenant-id *TEXT*

多租用戶 Lambda 函數的租用戶 ID。用來確保不同租用戶之間的運算隔離。叫用使用租戶隔離模式設定的函數時為必要。

--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。

範例

下列範例會叫用 AWS 雲端中支援的資源，並啟用偵錯記錄，以列印 產生的偵錯訊息和時間戳記 AWS SAM CLI：

```
$ sam remote invoke--debug
```

sam remote test-event

此頁面提供 AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam remote test-event` 命令的參考資訊。

- 如需 的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)
- 如需使用 AWS SAM CLIsam remote test-event命令的文件，請參閱 [使用 進行雲端測試的簡介 sam remote test-event](#)。

`sam remote test-event` 命令會與 Amazon EventBridge 結構描述登錄檔中的可共用測試事件互動。

Usage

```
$ sam remote test-event <options> <subcommand>
```

選項

`--help`, `-h`

顯示說明訊息並結束。

子命令

delete

從 EventBridge 結構描述登錄檔中刪除可共用的測試事件。如需更多參考資訊，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。如需更多參考資訊，請參閱 [sam remote test-event get](#)。

list

列出事件函數的現有可共用測試 AWS Lambda 事件。如需更多參考資訊，請參閱 [sam remote test-event list](#)。

put

將事件從本機檔案儲存至 EventBridge 結構描述登錄檔。如需更多參考資訊，請參閱 [sam remote test-event put](#)。

sam remote test-event delete

此頁面提供 AWS Serverless Application Model 命令列界面 (AWS SAM CLI) `sam remote test-event delete` 子命令的參考資訊。

- 如需的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)
- 如需使用 AWS SAM CLI `sam remote test-event` 命令的文件，請參閱 [使用 進行雲端測試的簡介 sam remote test-event](#)。

`sam remote test-event delete` 子命令會從 Amazon EventBridge 結構描述登錄檔中刪除可共用的測試事件。

Usage

```
$ sam remote test-event delete <arguments> <options>
```

引數

Resource ID (資源 ID)

與可共用測試事件相關聯的 AWS Lambda 事件函數 ID。

如果您提供邏輯 ID，您還必須使用 `--stack-name` 選項為與 Lambda 函數相關聯的 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*

在要使用的組態檔案中指定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 SAM CLI 組態檔案](#)。

`--config-file` *PATH*

組態檔案的路徑和檔案名稱，其中包含要使用的預設參數值。專案目錄根目錄中的預設值為「samconfig.toml」。如需關於組態檔案的詳細資訊，請參閱 [AWS SAM CLI 組態檔案](#)。

`--help`, `-h`

顯示說明訊息並結束。

`--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 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

此頁面提供 AWS Serverless Application Model 命令列界面 (AWS SAM CLI) `sam remote test-event get` 子命令的參考資訊。

- 如需 的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)
- 如需使用 AWS SAM CLIsam remote test-event命令的文件，請參閱 [使用 進行雲端測試的簡介 sam remote test-event](#)。

`sam remote test-event get` 子命令會從 Amazon EventBridge 結構描述登錄檔取得可共用的測試事件。

Usage

```
$ sam remote test-event get <arguments> <options>
```

引數

Resource ID (資源 ID)

與要取得的可共用測試事件相關聯的 AWS Lambda 事件函數 ID。

如果您提供邏輯 ID，您還必須使用 `--stack-name` 選項為與 Lambda 函數相關聯的 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*

在要使用的組態檔案中指定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 SAM CLI 組態檔案](#)。

`--config-file` *PATH*

組態檔案的路徑和檔案名稱，其中包含要使用的預設參數值。專案目錄根目錄中的預設值為「samconfig.toml」。如需關於組態檔案的詳細資訊，請參閱 [AWS SAM CLI 組態檔案](#)。

`--help`, `-h`

顯示說明訊息並結束。

`--name` *TEXT*

要取得的可共用測試事件名稱。

`--output-file` *FILENAME*

將事件儲存至本機電腦上的檔案路徑和名稱。

如果您不提供此選項，AWS SAM CLI 會將可共用測試事件的內容輸出到您的主控台。

`--stack-name` *TEXT*

與 Lambda 函數相關聯的 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

此頁面提供 AWS Serverless Application Model 命令列界面 (AWS SAM CLI) `sam remote test-event list` 子命令的參考資訊。

- 如需的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)
- 如需使用 AWS SAM CLI `sam remote test-event` 命令的文件，請參閱 [使用 進行雲端測試的簡介 sam remote test-event](#)。

`sam remote test-event list` 子命令會從 Amazon EventBridge 結構描述登錄檔列出特定 AWS Lambda 事件函數的現有可共用測試事件。

Usage

```
$ sam remote test-event list <arguments> <options>
```

引數

Resource ID (資源 ID)

與可共用測試事件相關聯的 Lambda 函數 ID。

如果您提供邏輯 ID，您還必須使用 `--stack-name` 選項為與 Lambda 函數相關聯的 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*

在要使用的組態檔案中指定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 SAM CLI 組態檔案](#)。

`--config-file` *PATH*

組態檔案的路徑和檔案名稱，其中包含要使用的預設參數值。專案目錄根目錄中的預設值為「samconfig.toml」。如需關於組態檔案的詳細資訊，請參閱 [AWS SAM CLI 組態檔案](#)。

`--help`, `-h`

顯示說明訊息並結束。

`--stack-name` *TEXT*

與 Lambda 函數相關聯的 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。

範例

如需使用此命令的範例，請參閱 [列出可共用的測試事件](#)。

sam remote test-event put

此頁面提供 AWS Serverless Application Model 命令列界面 (AWS SAM CLI) `sam remote test-event put` 子命令的參考資訊。

- 如需 的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)
- 如需使用 AWS SAM CLI `sam remote test-event` 命令的文件，請參閱 [使用 進行雲端測試的簡介 sam remote test-event](#)。

`sam remote test-event put` 子命令會將本機電腦的可共用測試事件儲存至 Amazon EventBridge 結構描述登錄檔。

Usage

```
$ sam remote test-event put <arguments> <options>
```

引數

Resource ID (資源 ID)

與可共用測試事件相關聯的 AWS Lambda 事件函數 ID。

如果您提供邏輯 ID，您還必須使用 `--stack-name` 選項為與 Lambda 函數相關聯的 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*

在要使用的組態檔案中指定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 SAM CLI 組態檔案](#)。

`--config-file` *PATH*

組態檔案的路徑和檔案名稱，其中包含要使用的預設參數值。專案目錄根目錄中的預設值為「samconfig.toml」。如需關於組態檔案的詳細資訊，請參閱 [AWS SAM CLI 組態檔案](#)。

`--file` *FILENAME*

事件到本機電腦上 的檔案路徑和名稱。

提供 - 做為要從 讀取的檔案名稱值stdin。

此選項為必要。

--force, -f

以相同名稱覆寫可共用的測試事件。

--help, -h

顯示說明訊息並結束。

--name *TEXT*

儲存可共用測試事件的名稱。

如果 EventBridge 結構描述登錄檔中存在同名的可共用測試事件，則 AWS SAM CLI不會覆寫它。若要覆寫，請新增 --force選項。

--output-file *FILENAME*

將事件儲存至本機電腦上的檔案路徑和名稱。

如果您不提供此選項，AWS SAM CLI會將可共用測試事件的內容輸出到您的主控台。

--stack-name *TEXT*

與 Lambda 函數相關聯的 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。

範例

如需使用此命令的範例，請參閱 [儲存可共用的測試事件](#)。

sam sync

此頁面提供 AWS Serverless Application Model Command Line Interface (AWS SAM CLI) sam sync命令的參考資訊。

- 如需 的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)
- 如需使用 的文件 AWS SAM CLI，請參閱 [AWS SAM CLI](#)。

sam sync 命令會將本機應用程式變更同步至 AWS 雲端。

Usage

```
$ sam sync <options>
```

選項

`--base-dir, -s DIRECTORY`

解決函數或 layer 的原始程式碼相對於此目錄的相對路徑。使用此選項可變更原始程式碼資料夾的相對路徑解析方式。根據預設，相對路徑會根據 AWS SAM 範本的位置進行解析。

除了您要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`

您要在建置應用程式時使用的[容器映像](#)的 URI。根據預設，AWS SAM 會使用來自 [Amazon Elastic Container Registry \(Amazon ECR\) Public](#) 的[容器映像](#)儲存庫 URI。指定此選項以使用不同的映像。

您可以在單一命令中多次使用此選項。每個選項都接受字串或鍵值對。

- 字串 – 指定應用程式中所有資源將使用的容器映像的 URI。以下是範例：

```
$ sam sync --build-image amazon/aws-sam-cli-build-image-python3.8
```

- 鍵/值對 – 將資源名稱指定為金鑰，並將要與該資源搭配使用的容器映像 URI 指定為值。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
```

此選項僅適用於指定 `--use-container` 選項時，否則將導致錯誤。

`--build-in-source | --no-build-in-source`

`--build-in-source` 可讓您直接在來源資料夾中建置專案。

--build-in-source 選項支援下列執行時間和建置方法：

- 執行時間 – [sam init --runtime](#) 選項支援的任何 Node.js 執行時間。
- 組建方法 – Makefile、esbuild。

--build-in-source 選項與下列選項不相容：

- --use-container

預設：--no-build-in-source

--capabilities *LIST*

您指定 CloudFormation 允許 建立特定堆疊的功能清單。有些堆疊範本可能包含可能影響 許可的資源 AWS 帳戶。例如，透過建立新的 AWS Identity and Access Management (IAM) 使用者。指定此選項以覆寫預設值。有效值包括下列項目：

- CAPABILITY_IAM
- CAPABILITY_NAMED_IAM
- CAPABILITY_RESOURCE_POLICY
- CAPABILITY_AUTO_EXPAND

預設：CAPABILITY_NAMED_IAM 和 CAPABILITY_AUTO_EXPAND

--code

根據預設，AWS SAM 會同步應用程式中的所有資源。指定此選項僅同步程式碼資源，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，會直接 AWS SAM 使用 AWS 服務 APIs，而不是透過 部署 AWS CloudFormation。若要更新您的 CloudFormation 堆疊，請執行 `sam sync --watch` 或 `sam deploy`。

`--config-env` *TEXT*

在要使用的組態檔案中指定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 SAM CLI 組態檔案](#)。

`--config-file` *PATH*

組態檔案的路徑和檔案名稱，其中包含要使用的預設參數值。專案目錄根目錄中的預設值為「`samconfig.toml`」。如需關於組態檔案的詳細資訊，請參閱 [AWS SAM CLI 組態檔案](#)。

`--dependency-layer` | `--no-dependency-layer`

指定是否要將個別函數的相依性分隔為另一層，以加速同步程序。

預設：`--dependency-layer`

`--express` | `--no-express`

指定是否使用 CloudFormation 快速模式進行基礎設施部署。作用中時，一旦套用資源組態，堆疊操作就會完成。資源會繼續在背景準備就緒。

快速模式僅適用於基礎設施同步路徑。僅程式碼同步 (Lambda 事件函數更新、部署) 會略過 CloudFormation，且不受此選項影響。

預設值為 `--no-express`。

 即將進行的預設變更

預設 `--express` 會在 2026 年 9 月 1 日變更為。

`--image-repository` *TEXT*

此命令上傳函數映像的 Amazon Elastic Container Registry (Amazon ECR) 儲存庫名稱。使用 Image 套件類型宣告的函數需要。

`--image-repositories` *TEXT*

函數映射至其 Amazon ECR 儲存庫 URI。依邏輯 ID 參考函數。以下是範例：

```
$ sam sync --image-repositories Function1=123456789012.dkr.ecr.us-east-1.amazonaws.com/my-repo
```

您可以在單一命令中多次指定此選項。

`--kms-key-id` *TEXT*

用來加密 Amazon S3 儲存貯體中靜態成品的 AWS Key Management Service (AWS KMS) 金鑰 ID。如果您未指定此選項，則 AWS SAM 會使用 Amazon S3-managed 加密金鑰。

`--metadata`

中繼資料映射，以連接到您在範本中參考的所有成品。

`--notification-arns` *LIST*

與堆疊建立 CloudFormation 關聯的 Amazon Simple Notification Service (Amazon SNS) 主題 ARNs 清單。

`--no-use-container`

可讓您使用 IDE 工具組來設定預設行為的選項。

`--parameter-overrides`

包含 CloudFormation 參數的字串會覆寫編碼為鍵值對的字串。使用與 AWS Command Line Interface () 相同的格式 AWS CLI。格式 AWS SAM CLI 是明確的索引鍵和值關鍵字，每個覆寫都會以空格分隔。以下是兩個範例：

- `--parameter-overrides ParameterKey=hello,ParameterValue=world`
- `--parameter-overrides ParameterKey=hello,ParameterValue=world
ParameterKey=example1,ParameterValue=example2
ParameterKey=apple,ParameterValue=banana`

`--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。選項支援 `--code` 此選項。此值必須是下列出的資源之一 `--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。選項支援 `--code` 此選項。例如 `--resource-id Function1 --resource-id Function2`。

`--role-arn` *TEXT*

套用變更集時 CloudFormation 擔任之 IAM 角色的 Amazon Resource Name (ARN)。

`--s3-bucket` *TEXT*

此命令上傳 CloudFormation 範本的 Amazon Simple Storage Service (Amazon S3) 儲存貯體名稱。如果您的範本大於 51,200 個位元組，則需要 `--s3-bucket` 或 `--resolve-s3` 選項。如果您同時指定 `--s3-bucket` 和 `--resolve-s3` 選項，則會發生錯誤。

`--s3-prefix` *TEXT*

字首會新增至您上傳至 Amazon S3 儲存貯體的成品名稱。字首名稱是 Amazon S3 儲存貯體的路徑名稱（資料夾名稱）。這僅適用於使用 Zip 套件類型宣告的函數。

`--save-params`

將您在命令列提供的參數儲存至 AWS SAM 組態檔案。

`--skip-deploy-sync` | `--no-skip-deploy-sync`

指定在不需要時 `--skip-deploy-sync` 略過初始基礎設施同步。AWS SAM CLI 將比較您的本機 AWS SAM 範本與部署的 CloudFormation 範本，並只在偵測到變更時才執行部署。

指定在每次執行 `sam sync` 時 `--no-skip-deploy-sync` 執行 CloudFormation 部署。

如需詳細資訊，請參閱 [略過初始 CloudFormation 部署](#)。

預設：`--skip-deploy-sync`

`--stack-name` *TEXT*

您應用程式的 CloudFormation 堆疊名稱。

此選項為必要。

`--tags` *LIST*

要與建立或更新的堆疊建立關聯的標籤清單。CloudFormation 也會將這些標籤傳播到堆疊中支援該標籤的資源。

`--template-file`, `--template`, `-t` *PATH*

範本 AWS SAM 所在的路徑和檔案名稱。

 Note

如果您指定此選項，則 AWS SAM 只會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相依性的套件，請使用此選項在 AWS Lambda 類似 Docker 容器內建置函數。

 Note

目前，此選項與 `--dependency-layer` 不相容。如果您 `--use-container` 搭配使用 `--dependency-layer`，AWS SAM CLI 會通知您並繼續搭配使用 `--no-dependency-layer`。

`--watch`

啟動可監控本機應用程式變更的程序，並自動將其同步至 AWS 雲端。根據預設，當您指定此選項時，會在您更新應用程式中的所有資源時 AWS SAM 同步。使用此選項，AWS SAM 會執行初始 CloudFormation 部署。然後，AWS SAM 會使用 AWS 服務 APIs 來更新程式碼資源。當您更新 AWS SAM 範本時，AWS SAM 會使用 CloudFormation 來更新基礎設施資源。

`--watch-exclude` *TEXT*

排除觀察檔案或資料夾以進行檔案變更。若要使用此選項，`--watch` 也必須提供。

此選項會收到金鑰/值對：

- 金鑰 – 應用程式中 Lambda 函數的邏輯 ID。
- 值 – 要排除的關聯檔案名稱或資料夾。

當您更新使用 `--watch-exclude` 選項指定的任何檔案或資料夾時，AWS SAM CLI 將不會啟動同步。不過，當其他檔案或資料夾的更新啟動同步時，這些檔案或資料夾會包含在該同步中。

您可以在單一命令中多次提供此選項。

範例

如需使用此命令的範例，請參閱 [sam 同步命令的選項](#)。

sam traces

此頁面提供 AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam traces` 命令的參考資訊。

如需的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)

`sam traces` 命令會在 的 中擷取 AWS X-Ray 追蹤 AWS 帳戶 AWS 區域。

Usage

```
$ sam traces <options>
```

選項

`--config-env` *TEXT*

在要使用的組態檔案中指定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 SAM CLI 組態檔案](#)。

`--config-file` *PATH*

組態檔案的路徑和檔案名稱，其中包含要使用的預設參數值。專案目錄根目錄中的預設值為「samconfig.toml」。如需關於組態檔案的詳細資訊，請參閱 [AWS SAM CLI 組態檔案](#)。

`--end-time` *TEXT*

擷取截至目前為止的追蹤。時間可以是「5 分鐘前」、「明天」等相對值，或「2018-01-01 10:10:10」等格式化時間戳記。

`--output` *TEXT*

指定日誌的輸出格式。若要列印格式化日誌，請指定 `text`。若要將日誌列印為 JSON，請指定 `json`。

`--save-params`

將您在命令列提供的參數儲存至 AWS SAM 組態檔案。

`--start-time` *TEXT*

擷取此時開始的追蹤。時間可以是「5 分鐘前」、「昨天」等相對值，或「2018-01-01 10:10:10」等格式化時間戳記。預設為 '10 分鐘前'。

`--tail`

尾隨追蹤輸出。這會忽略結束時間引數，並在追蹤可用時繼續顯示追蹤。

`--trace-id` *TEXT*

X-Ray 追蹤的唯一識別符。

範例

執行下列命令，依 ID 擷取 X-Ray 追蹤。

```
$ sam traces --trace-id tracing-id-1 --trace-id tracing-id-2
```

執行下列命令，以在 X-Ray 追蹤可用時結尾。

```
$ sam traces --tail
```

sam validate

此頁面提供 AWS Serverless Application Model Command Line Interface (AWS SAM CLI) `sam validate` 命令的參考資訊。

如需 的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)

`sam validate` 命令會驗證 AWS SAM 範本檔案是否有效。

Usage

```
$ sam validate <options>
```

選項

`--config-env` *TEXT*

在要使用的組態檔案中指定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 SAM CLI 組態檔案](#)。

`--config-file` *PATH*

組態檔案的路徑和檔案名稱，其中包含要使用的預設參數值。預設值為專案目錄根目錄中的 "samconfig.toml"。如需關於組態檔案的詳細資訊，請參閱 [AWS SAM CLI 組態檔案](#)。

`--debug`

開啟偵錯記錄以列印 AWS SAM CLI 產生的偵錯訊息，並顯示時間戳記。

`--lint`

透過 在範本上執行 linting 驗證 `cfn-lint`。建立 `cfnlintrc` 組態檔案以指定其他參數。如需詳細資訊，請參閱 CloudFormation GitHub 儲存庫中的 [cfn-lint](#)。

--profile *TEXT*

從您的登入資料檔案中取得 AWS 登入資料的特定設定檔。

--region *TEXT*

AWS 要部署的區域。例如 us-east-1。

--save-params

將您在命令列提供的參數儲存至 AWS SAM 組態檔案。

--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，且名為 `template.[yaml|yml|json]`，則不需要此選項。

如果您只執行 `sam build`，則不需要此選項。

範例

如需使用此命令驗證範本的範例，請參閱 [驗證 AWS SAM 範本檔案](#)。

如需搭配 `cfn-lint` 使用此命令的範例，請參閱 [使用 CloudFormation Linter 驗證您的 AWS SAM 應用程式](#)。

AWS SAM CLI 管理

本節包含有關如何管理和自訂 版本的資訊 AWS SAM CLI。這包括如何使用專案層級組態檔案設定 AWS SAM CLI 命令參數值的相關資訊。它也包含管理不同 版本的資訊 AWS SAM CLI、設定 AWS 登入資料，讓 AWS SAM 可以代表您呼叫 AWS 服務，以及您可以自訂的不同方式 AWS SAM。本節以一般 AWS SAM 疑難排解的區段結尾。

主題

- [AWS SAM CLI 組態檔案](#)
- [管理 AWS SAM CLI 版本](#)
- [設定 AWS 登入資料](#)
- [中的遙測 AWS SAM CLI](#)

- [AWS SAM CLI 故障診斷](#)

AWS SAM CLI 組態檔案

AWS Serverless Application Model 命令列界面 (AWS SAM CLI) 支援專案層級組態檔案，可用來設定 AWS SAM CLI 命令參數值。

如需建立和使用組態檔案的文件，請參閱 [設定 AWS SAM CLI](#)。

主題

- [預設組態檔案設定](#)
- [支援的組態檔案格式](#)
- [指定組態檔案](#)
- [組態檔案基本概念](#)
- [參數值規則](#)
- [組態優先順序](#)
- [建立和修改組態檔案](#)

預設組態檔案設定

AWS SAM 使用以下預設組態檔案設定：

- Name (名稱) – samconfig。
- 位置 – 位於專案的根目錄。這與 template.yaml 檔案的位置相同。
- 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 支援 和 [YAML|YML] 格式。請參閱下列基本語法：

TOML

```
version = 0.1
[environment]
[environment.command]
[environment.command.parameters]
```

```
option = parameter value
```

YAML

```
version: 0.1
environment:
  command:
    parameters:
      option: parameter value
```

指定組態檔案

根據預設，AWS SAM CLI會依下列順序尋找組態檔案：

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，則 AWS SAM CLI 會先尋找此檔案。
2. 預設 **samconfig.toml** 檔案 – 這是位於專案根目錄的預設組態檔案名稱和格式。如果您未指定自訂組態檔案，則 AWS SAM CLI 會接著尋找此檔案。
3. **samconfig.[yaml|yml]** 檔案 – 如果 `samconfig.toml` 不存在於專案的根目錄，則 AWS SAM CLI 會尋找此檔案。

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
```

Note

`--config-file` 參數必須相對於 AWS SAM 範本檔案的位置，因為 AWS SAM CLI 需要判斷套用組態的內容。`samconfig.toml` 檔案會管理您版本的組態設定 AWS SAM CLI，而 CLI 會在 `samconfig.toml` 檔案的相對資料夾中尋找 `template.yaml` 檔案（或覆寫組態檔案參數）。

組態檔案基本概念

Environment

環境是具名識別符，其中包含一組唯一的組態設定。您可以在單一 AWS SAM 應用程式中擁有多個環境。

預設環境名稱為 `default`。

使用 `AWS SAM CLI --config-env` 選項來指定要使用的環境。

命令

命令是 AWS SAM CLI 指定參數值的命令。

若要為所有命令指定參數值，請使用 `global` 識別符。

參考命令 AWS SAM CLI 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`

Parameters

參數指定為鍵/值對。

- 索引鍵是 AWS SAM CLI 命令選項名稱。
- 值是要指定的值。

指定金鑰時，請使用長格式的命令選項名稱，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"`。

- 對於包含鍵/值對清單的值，對是空格分隔的 ()，每個對的值由編碼的引號 () 包圍 \" \". 例如 `tags = "project=\"my-application\" stage=\"production\""`。
- 對於可多次指定的參數值，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 (") 是選用的。例如 `region: us-west-2`。這包括包含以單一字串提供的多個鍵值對的項目。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，或可在單一命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:
    parameters:
      parameter:
        - InvocationType=Event
        - LogType=None
```

組態優先順序

設定值時，會發生下列優先順序：

- 您在命令列提供的參數值優先於組態檔案和範本檔案 `Parameters` 區段中的對應值。

- 如果在命令列或具有 `parameter_overrides` 金鑰的組態檔案中使用 `--parameter-overrides` 選項，則其值優先於範本檔案 `Parameters` 區段中的值。
- 在您的組態檔案中，針對特定命令提供的項目優先於全域項目。在下列範例中，`sam deploy` 命令將使用堆疊名稱 `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
```

建立和修改組態檔案

建立組態檔案

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。

修改組態檔案

您可以手動修改您的組態檔案。此外，在任何 AWS SAM CLI 互動式流程中，設定的數值會顯示在括號 `()` 中 `[ ]`。如果您修改這些值，AWS SAM CLI 會更新您的組態檔案。

以下是使用 `sam deploy --guided` 命令的範例互動式流程：

```
$ 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，AWS SAM CLI會處理全域值，如下所示：

- 如果 參數值存在於組態檔案的 global區段中，則 AWS SAM CLI不會將值寫入特定命令區段。
- 如果 參數值同時存在於 global和特定命令區段中，則 會 AWS SAM CLI刪除特定項目，以便全域值。

管理 AWS SAM CLI版本

透過升級、降級和解除安裝來管理您的 AWS Serverless Application Model 命令列界面 (AWS SAM CLI) 版本。或者，您可以下載並安裝 AWS SAM CLI每晚組建。

主題

- [升級 AWS SAM CLI](#)
- [解除安裝 AWS SAM CLI](#)
- [從 切換Homebrew為使用 來管理 AWS SAM CLI](#)
- [管理 AWS SAM CLI每夜組建](#)
- [使用 將 AWS SAM CLI 安裝到虛擬環境中 pip](#)
- [使用 AWS SAM CLI管理 Homebrew](#)

- [疑難排解](#)

升級 AWS SAM CLI

Linux

若要升級 Linux AWS SAM CLI 上的，請遵循 中的安裝說明[安裝 AWS SAM CLI](#)，但將 `--update` 選項新增至安裝命令，如下所示：

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

macOS

必須透過用來安裝它的 AWS SAM CLI 相同方法進行升級。我們建議您使用 套件安裝程式來安裝和升級 AWS SAM CLI。

若要使用套件安裝程式升級 AWS SAM CLI，請安裝最新的套件版本。如需說明，請參閱[安裝 AWS SAM CLI](#)。

Windows

若要升級 AWS SAM CLI，請[安裝 AWS SAM CLI](#) 再次重複 中的 Windows 安裝步驟。

解除安裝 AWS SAM CLI

Linux

若要在 Linux 上解除安裝 AWS SAM CLI，您必須執行下列命令來刪除 symlink 和安裝目錄：

1. 找到符號連結並安裝路徑。

- 使用 `which` 命令尋找符號連結：

```
which sam
```

輸出會顯示 AWS SAM 二進位檔所在的路徑，例如：

```
/usr/local/bin/sam
```

- 使用 `ls` 命令尋找符號連結指向的目錄：

```
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. 刪除符號連結。

```
sudo rm /usr/local/bin/sam
```

3. 刪除安裝目錄。

```
sudo rm -rf /usr/local/aws-sam-cli
```

macOS

透過用來安裝 AWS SAM CLI 的相同方法解除安裝。我們建議您使用 套件安裝程式來安裝 AWS SAM CLI。

如果您使用套件安裝程式安裝 AWS SAM CLI，請依照下列步驟解除安裝。

若要解除安裝 AWS SAM CLI

1. 修改並執行下列動作以移除 AWS SAM CLI 程式：

```
$ sudo rm -rf /path-to/aws-sam-cli
```

- sudo** – 如果您的使用者擁有 AWS SAM CLI 程式安裝位置的寫入許可，`sudo` 則不需要。否則，`sudo` 是必要的。
- /path-to** – 您安裝 AWS SAM CLI 程式的路徑。預設位置為 `/usr/local`。

2. 透過修改並執行以下項目 \$PATH 來 AWS SAM CLI 移除：

```
$ sudo rm -rf /path-to-symlink-directory/sam
```

- sudo** – 如果您的使用者具有的寫入許可 \$PATH，`sudo` 則不需要。否則，`sudo` 是必要的。
- path-to-symlink-directory** – 您的 \$PATH 環境變數。預設位置為 `/usr/local/bin`。

3. 執行下列動作，確認已解除安裝 AWS SAM CLI：

```
$ sam --version
command not found: sam
```

Windows

若要使用 Windows 設定解除安裝 AWS SAM CLI，請遵循下列步驟：

1. 從開始功能表中，搜尋「新增或移除程式」。
2. 選擇名為AWS SAM 命令列界面的結果，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 SAM CLI。

從 切換Homebrew為使用 來管理 AWS SAM CLI

如果您使用 Homebrew 安裝和升級 AWS SAM CLI，建議您使用 AWS 支援的 方法。請依照這些指示 切換到支援的 方法。

使用 從 切換 Homebrew

1. 請依照 的指示[解除安裝Homebrew已安裝的 AWS SAM CLI](#)解除安裝 Homebrew受管版本。
2. 遵循 中的指示[安裝 AWS SAM CLI](#)，使用支援的方法安裝 AWS SAM CLI。

管理 AWS SAM CLI每夜組建

您可以下載並安裝 AWS SAM CLI每晚組建。它包含程式碼的 AWS SAM CLI發行前版本，可能比生產版本不穩定。安裝後，您可以搭配 `sam-nightly` 命令使用夜間組建。您可以同時安裝和使用 AWS SAM CLI的生產版本和夜間建置版本。

Note

每夜組建不包含組建映像的發行前版本。因此，使用 `--use-container` 選項建置您的無伺服器應用程式會使用建置映像的最新生產版本。

安裝 AWS SAM CLI每晚組建

若要安裝 AWS SAM CLI每晚組建，請遵循下列指示。

Linux

您可以使用 套件安裝程式，在 Linux x86_64 平台上安裝 的夜間建置版本 AWS SAM CLI。

安裝 AWS SAM CLI 每晚組建

1. 在 aws-sam-cli GitHub sam-cli 儲存庫中，從 [sam-cli-nightly](#) 下載套件安裝程式。
2. 請依照 [安裝 AWS SAM CLI](#) 的步驟，安裝夜間建置套件。

macOS

您可以使用夜間建置套件安裝程式 macOS，在 上安裝 的夜間建置版本 AWS SAM CLI。

安裝 AWS SAM CLI 每晚組建

1. 從 aws-sam-cli 儲存庫中的 [sam-cli-nightly](#) 下載您平台的套件安裝程式。aws-sam-cli GitHub
2. 請依照 [安裝 AWS SAM CLI](#) 的步驟，安裝夜間建置套件。

Windows

此下載連結提供 的 AWS SAM CLI 每夜建置版本：[AWS SAM CLI 每夜建置](#)。若要在 Windows 上安裝 每夜組建，請執行與 相同的步驟 [安裝 AWS SAM CLI](#)，但改用每夜組建下載連結。

若要確認您已安裝每晚建置版本，請執行 `sam-nightly --version` 命令。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

如果您使用 Homebrew 來安裝和升級 AWS SAM CLI 每夜組建，並想要使用套件安裝程式切換到 ，請遵循以下步驟。

從 切換 Homebrew 到 套件安裝程式

1. 解除安裝 Homebrew 已安裝 AWS SAM CLI 的夜間組建。

```
$ brew uninstall aws-sam-cli-nightly
```

2. 執行下列動作，AWS SAM CLI 確認已解除安裝每夜組建：

```
$ sam-nightly --version
zsh: command not found: sam-nightly
```

3. 請依照上一節中的步驟安裝 AWS SAM CLI 每夜組建。

使用 將 AWS SAM CLI 安裝到虛擬環境中 pip

我們建議您使用原生套件安裝程式來安裝 AWS SAM CLI。如果您必須使用 pip，我們建議您將 AWS SAM CLI 安裝到虛擬環境中。這可確保乾淨的安裝環境，並在發生錯誤時隔離環境。

Note

截至 2023 AWS SAM CLI 年 10 月 24 日，不再支援 Python 3.7。如需詳細資訊，請參閱 [AWS SAM CLI 停止對的支援 Python 3.7。](#)

將 AWS SAM CLI 安裝到虛擬環境

1. 從您選擇的起始目錄中，建立虛擬環境並將其命名。

Linux / macOS

```
$ mkdir project
$ cd project
$ python3 -m venv venv
```

Windows

```
> mkdir project
> cd project
> py -3 -m venv venv
```

2. 啟用虛擬環境

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 AWS SAM CLI 安裝到您的虛擬環境。

```
(venv) $ pip install --upgrade aws-sam-cli
```

4. 確認 AWS SAM CLI 已正確安裝。

```
(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。

使用 AWS SAM CLI 管理 Homebrew

Note

從 2023 年 9 月開始，AWS 將不再維護 () AWS SAM CLI 的 AWS 受管 Homebrew 安裝程式 `aws/tap/aws-sam-cli`。若要繼續使用 Homebrew，您可以使用 社群受管安裝程式 (`aws-sam-cli`)。從 2023 年 9 月開始，任何參考的 Homebrew 命令 `aws/tap/aws-sam-cli` 都會重新導向至 `aws-sam-cli`。

我們建議您使用我們支援的 [安裝](#) 和 [升級](#) 方法。

使用 安裝 AWS SAM CLI Homebrew

Note

這些指示使用社群受 AWS SAM CLI 管 Homebrew 安裝程式。如需進一步支援，請參閱 [homebrew-core](#) 儲存庫。

安裝 AWS SAM CLI

1. 執行下列命令：

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

2. 驗證安裝：

```
$ sam --version
```

成功安裝 後 AWS SAM CLI，您應該會看到如下所示的輸出：

```
SAM CLI, version 1.94.0
```

使用 升級 AWS SAM CLI Homebrew

若要使用 升級 AWS SAM CLI Homebrew，請執行下列命令：

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

解除安裝 Homebrew 已安裝的 AWS SAM CLI

如果 AWS SAM CLI 使用 安裝 Homebrew，請依照下列步驟解除安裝它。

解除安裝 AWS SAM CLI

1. 執行下列命令：

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

2. 執行下列動作，確認已解除安裝 AWS SAM CLI：


```
$ sam --version  
command not found: sam
```

切換到社群受管Homebrew安裝程式

如果您使用的是 AWS 受管Homebrew安裝程式 (aws/tap/aws-sam-cli)，而且偏好繼續使用 Homebrew，建議您切換到 社群受管Homebrew安裝程式 (aws-sam-cli)。

若要在單一命令中切換，請執行下列動作：

```
$ brew uninstall aws-sam-cli && brew untap aws/tap && brew cleanup aws/tap && brew  
update && brew install aws-sam-cli
```

請依照這些指示個別執行每個命令。

切換到社群受管Homebrew安裝程式

1. 解除安裝的 AWS 受管Homebrew版本 AWS SAM CLI：

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

2. 確認 AWS SAM CLI 已解除安裝：

```
$ which sam  
sam not found
```

3. 移除 AWS 受管 AWS SAM CLI點選：

```
$ brew untap aws/tap
```

如果您收到類似以下的錯誤，請新增 `--force` 選項，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 的社群受管版本 AWS SAM CLI：

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

7. 確認已成功安裝 AWS SAM CLI：

```
$ sam --version  
SAM CLI, version 1.94.0
```

疑難排解

如果您在安裝或使用時遇到錯誤 AWS SAM CLI，請參閱 [AWS SAM CLI 故障診斷](#)。

設定 AWS 登入資料

AWS SAM 命令列界面 (CLI) 需要您設定 AWS 登入資料，以便可以代表您呼叫 AWS 服務。例如，AWS SAM CLI會呼叫 Amazon S3 和 CloudFormation。

您可能已經設定 AWS 登入資料來使用 AWS 工具，例如其中一個 AWS SDKs或 AWS CLI。如果您尚未這麼做，本主題會顯示設定 AWS 登入資料的建議方法。

若要設定 AWS 登入資料，您必須擁有要設定的 IAM 使用者的存取金鑰 ID 和私密存取金鑰。如需有關存取金鑰 IDs和私密存取金鑰的資訊，請參閱 [《IAM 使用者指南》中的管理 IAM 使用者的存取金鑰](#)。

接著，判斷您是否 AWS CLI 已安裝。然後遵循下列其中一個區段中的指示：

使用 AWS CLI

如果您 AWS CLI 已安裝，請使用 `aws configure`命令並依照提示操作：

```
$ 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]:
```

如需 `aws configure` 命令的相關資訊，請參閱 AWS Command Line Interface 《使用者指南》中的 [快速設定 AWS CLI](#)。

不使用 AWS CLI

如果您沒有 AWS CLI 安裝，您可以建立登入資料檔案或設定環境變數：

- 登入資料檔案 – 您可以在本機系統的登入資料檔案中設定 AWS 登入資料。此檔案必須位於下列其中一個位置：
 - `~/.aws/credentials` Linux 或 macOS 上的
 - 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，請使用匯出命令：

```
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` 命令：

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

中的遙測 AWS SAM CLI

在 AWS，我們根據與客戶互動的經驗來開發和啟動服務。我們使用客戶意見回饋來反覆查看我們的產品。遙測是額外資訊，有助於我們進一步了解客戶的需求、診斷問題，以及提供可改善客戶體驗的功能。

AWS SAM 命令列界面 (CLI) 會收集遙測，例如一般用量指標、系統和環境資訊和錯誤。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 SAM CLI 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。

客戶會控制是否開啟遙測，也可以隨時變更其設定。如果遙測保持開啟，AWS SAM CLI 會在背景傳送遙測資料，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

當您在作業系統上執行 AWS SAM CLI 時，請執行下列命令來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

您可以使用下列命令，在終端機視窗的生命週期內暫時設定環境變數：

如果使用命令提示字元：

```
set SAM_CLI_TELEMETRY=0
```

如果使用 PowerShell：

```
$env:SAM_CLI_TELEMETRY=0
```

若要在 Command Prompt 或 PowerShell 中永久設定環境變數，請使用下列命令：

```
setx SAM_CLI_TELEMETRY 0
```

Note

在終端機關閉並重新開啟之前，變更不會生效。

收集的資訊類型

- 用量資訊 – 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、內部例外名稱，以及連線至 Docker 時失敗。
- 系統和環境資訊 – Python 版本、作業系統 (Windows、Linux 或 macOS)、執行環境 AWS SAM CLI (例如 AWS CodeBuild IDE AWS 工具組或終端機)，以及用量屬性的雜湊值。

進一步了解

收集的 AWS SAM CLI 遙測資料符合 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](#)
- [資料隱私權常見問答集](#)

AWS SAM CLI 故障診斷

本節提供有關如何在使用、安裝和管理 AWS Serverless Application Model 命令列界面 () 時疑難排解錯誤訊息的詳細資訊AWS SAM CLI。

主題

- [疑難排解](#)
- [錯誤訊息](#)
- [警告訊息](#)

疑難排解

如需與 相關的疑難排解指引 AWS SAM CLI，請參閱 [對安裝錯誤進行故障診斷](#)。

錯誤訊息

Curl 錯誤：「curl：(6) 無法解析：...」

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)
```

這表示您嘗試將請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curl命令中有錯別字，就會發生這種情況。使用 CloudFormation 主控台或 驗證應用程式是否已成功部署 AWS CLI，並驗證您的curl命令是否正確。

錯誤：找不到具有指定堆疊名稱的確切資源資訊

在包含單一 Lambda 事件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.
```

可能原因：您未提供 `--stack-name`選項。

如果未提供函數 ARN 做為引數，則`sam remote invoke`命令需要提供 `--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：找不到堆疊名稱中的資源資訊

執行 `sam remote invoke` 命令並以引數形式傳遞 Lambda 函數 ARN 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>)
```

可能原因：您在 **samconfig.toml** 檔案中已定義堆疊名稱值。

AWS SAM CLI 先檢查您的 `samconfig.toml` 檔案是否有堆疊名稱。如果指定，引數會以邏輯 ID 值傳遞。

解決方案：改為傳遞函數的邏輯 ID。

您可以傳遞函數的邏輯 ID 做為引數，而不是函數的 ARN。

解決方案：從您的組態檔案移除堆疊名稱值。

您可以從組態檔案移除堆疊名稱值。這可防止 AWS SAM CLI 將函數 ARN 做為邏輯 ID 值傳遞。

修改組態檔案 `sam build` 後執行。

錯誤：無法建立受管資源：找不到登入資料

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 登入資料，讓 AWS SAM CLI 能夠進行 AWS 服務呼叫。若要修正此問題，您必須設定 AWS 登入資料。如需詳細資訊，請參閱[設定 AWS 登入資料](#)。

錯誤：Windows 中的 FileNotFoundError

在 Windows 上執行命令 AWS SAM CLI 時，您可能會看到下列錯誤：

```
Error: FileNotFoundError
```

可能原因：AWS SAM CLI 可能會與超過 Windows 路徑上限的檔案路徑互動。

解決方案：若要解決此問題，必須啟用新的長路徑行為。若要這樣做，請參閱 Microsoft [Windows 應用程式開發文件](#)中的在 Windows 10 版本 1607 和更新版本中啟用長路徑。

錯誤：pip 的相依性解析程式...

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.
```

可能原因：如果您使用 pip 來安裝套件，套件之間的相依性可能會衝突。

套件的每個版本 aws-sam-cli 都取決於 aws-sam-translator 套件的版本。例如，aws-sam-cliv1.58.0 可能取決於 aws-sam-translator v1.51.0。

如果您使用 安裝 AWS SAM CLI pip，然後安裝另一個取決於較新版本 的套件 aws-sam-translator，則會發生下列情況：

- aws-sam-translator 將安裝較新版本的。
- 目前版本的 aws-sam-cli 和較新版本的 aws-sam-translator 可能不相容。
- 當您使用時 AWS SAM CLI，會發生相依性解析程式錯誤。

解決方案：

1. 使用 AWS SAM CLI 原生套件安裝程式。
 - a. 使用 pip 解除安裝 AWS SAM CLI。如需說明，請參閱[解除安裝 AWS SAM CLI](#)。
 - b. 使用原生套件安裝程式安裝 AWS SAM CLI。如需說明，請參閱[安裝 AWS SAM CLI](#)。
 - c. 如有必要，請使用原生套件安裝程式升級 AWS SAM CLI。如需說明，請參閱[升級 AWS SAM CLI](#)。
2. 如果您必須使用 pip，我們建議您將 AWS SAM CLI 安裝到虛擬環境中。這可確保安裝環境乾淨，並在發生錯誤時隔離環境。如需說明，請參閱[使用 將 AWS SAM CLI 安裝到虛擬環境中 pip](#)。

錯誤：沒有這類命令「遠端」

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'.
```

可能原因：您的 AWS SAM CLI 版本已過期。

此 AWS SAM CLI `sam remote invoke` 命令已發行 AWS SAM CLI 1.88.0 版。您可以執行 `sam --version` 命令來檢查版本。

解決方案：將您的 AWS SAM CLI 升級至最新版本。

如需說明，請參閱[升級 AWS SAM CLI](#)。

錯誤：在本機執行 AWS SAM 專案需要 Docker。是否已安裝？

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?
```

這表示您尚未 Docker 正確安裝。Docker 需要在本機測試您的應用程式。若要修正此問題，請遵循為您的開發主機安裝 Docker 的指示。如需詳細資訊，請參閱[安裝 Docker](#)。

錯誤：未滿足安全限制

執行 `sam deploy --guided`，系統會提示您輸入問題 *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，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，無需授權。透過**N**回應此提示，您表示這不行。

若要修正此問題，您有下列選項：

- 使用授權設定您的應用程式。如需設定授權的資訊，請參閱 [使用 AWS SAM 範本控制 API 存取](#)。
- 如果您的目的是在未經授權的情況下擁有可公開存取的 API 端點，請重新啟動您的部署，並使用 回應此問題**Y**，以指出您可以進行部署。

訊息：缺少身分驗證字符

嘗試叫用 API Gateway 端點時，您會看到下列錯誤：

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

這表示您嘗試將請求傳送至正確的網域，但無法辨識 URI。若要修正此問題，請驗證完整的 URL，並使用正確的 URL 更新curl命令。

警告訊息

警告：... AWS 將不再維護 的Homebrew安裝程式 AWS SAM ...

使用 安裝 AWS SAM CLI 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支援。

從 2023 年 9 月開始，AWS 將不再維護 的Homebrew安裝程式 AWS SAM CLI。

解決方案：使用 AWS 支援的安裝方法。

- 您可以在 [找到 AWS 支援的安裝方法](#) [安裝 AWS SAM CLI](#)。

解決方案：若要繼續使用 Homebrew，請使用 社群受管安裝程式。

- 您可以自行決定使用社群受管Homebrew安裝程式。如需說明，請參閱[使用 AWS SAM CLI管理 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 連接器管理資源許可](#)。

連接器支援的來源和目的地資源類型

`AWS::Serverless::Connector` 資源類型支援來源和目的地資源之間的選取連線數。在 AWS SAM 範本中設定連接器時，請使用下表來參考支援的連線，以及需要為每個來源和目的地資源類型定義的屬性。如需在範本中設定連接器的詳細資訊，請參閱 [AWS::Serverless::Connector](#)。

對於來源和目的地資源，當在相同範本中定義時，請使用 `Id` 屬性。或者，`Qualifier`您可以新增 來縮小已定義資源的範圍。當資源不在相同的範本內時，請使用支援的屬性組合。

若要請求新的連線，請在 `serverless-application-model` AWS GitHub儲存庫[提交新問題](#)。

Source type (來源類型)	目的地類型	許可	來源屬性	目的地屬性
<code>AWS::ApiGateway::RestApi</code>	<code>AWS::Lambda::Function</code>	Write	Id 或 ResourceId、Qualifier 和 Type	Id 或 Arn和 Type
<code>AWS::ApiGateway::RestApi</code>	<code>AWS::Serverless::Function</code>	Write	Id 或 ResourceId、Qualifier 和 Type	Id 或 Arn和 Type

Source type (來源類型)	目的地類型	許可	來源屬性	目的地屬性
AWS::ApiGatewayV2::Api	AWS::Lambda::Function	Write	Id 或 ResourceId 、Qualifier 和 Type	Id 或 Arn和 Type
AWS::ApiGatewayV2::Api	AWS::Serverless::Function	Write	Id 或 ResourceId 、Qualifier 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
AWS::AppSync::DataSource	AWS::Serverless::SimpleTable	Read	Id 或 RoleName和 Type	Id 或 Arn和 Type

Source 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	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::Event Bus	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

Source 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 QueueUrl、Arn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

Source 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::S3::Bucket	Read, Write	Id 或 RoleName和 Type	Id 或 Arn和 Type
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 Name、Arn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 Name、Arn和 Type
AWS::S3::Bucket	AWS::Lambda::Function	Write	Id 或 Arn和 Type	Id 或 Arn和 Type

Source type (來源類型)	目的地類型	許可	來源屬性	目的地屬性
AWS::S3::Bucket	AWS::Serverless::Function	Write	Id 或 Arn和 Type	Id 或 Arn和 Type
AWS::Serverless::Api	AWS::Lambda::Function	Write	Id 或 ResourceId 、Qualifier 和 Type	Id 或 Arn和 Type
AWS::Serverless::Api	AWS::Serverless::Function	Write	Id 或 ResourceId 、Qualifier 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::Event Bus	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

Source 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::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 Name、Arn和 Type
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 Name、Arn和 Type
AWS::Serverless::HttpApi	AWS::Lambda::Function	Write	Id 或 ResourceId、Qualifier 和 Type	Id 或 Arn和 Type
AWS::Serverless::HttpApi	AWS::Serverless::Function	Write	Id 或 ResourceId、Qualifier 和 Type	Id 或 Arn和 Type
AWS::Serverless::SimpleTable	AWS::Lambda::Function	Read	Id 或 Arn和 Type	Id 或 RoleName和 Type

Source type (來源類型)	目的地類型	許可	來源屬性	目的地屬性
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
AWS::Serverless::StateMachine	AWS::Events::Event Bus	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

Source 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::Serverless::StateMachine	Read, Write	Id 或 RoleName和 Type	Id 或 Name、Arn和 Type
AWS::Serverless::StateMachine	AWS::SNS::Topic	Write	Id 或 RoleName和 Type	Id 或 Arn和 Type
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 Name、Arn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 QueueUrl、Arn和 Type
AWS::SQS::Queue	AWS::Lambda::Function	Read, Write	Id 或 Arn和 Type	Id 或 RoleName和 Type

Source type (來源類型)	目的地類型	許可	來源屬性	目的地屬性
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
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

Source type (來源類型)	目的地類型	許可	來源屬性	目的地屬性
AWS::Step Functions::StateMachine	AWS::Serverless::StateMachine	Read, Write	Id 或 RoleName和 Type	Id 或 Name、Arn和 Type
AWS::Step Functions::StateMachine	AWS::SNS::Topic	Write	Id 或 RoleName和 Type	Id 或 Arn和 Type
AWS::Step Functions::StateMachine	AWS::SQS::Queue	Write	Id 或 RoleName和 Type	Id 或 Arn和 Type
AWS::Step Functions::StateMachine	AWS::Step Functions::StateMachine	Read, Write	Id 或 RoleName和 Type	Id 或 Name、Arn和 Type

連接器建立的 IAM 政策

本節記錄使用連接器 AWS SAM 時由 建立的 AWS Identity and Access Management (IAM) 政策。

AWS::DynamoDB::Table 至 AWS::Lambda::Function

政策類型

連接至AWS::Lambda::Function角色[的客戶受管政策](#)。

存取類別

Read

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

```
{
  "Effect": "Allow",
  "Action": [
    "dynamodb:DescribeStream",
    "dynamodb:GetRecords",
    "dynamodb:GetShardIterator",
    "dynamodb:ListStreams"
  ],
  "Resource": [
    "%{Source.Arn}/stream/*"
  ]
}
```

AWS::Events::Rule 至 AWS::SNS::Topic

政策類型

[AWS::SNS::TopicPolicy](#) 連接到 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 至 AWS::Events::EventBus

政策類型

連接至AWS::Events::Rule角色[的客戶受管政策](#)。

存取類別

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:PutEvents"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

AWS::Events::Rule 至 AWS::StepFunctions::StateMachine

政策類型

連接至AWS::Events::Rule角色[的客戶受管政策](#)。

存取類別

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:StartExecution"
      ],
      "Resource": [
```

```
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

AWS::Events::Rule 至 AWS::Lambda::Function

政策類型

[AWS::Lambda::Permission](#) 連接到 AWS::Lambda::Function。

存取類別

Write

```
{
  "Action": "lambda:InvokeFunction",
  "Principal": "events.amazonaws.com",
  "SourceArn": "%{Source.Arn}"
}
```

AWS::Events::Rule 至 AWS::SQS::Queue

政策類型

[AWS::SQS::QueuePolicy](#) 連接到 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 至 AWS::Lambda::Function

政策類型

連接至AWS::Lambda::Function角色[的客戶受管政策](#)。

存取類別

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeAsync",
        "lambda:InvokeFunction"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

AWS::Lambda::Function 至 AWS::S3::Bucket

政策類型

連接至AWS::Lambda::Function角色[的客戶受管政策](#)。

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 至 AWS::DynamoDB::Table

政策類型

連接至AWS::Lambda::Function角色[的客戶受管政策](#)。

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 至 AWS::SQS::Queue

政策類型

連接至AWS::Lambda::Function角色[的客戶受管政策](#)。

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 至 AWS::SNS::Topic

政策類型

連接至AWS::Lambda::Function角色[的客戶受管政策](#)。

存取類別

Write

```
{  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "sns:Publish"  
      ],  
      "Resource": [  
        "%{Destination.Arn}"  
      ]  
    }  
  ]  
}
```

AWS::Lambda::Function 至 AWS::StepFunctions::StateMachine

政策類型

連接至AWS::Lambda::Function角色[的客戶受管政策](#)。

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 至 AWS::Events::EventBus

政策類型

連接至AWS::Lambda::Function角色[的客戶受管政策](#)。

存取類別

Write

```

{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:PutEvents"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}

```

AWS::Lambda::Function 至 AWS::Location::PlaceIndex

政策類型

連接至AWS::Lambda::Function角色[的客戶受管政策](#)。

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 至 AWS::Lambda::Function

政策類型

[AWS::Lambda::Permission](#) 連接到 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 至 AWS::Lambda::Function

政策類型

[AWS::Lambda::Permission](#) 連接到 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 至 AWS::SQS::Queue

政策類型

[AWS::SQS::QueuePolicy](#) 連接到 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 至 AWS::Lambda::Function

政策類型

[AWS::Lambda::Permission](#) 連接到 AWS::Lambda::Function。

存取類別

Write

```
{
  "Action": "lambda:InvokeFunction",
  "Principal": "sns.amazonaws.com",
  "SourceArn": "%{Source.Arn}"
}
```

AWS::SQS::Queue 至 AWS::Lambda::Function

政策類型

連接至AWS::Lambda::Function角色[的客戶受管政策](#)。

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 至 AWS::Lambda::Function

政策類型

[AWS::Lambda::Permission](#) 連接到 AWS::Lambda::Function。

存取類別

Write

```
{
  "Action": "lambda:InvokeFunction",
  "Principal": "s3.amazonaws.com",
  "SourceArn": "%{Source.Arn}",
  "SourceAccount": "${AWS::AccountId}"
}
```

AWS::StepFunctions::StateMachine 至 AWS::Lambda::Function

政策類型

連接至AWS::StepFunctions::StateMachine角色[的客戶受管政策](#)。

存取類別

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeAsync",
        "lambda:InvokeFunction"
      ],
```

```
    "Resource": [
      "%{Destination.Arn}"
    ]
  }
]
```

AWS::StepFunctions::StateMachine 至 AWS::SNS::Topic

政策類型

連接至AWS::StepFunctions::StateMachine角色[的客戶受管政策](#)。

存取類別

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

AWS::StepFunctions::StateMachine 至 AWS::SQS::Queue

政策類型

連接至AWS::StepFunctions::StateMachine角色[的客戶受管政策](#)。

存取類別

Write

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

```

    "Effect": "Allow",
    "Action": [
        "sqs:SendMessage"
    ],
    "Resource": [
        "%{Destination.Arn}"
    ]
}
]
}

```

AWS::StepFunctions::StateMachine 至 AWS::S3::Bucket

政策類型

連接至AWS::StepFunctions::StateMachine角色[的客戶受管政策](#)。

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 至 AWS::DynamoDB::Table

政策類型

連接至AWS::StepFunctions::StateMachine角色[的客戶受管政策](#)。

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 至 AWS::StepFunctions::StateMachine

政策類型

連接至AWS::StepFunctions::StateMachine角色[的客戶受管政策](#)。

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 至 AWS::Events::EventBus

政策類型

連接至AWS::StepFunctions::StateMachine角色[的客戶受管政策](#)。

存取類別

Write

```

{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "events:PutEvents"
            ],
            "Resource": [
                "%{Destination.Arn}"
            ]
        }
    ]
}

```

AWS::AppSync::DataSource 至 AWS::DynamoDB::Table

政策類型

連接至AWS::AppSync::DataSource角色[的客戶受管政策](#)。

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 至 AWS::Lambda::Function

政策類型

連接至AWS::AppSync::DataSource角色[的客戶受管政策](#)。

存取類別

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeAsync",
        "lambda:InvokeFunction"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}:*"
      ]
    }
  ]
}
```

AWS::AppSync::DataSource 至 AWS::Events::EventBus

政策類型

連接至AWS::AppSync::DataSource角色[的客戶受管政策](#)。

存取類別

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:PutEvents"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

AWS::AppSync::GraphQLApi 至 AWS::Lambda::Function

政策類型

[AWS::Lambda::Permission](#) 連接到 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 Docker 以搭配 使用 AWS SAM CLI

Docker 是在您的機器上執行容器的應用程式。透過 Docker，AWS SAM 可以提供類似於 AWS Lambda 容器的本機環境，以建置、測試和偵錯無伺服器應用程式。

Note

Docker 只有在本機測試您的應用程式，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

請依照這些指示，在您的作業系統Docker上安裝。

Linux

Docker 可在許多不同的作業系統上使用，包括最現代化的 Linux 發行版本，例如 CentOS、Debian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](#)。

在 Amazon Linux 2 或 Amazon Linux 2023 Docker上安裝

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 Community Edition 套件。

- 針對 Amazon Linux 2，執行下列動作：

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

- 對於 Amazon Linux 2023，請執行下列動作：

```
$ sudo yum install -y docker
```

3. 啟動 Docker 服務。

```
$ sudo service docker start
```

4. 將 ec2-user新增至 docker群組，讓您無需使用 即可執行Docker命令sudo。

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

5. 登出並重新登入，以挑選新的docker群組許可。若要執行此作業，請關閉目前的 SSH 終端機視窗，然後重新連線至新的執行個體。您的新 SSH 工作階段應具有適當的docker群組許可。
6. 確認 ec2-user可以在不使用 的情況下執行 Docker 命令sudo。

```
$ docker ps
```

您應該會看到下列輸出，確認已安裝並執行 Docker：

CONTAINER ID STATUS	IMAGE PORTS	COMMAND NAMES	CREATED
------------------------	----------------	------------------	---------

Note

在 Linux 上，若要使用與主機機器不同的指令集架構來建置和執行 Lambda 事件函數，還有其他步驟可設定 Docker。例如，若要在 x86_64 機器上執行 arm64 函數，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 [對安裝錯誤進行故障診斷](#)。或者，請參閱 Docker Docs 網站上的 Linux 安裝後步驟 [疑難排解](#) 一節。

macOS

Note

Docker 桌面已正式支援，但從 AWS SAM CLI 1.47.0 版開始，只要它們使用 Docker 執行時間，您就可以使用替代方案。

1. 安裝 Docker

AWS SAM CLI 支援在 macOS Sierra 10.12 或更新版本 Docker 上執行。如需如何安裝 Docker，請參閱 Docker Docs 網站上的 [Install Docker Desktop for Mac](#)。

2. 設定共用磁碟機

AWS SAM CLI 需要將專案目錄或任何父目錄列在共用磁碟機中。這可能需要檔案共用。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 之後，請確認它是否正常運作。同時確認您可以從 Docker 命令列執行命令（例如 `docker ps`）。您不需要安裝、擷取或提取任何容器，AWS SAM CLI 會視需要自動執行此操作。

如果您在安裝時遇到問題 Docker，如需更多故障診斷秘訣，請參閱 Docker 文件網站的[故障診斷一節](#)。

Windows

Note

AWS SAM 正式支援 Docker Desktop。不過，從 AWS SAM CLI 1.47.0 版開始，只要替代方案使用 Docker 執行時間，您就可以使用它們。

1. 安裝 Docker。

Docker 桌面支援最新的 Windows 作業系統。對於舊版 Windows，可使用 Docker 工具箱。選擇您的 Windows 版本，以取得正確的 Docker 安裝步驟：

- 若要安裝 Docker for Windows 10，請參閱 Docker Docs 網站上的 [Install Docker Desktop for Windows](#)。
- 若要 Docker 為舊版 Windows 安裝，請參閱 [Docker Toolbox](#) GitHub 儲存庫上的 Docker 工具箱。

2. 設定您的共用磁碟機。

AWS SAM CLI 需要將專案目錄或任何父目錄列在共用磁碟機中。在某些情況下，您必須共用您的磁碟機，Docker 才能正常運作。

3. 驗證安裝。

安裝 Docker 之後，請確認它是否正常運作。同時確認您可以從 Docker 命令列執行命令（例如 `docker ps`）。您不需要安裝、擷取或提取任何容器，AWS SAM CLI 會視需要自動執行此操作。

如果您在安裝時遇到問題 Docker，如需更多故障診斷秘訣，請參閱 Docker 文件網站的[故障診斷一節](#)。

後續步驟

如需如何安裝 AWS SAM CLI，請參閱 [安裝 AWS SAM CLI](#)。

安裝 Finch 以搭配 CLI AWS SAM 使用

Finch 是 AWS 的容器開發工具，提供的替代方案 Docker，特別有益於 macOS 和 Linux 系統。如需詳細資訊，請參閱 [GitHub 上的 Finch](#)。透過 Finch，AWS SAM 可以提供類似於 AWS Lambda 容器的本機環境，以建置、測試和偵錯無伺服器應用程式。

Note

您需要 Finch 在本機測試應用程式、使用 `--use-container` 選項建置部署套件，以及建置 OCI 映像函數。從 AWS SAM CLI 1.145.0 版開始，AWS SAM CLI 支援 Finch 做為的替代容器執行時間 Docker。

主題

- [安裝 Finch](#)
- [Finch 使用 CLI AWS SAM 設定](#)
- [驗證安裝](#)
- [停用 Finch](#)
- [疑難排解](#)
- [後續步驟](#)

安裝 Finch

請依照這些指示，在您的作業系統 Finch 上安裝。

先決條件

Note

Finch CLI AWS SAM 中的整合可在 macOS 和 Linux 系統上使用。對於 Windows 使用者，請考慮使用虛擬機器或 Linux 環境來使用 Finch 功能。

macOS 版本：

- macOS 15 Sequoia

- macOS 14 Sonoma

 Note

Finch 支援最新兩個版本的 macOS。

Linux :

- Linux 核心 v4.x+

安裝 Finch

macOS

在 macOS 上安裝 Finch

1. Finch 使用 brew CLI 安裝。如果出現提示，請輸入您的 macOS 密碼。

```
$ brew install finch
```

2. 檢查安裝的Finch版本。

```
$ finch --version
```

3. 初始化Finch虛擬機器（第一次設定）。

```
$ finch vm init
```

4. 啟動Finch虛擬機器。

```
$ finch vm start
```

5. 檢查Finch虛擬機器的狀態。

```
$ finch vm status
```

6. 驗證 是否正確Finch執行。

```
$ finch --info
```

```
$ finch ps
```

如需詳細資訊，請參閱[在 macOS 上安裝 Finch](#)。

Linux

您可以在 Finch Amazon Linux、Ubuntu 或使用一般安裝方法來安裝。如需詳細資訊，請參閱在[Linux 上安裝 Finch](#)。

Finch 使用 CLI AWS SAM 設定

安裝之後 Finch，CLI AWS SAM 會自動偵測，並在 Docker 未執行時使用 Finch 做為容器執行時間。

Note

Finch 做為 的後援 Docker。如果您同時安裝 Docker 和 Finch 並執行，AWS SAM CLI Docker 預設會排定優先順序。

管理員偏好設定組態 (macOS)

您可以建立管理員偏好設定檔案，將 AWS SAM CLI 設定為使用特定容器執行期做為預設值。此全系統偏好設定會覆寫預設的備用行為。

可用的組態選項

- Finch 設定為偏好的容器執行時間：

```
$ sudo /usr/libexec/PlistBuddy -c "Add :DefaultContainerRuntime string finch" /Library/Preferences/com.amazon.samcli.plist
```

- Docker 設定為偏好的容器執行時間：

```
$ sudo /usr/libexec/PlistBuddy -c "Add :DefaultContainerRuntime string docker" /Library/Preferences/com.amazon.samcli.plist
```

- 移除偏好設定（返回預設行為）：

```
$ sudo /usr/libexec/PlistBuddy -c "Remove :DefaultContainerRuntime" /Library/Preferences/com.amazon.samcli.plist
```

Note

未設定管理員偏好設定時，AWS SAM CLI Docker 預設會在可用時使用，在 Docker 未執行時將 Finch作為備用。

目錄掛載差異 (macOS)

在 macOS 上，如果您在主目錄 (~) 或 之外找到專案，AWS SAM 本機命令會失敗/Volumes。基於安全考量，此設計會限制自動掛載目錄，而無需您了解。

緩解選項

- 選項 1：在文字編輯器中開啟 Finch 組態 ~/.finch/finch.yaml，並在 additional_directories 區段下新增專案路徑，然後重新啟動 Finch 虛擬機器。如需詳細資訊，請參閱 [Finch 文件](#)。

```
$ finch vm stop  
$ finch vm start
```

- 選項 2：將專案移至主目錄。

驗證安裝

安裝並啟動 後 Finch，請確認它是否可與 CLI AWS SAM 搭配使用。

基本驗證

執行下列命令，以確認 AWS SAM CLI 可以使用 Finch：

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

您應該會看到使用 AWS SAM 在容器環境中 Finch 建置應用程式的 CLI。

Note

Finch 做為 的後援 Docker。如果 Docker 執行中，AWS SAM CLI Docker 預設會排定優先順序。

Finch 測試為備用

若要 Docker 在執行時測試 Finch 為備用，您必須先停止 Docker。

測試 Finch 備用功能

1. 停止為您的作業系統 Docker 使用適當的命令。

macOS：

```
$ docker stop $(docker ps -q)
$ pkill -f "Docker Desktop"
```

驗證 Docker 已停止。

```
$ docker ps
```

Linux：

```
$ docker stop $(docker ps -q)
$ sudo systemctl stop docker
```

驗證 Docker 已停止。

```
$ sudo systemctl status docker
```

2. 執行 `sam build --use-container` 命令來驗證 CLI AWS SAM 會自動使用 Finch 做為備用。

停用 Finch

如果您需要停止 Finch，請使用下列命令：

macOS：

```
$ finch vm stop
$ sudo pkill -f finch
```

驗證 Finch 已停止。

```
$ finch vm status
```

Linux：

```
$ sudo systemctl stop finch
$ sudo pkill -f finch
```

驗證Finch已停止。

```
$ sudo systemctl status finch
```

疑難排解

如果您遇到 問題Finch：

- 確認您的作業系統符合先決條件。
- 確認您已在系統上啟用虛擬化。
- 對於 macOS，請確認您的虛擬機器有足夠的磁碟空間。
- 對於 Linux，請確認您具有執行systemctl命令的適當許可。
- 如果您在將多架構映像推送至 Amazon Elastic Container Registry (Amazon ECR) 時遇到「不支援媒體類型」錯誤的 CloudFormation 部署失敗，請建立單一映像成品，而不是使用多架構映像。這是因為 會將影像索引和影像同時Finch推送至 Amazon ECR，而 只會Docker推送影像。
- 如果跨平台建置在目標函數架構與 Linux 上的主機機器架構不相符時失敗，請執行 來啟用跨平台模擬sudo finch run --privileged --rm tonistiigi/binfmt:master --install all。

如需其他疑難排解，請參閱 [Finch 文件](#) 或 [AWS SAM CLI 故障診斷](#)。

後續步驟

現在Finch已安裝，您可以搭配 CLI AWS SAM 使用它來：

- 使用容器搭配 建置無伺服器應用程式sam build --use-container。如需詳細資訊，請參閱[the section called “sam build”](#)。
- 使用 在本機測試 Lambda 事件函數sam local invoke。如需詳細資訊，請參閱[the section called “sam local invoke”](#)。

- 使用 啟動本機 API Gatewaysam local start-api。如需詳細資訊，請參閱[the section called “sam local start-api”](#)。
- 使用 啟動本機 Lambda 端點sam local start-lambda。如需詳細資訊，請參閱[the section called “sam local start-lambda”](#)。

如需搭配容器使用 AWS SAM CLI 的詳細資訊，請參閱 [使用的預設建置 AWS SAM](#)。

的影像儲存庫 AWS SAM

AWS SAM 透過建置容器映像的協助，簡化無伺服器應用程式的持續整合和持續交付 (CI/CD) 任務。AWS SAM 提供的映像包含 AWS SAM 命令列界面 (CLI) 和許多支援 AWS Lambda 執行時間的建置工具。這可讓您更輕鬆地使用 建置和封裝無伺服器應用程式 AWS SAM CLI。您可以將這些映像與 CI/CD 系統搭配使用，以自動化 AWS SAM 應用程式的建置和部署。如需範例，請參閱 [使用 CI/CD 系統和管道部署](#)。

AWS SAM 組建容器映像 URIs 會標記該映像中包含的 AWS SAM CLI 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` 使用最新的映像。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。

從 1.33.0 版開始 AWS SAM CLI，x86_64 和 arm64 容器映像都可用於支援的執行時間。如需詳細資訊，請參閱《AWS Lambda 開發人員指南》中的 [Lambda 執行時間](#)。

Note

在 1.22.0 版之前 AWS SAM CLI，DockerHub 是 AWS SAM CLI 提取容器映像的預設儲存庫。從 1.22.0 版開始，預設儲存庫已變更為 Amazon Elastic Container Registry Public (Amazon ECR Public)。若要從目前預設值以外的儲存庫提取容器映像，您可以使用 [sam build](#) 命令搭配 `--build-image` 選項。本主題結尾的範例示範如何使用 DockerHub 儲存庫映像建置應用程式。

映像儲存庫 URIs

下表列出 [Amazon ECR Public](#) 建置容器映像 URIs，您可以使用這些映像來建置和封裝無伺服器應用程式 AWS SAM。

Note

從 1.22.0 版 DockerHub 開始取代 AWS SAM CLI Amazon ECR Public。如果您使用的是舊版 AWS SAM CLI，我們建議您升級。

執行時期	Amazon ECR Public
自訂執行時間 (AL2023)	public.ecr.aws/sam/build-provided.al2023
自訂執行時間 (AL2)	public.ecr.aws/sam/build-provided.al2
自訂執行時間	public.ecr.aws/sam/build 提供的
Java 25	public.ecr.aws/sam/build-java25
Java 21	public.ecr.aws/sam/build-java21
Java 17	public.ecr.aws/sam/build-java17
Java 17 (AL2023)	public.ecr.aws/sam/build-java17.al2023
Java 11	public.ecr.aws/sam/build-java11
Java 11 (AL2023)	public.ecr.aws/sam/build-java11.al2023
Java 8 (AL2023)	public.ecr.aws/sam/build-java8.al2023
Java 8	public.ecr.aws/sam/build-java8
.NET 9	public.ecr.aws/sam/build-dotnet9
.NET 8	public.ecr.aws/sam/build-dotnet8
.NET 7	public.ecr.aws/sam/build-dotnet7
.NET 6	public.ecr.aws/sam/build-dotnet6
Node.js 24	public.ecr.aws/sam/build-nodejs24.x
Node.js 22	public.ecr.aws/sam/build-nodejs22.x

執行時期	Amazon ECR Public
Node.js 20	public.ecr.aws/sam/build-nodejs20.x
Node.js 18	public.ecr.aws/sam/build-nodejs18.x
Node.js 16	public.ecr.aws/sam/build-nodejs16.x
Python 3.14	public.ecr.aws/sam/build-python3.14
Python 3.13	public.ecr.aws/sam/build-python3.13
Python 3.12	public.ecr.aws/sam/build-python3.12
Python 3.11	public.ecr.aws/sam/build-python3.11
Python 3.10	public.ecr.aws/sam/build-python3.10
Python 3.9	public.ecr.aws/sam/build-python3.9
Python 3.8	public.ecr.aws/sam/build-python3.8
Ruby 4.0	public.ecr.aws/sam/build-ruby4.0
Ruby 3.4	public.ecr.aws/sam/build-ruby3.4
Ruby 3.3	public.ecr.aws/sam/build-ruby3.3
Ruby 3.2	public.ecr.aws/sam/build-ruby3.2

範例

下列兩個範例命令會使用來自映像儲存庫的容器映像來建置應用程式：

使用從 Amazon ECR 提取的容器映像建置 Node.js 24 應用程式：

```
$ sam build --use-container --build-image public.ecr.aws/sam/build-nodejs24.x
```

使用從 Amazon ECR 提取的 Python 3.14 容器映像建置函數資源：


```
$ sam build --use-container --build-image Function1=public.ecr.aws/sam/build-python3.14
```

使用 逐步部署無伺服器應用程式 AWS SAM

AWS Serverless Application Model (AWS SAM) 內建 [CodeDeploy](#)，可提供逐步 AWS Lambda 部署。只要幾行組態，就會為您 AWS SAM 執行下列動作：

- 部署 Lambda 事件函數的新版本，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，直到您對它如預期般運作感到滿意為止。如果更新無法正常運作，您可以轉返變更。
- 定義流量前和流量後測試函數，以確認新部署的程式碼已正確設定，且您的應用程式可如預期運作。
- 如果觸發 CloudWatch 警示，會自動復原部署。

Note

如果您透過 AWS SAM 範本啟用逐步部署，會自動為您建立 CodeDeploy 資源。您可以透過直接檢視 CodeDeploy 資源 AWS 管理主控台。

範例

下列範例示範如何使用 CodeDeploy 逐步將客戶轉移到新部署的 Lambda 函數版本：

```
Resources:
  MyLambdaFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.handler
      Runtime: nodejs20.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動作：

- **AutoPublishAlias**：透過新增此屬性並指定別名名稱，AWS SAM即可：
 - 根據 Lambda 函數的 Amazon S3 URI 的變更，偵測何時部署新程式碼。
 - 使用最新的程式碼建立和發佈該函數的更新版本。
 - 使用您提供的名稱建立別名（除非別名已存在），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**：在先前的範例中，10% 的客戶流量會立即轉移到新版本。10 分鐘後，所有流量都會轉移到新版本。不過，如果您的流量前或流量後測試失敗，或觸發 CloudWatch 警示，CodeDeploy 會復原您的部署。您可以透過下列方式指定如何在版本之間轉移流量：
 - **Canary**：流量以兩個增量轉移。您可以從預先定義的 Canary 選項中進行選擇。選項會指定在第一個增量中轉移到更新 Lambda 函數版本的流量百分比，以及剩餘流量在第二個增量中轉移之前的間隔，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。您可以選擇預先定義的線性選項，以指定以每個增量移動的流量百分比，以及每個增量之間的分鐘數。
 - **AllAtOnce**：所有流量都會一次從原始 Lambda 函數轉移到更新的 Lambda 函數版本。

下表概述範例所用流量轉移選項以外的其他可用流量轉移選項。

部署偏好類型

Canary10Percent30Minutes

Canary10Percent5Minutes

Canary10Percent10Minutes

Canary10Percent15Minutes

Linear10PercentEvery10Minutes

部署偏好類型

Linear10PercentEvery1Minute

Linear10PercentEvery2Minutes

Linear10PercentEvery3Minutes

AllAtOnce

- **Alarms**：這些是由部署引發的任何錯誤觸發的 CloudWatch 警示。遇到時，它們會自動復原您的部署。例如，如果您正在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 指標已超過警示閾值。
- **Hooks**：這些是流量轉移開始到新版本之前，以及在流量轉移完成後執行檢查的流量前和流量後測試函數。
 - **PreTraffic**：在流量轉移開始之前，CodeDeploy 會叫用預先流量掛鉤 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，它會中止並回報失敗 CloudFormation。如果函數成功，CodeDeploy 會繼續進行流量轉移。
 - **PostTraffic**：流量轉移完成後，CodeDeploy 會叫用流量後勾點 Lambda 函數。這類似於預先流量掛鉤，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](#)。

第一次逐漸部署 Lambda 事件函數

逐步部署 Lambda 函數時，CodeDeploy 需要先前部署的函數版本，才能從中轉移流量。因此，您的第一個部署應完成兩個步驟：

- 步驟 1：部署 Lambda 函數，並使用自動建立別名 `AutoPublishAlias`。
- 步驟 2：使用執行逐步部署 `DeploymentPreference`。

在兩個步驟中執行您的第一次逐步部署，可讓 CodeDeploy 先前的 Lambda 函數版本轉移流量。

步驟 1：部署 Lambda 事件函數

Resources:

MyLambdaFunction:

```
Type: AWS::Serverless::Function
Properties:
  Handler: index.handler
  Runtime: nodejs20.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: nodejs20.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作範例，請參閱完成 AWS SAM 研討會中的[單元 5 - Canary Deployments](#)。

的重要參考備註 AWS SAM

本節包含 AWS Serverless Application Model () 的重要備註和公告AWS SAM。

主題

- [2023 年重要備註](#)

2023 年重要備註

2023 年 10 月

AWS SAM CLI 停止對 的支援 Python 3.7

發佈 2023-10-20

Python 3.7 已於 2023 年 6 月收到end-of-life狀態。將於 AWS SAM CLI2023 年 10 月 24 Python 3.7 日停止對 的支援。如需詳細資訊，請參閱 aws-sam-cli GitHub儲存庫的[公告](#)。

此變更會影響下列使用者：

- 如果您透過 AWS SAM CLI使用Python 3.7並安裝 pip。
- 如果您使用 aws-sam-cli做為程式庫，並使用 建置應用程式Python 3.7。

如果您透過其他方法安裝和管理 AWS SAM CLI，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。

此變更不會影響對Python 3.7 AWS Lambda 執行時間環境的支援。如需詳細資訊，請參閱《AWS Lambda 開發人員指南》中的[執行期棄用政策](#)。

AWS SAM CLI Terraform 支援

本節涵蓋搭配您的Terraform專案和Terraform雲端使用 AWS Serverless Application Model 命令列界面 (AWS SAM CLI)。

若要提供意見回饋並提交功能請求，請建立[GitHub問題](#)。

主題

- [開始使用的 Terraform 支援 AWS SAM CLI](#)
- [搭配使用 AWS SAM CLITerraform進行本機偵錯和測試](#)
- [搭配 AWS SAM CLIServerless.tf 使用 進行本機偵錯和測試](#)
- [AWS SAM CLI 含Terraform參考](#)
- [什麼是 AWS SAM CLI 的支援Terraform？](#)

開始使用的 Terraform 支援 AWS SAM CLI

本主題說明如何開始使用 AWS Serverless Application Model 命令列界面 (AWS SAM CLI) 搭配 Terraform。

若要提供意見回饋並提交功能請求，請建立[GitHub問題](#)。

主題

- [AWS SAM CLI Terraform 先決條件](#)
- [搭配使用 AWS SAM CLI命令 Terraform](#)
- [設定Terraform專案](#)
- [設定 Terraform Cloud](#)

AWS SAM CLI Terraform 先決條件

完成所有先決條件，以開始使用 AWS SAM CLI搭配您的Terraform專案。

1. 安裝或升級 AWS SAM CLI

若要檢查是否已安裝 AWS SAM CLI，請執行下列動作：

```
$ sam --version
```

如果已安裝 AWS SAM CLI，輸出會顯示版本。若要升級至最新版本，請參閱 [升級 AWS SAM CLI](#)。

如需安裝 AWS SAM CLI 及其所有先決條件的說明，請參閱 [安裝 AWS SAM CLI](#)。

2. 安裝 Terraform

若要檢查您是否 Terraform 已安裝，請執行下列動作：

```
$ terraform -version
```

若要安裝 Terraform，請參閱在 Terraform 登錄檔中 [安裝 Terraform](#)。

3. 安裝 Docker 進行本機測試

需要 AWS SAM CLI Docker 進行本機測試。若要安裝 Docker，請參閱 [安裝 Docker 以搭配使用 AWS SAM CLI](#)。

搭配使用 AWS SAM CLI 命令 Terraform

當您執行支援的 AWS SAM CLI 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
```

您可以使用下列方式在 AWS SAM CLI 組態檔案中設定此選項：

```
hook_name = "terraform"
```

設定 Terraform 專案

完成本主題中的步驟，以搭配 Terraform 專案使用 AWS SAM CLI。

如果您在 Terraform 專案外部建置 AWS Lambda 成品，則不需要額外的設定。請參閱 [搭配使用 AWS SAM CLI Terraform 進行本機偵錯和測試](#) 以開始使用 AWS SAM CLI。

如果您在 Terraform 專案中建置 Lambda 成品，您必須執行下列動作：

1. 安裝 Python 3.8 或更新版本
2. 安裝 Make 工具。
3. 定義 Lambda 成品在 Terraform 專案中建置邏輯。
4. 定義 sam metadata 資源，以通知 AWS SAM CLI 您的建置邏輯。
5. 使用 AWS SAM CLI `sam build` 命令來建置 Lambda 成品。

安裝 Python 3.8 或更新版本

Python 需要 3.8 或更新版本才能與 搭配使用 AWS SAM CLI。當您執行 `sam build`，AWS SAM CLI 會建立 `makefiles`，其中包含建置 Lambda 成品的 Python 命令。

如需安裝說明，請參閱 [Python 入門指南中的下載 Python](#)。

執行下列動作，確認 Python 3.8 或更新版本已新增至您的機器路徑：

```
$ python --version
```

輸出應會顯示 3.8 或更新版本的 Python 版本。

安裝 Make 工具

GNU [Make](#) 是一種工具，可控制專案產生可執行檔和其他非來源檔案。AWS SAM CLI 會建立 `makefiles` 依賴此工具來建置 Lambda 成品。

如果您尚未在本機電腦上安裝 Make，請在繼續之前安裝它。

對於 Windows，您可以使用 [Chocolatey](#) 安裝。如需說明，請參閱如何在 Windows 中安裝和使用「製作」中的 [使用 Chocolatey](#)。

定義 Lambda 成品建置邏輯

使用 `null_resource` Terraform 資源類型來定義 Lambda 建置邏輯。以下是使用自訂建置指令碼來建置 Lambda 事件函數的範例。

```
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sam metadata資源

sam metadata 資源是一種 `null_resource` Terraform 資源類型，可提供 AWS SAM CLI 找到 Lambda 成品所需的資訊。專案中每個 Lambda 函數或 layer 都需要唯一的 sam metadata 資源。若要進一步了解此資源類型，請參閱 Terraform 登錄檔中的 [null_resource](#)。

定義sam metadata資源

1. 以開頭為您的資源命名 `sam_metadata_`，以將資源識別為 sam metadata 資源。
2. 在資源的 `triggers` 區塊內定義 Lambda 成品屬性。
3. 使用 `depends_on` 引數指定 `null_resource` 包含 Lambda 建置邏輯的。

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
  ]
}

```

以下是範例 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
    ]
}

```

資源的內容sam metadata會根據 Lambda 資源類型（函數或層）和封裝類型（ZIP 或映像）而有所不同。如需詳細資訊，以及範例，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 SAM CLI命令時，AWS SAM CLI會在執行命令 AWS SAM CLI之前產生中繼資料檔案。產生此檔案後，您可以使用 `--skip-prepare-infra` 選項搭配未來的 AWS SAM CLI命令，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 AWS SAM CLI 建置 Lambda 成品

使用 AWS SAM CLI的 `build` 命令來建置 Lambda 成品。當您執行 `aws-sam build` 時，AWS SAM CLI執行下列動作：

1. 尋找Terraform專案中的sam metadata資源，以了解並找到您的 Lambda 資源。
2. 啟動 Lambda 建置邏輯以建置 Lambda 成品。
3. 建立 `.aws-sam` 目錄來組織您的Terraform專案，以便與 AWS SAM CLI的 `local` 命令搭配使用。

使用 sam 建置建置

1. 從包含Terraform根模組的目錄中，執行下列動作：

```
$ sam build --hook-name terraform
```

2. 若要建置特定的 Lambda 函數或 layer，請執行下列動作

```
$ sam build --hook-name terraform lambda-resource-id
```

Lambda 資源 ID 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_function.this[0]`。

如果您的函數原始程式碼或其他Terraform組態檔案位於包含Terraform根模組的目錄之外，您需要指定位置。使用 `--terraform-project-root-path` 選項指定包含這些檔案的最上層目錄的絕對或相對路徑。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 AWS SAM CLI `sam build` 命令時，您可以設定 AWS SAM CLI 以使用本機 Docker 容器建置應用程式。

Note

您必須 Docker 已安裝並設定。如需說明，請參閱 [安裝 Docker 以搭配使用 AWS SAM CLI](#)。

使用容器建置

1. 建立 Dockerfile 包含 Terraform、Python 和 Make 工具的。您也應該包含 Lambda 函數執行時間。

以下是範例 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 libgcc libgcc-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. 使用 [docker build](#) 建置 Docker 映像。

以下是範例：

```
$ docker build --tag terraform-build:v1 <path-to-directory-containing-Dockerfile>
```

3. 使用 AWS SAM CLI `--use-container` 和 `--build-image` 選項執行 `sam build` 命令。

以下是範例：

```
$ sam build --use-container --build-image terraform-build:v1
```

後續步驟

若要開始將 AWS SAM CLI 與 Terraform 專案搭配使用，請參閱 [搭配使用 AWS SAM CLITerraform 進行本機偵錯和測試](#)。

設定 Terraform Cloud

我們建議您使用 Terraform v1.6.0 或更新版本。如果您使用的是較舊的版本，您必須在本機產生 Terraform 計劃檔案。本機計劃檔案會提供 AWS SAM CLI 執行本機測試和偵錯所需的資訊。

產生本機計劃檔案

Note

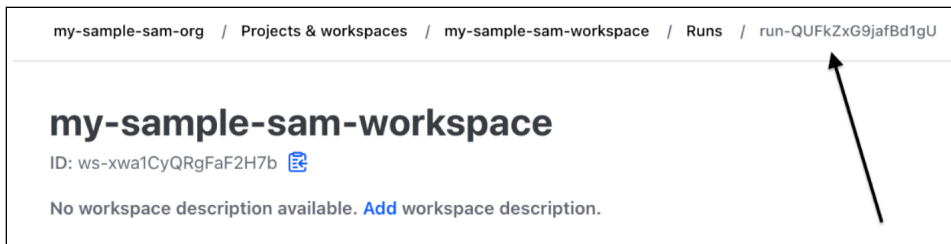
Terraform v1.6.0 或更新版本不需要這些步驟。若要開始使用 AWS SAM CLI 搭配 Terraform Cloud，請參閱 [搭配使用 AWS SAM CLI Terraform](#)。

1. 設定 API 字符 – 字符類型將取決於您的存取層級。若要進一步了解，請參閱 Terraform Cloud 文件中的 [API 權杖](#)。
2. 設定您的 API 字符環境變數 – 以下是命令列中的範例：

```
$ export TOKEN="<api-token-value>"
```

3. 取得您的執行 ID – 從 Terraform Cloud 主控台，找到您要與 搭配使用之 Terraform 執行的執行 ID AWS SAM CLI。

執行 ID 位於執行的導覽路徑中。



4. 擷取計劃檔案 – 使用您的 API 字符，取得您的本機計劃檔案。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
```

您現在可以將 AWS SAM CLI 與 搭配使用 Terraform Cloud。使用支援的 AWS SAM CLI 命令時，請使用 `--terraform-plan-file` 選項來指定本機計劃檔案的名稱和路徑。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` 命令的範例：

```
$ sam local start-api --hook-name terraform --terraform-plan-file custom-plan.json
```

如需可與這些範例搭配使用的範例應用程式，請參閱 `aws-samples` GitHub 儲存庫中的 [api_gateway_v2_tf_cloud](#)。

後續步驟

若要開始使用 AWS SAM CLI 搭配 Terraform Cloud，請參閱 [搭配 使用 AWS SAM CLITerraform 進行本機偵錯和測試](#)。

搭配 使用 AWS SAM CLITerraform 進行本機偵錯和測試

本主題說明如何搭配您的 Terraform 專案和 使用支援的 AWS Serverless Application Model 命令列界面 (AWS SAM CLI) 命令 Terraform Cloud。

若要提供意見回饋並提交功能請求，請建立 [GitHub 問題](#)。

主題

- [使用 進行本機測試 sam local invoke](#)
- [使用 進行本機測試 sam local start-api](#)
- [使用 進行本機測試 sam local start-lambda](#)
- [Terraform 限制](#)

使用 進行本機測試 sam local invoke

Note

若要使用 在 AWS SAM CLI 本機測試，您必須安裝並設定 Docker。如需說明，請參閱 [安裝 Docker 以搭配 使用 AWS SAM CLI](#)。

以下是透過傳入事件在本機測試 Lambda 事件函數的範例：

```
$ sam local invoke --hook-name terraform hello_world_function -e events/event.json -
```

若要進一步了解如何使用此命令，請參閱 [使用 進行測試的簡介 sam local invoke](#)。

使用 進行本機測試 sam local start-api

若要 sam local start-api 搭配 使用 Terraform，請執行下列動作：

```
$ 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)
```

若要進一步了解此命令，請參閱 [使用 進行測試的簡介 sam local start-api](#)。

使用 Lambda 授權方的 Lambda 事件函數

對於設定為使用 Lambda 授權方的 Lambda 函數，在 AWS SAM CLI 叫用 Lambda 函數端點之前，會自動叫用您的 Lambda 授權方。

- 若要進一步了解 中的此功能 AWS SAM CLI，請參閱 [使用 Lambda 授權方的 Lambda 事件函數](#)。
- 如需在 中使用 Lambda 授權方的詳細資訊 Terraform，請參閱 Terraform 登錄 [Resource: aws_api_gateway_authorizer](#) 檔中的。

使用 進行本機測試 sam local start-lambda

以下是使用 AWS Command Line Interface () 在本機測試 Lambda 函數的範例 AWS CLI：

1. 使用 AWS SAM CLI 建立本機測試環境：

```
$ sam local start-lambda --hook-name terraform hello_world_function
```

2. 使用 AWS CLI 在本機叫用您的 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
```

若要進一步了解此命令，請參閱 [使用 進行測試的簡介 sam local start-lambda](#)。

Terraform 限制

以下是搭配 AWS SAM CLI 使用 時的限制 Terraform：

- 連結至多層的 Lambda 函數。
- Terraform 定義資源之間連結的本機變數。
- 參考尚未建立的 Lambda 函數。這包括在 REST API 資源的內文屬性中定義的函數。

若要避免這些限制，您可以在新增資源 `terraform apply` 時執行。

搭配 AWS SAM CLI `Serverless.tf` 使用 進行本機偵錯和測試

AWS Serverless Application Model 命令列界面 (AWS SAM CLI) 可與 `Serverless.tf` 模組搭配使用，以進行 AWS Lambda 事件函數和層的本機偵錯和測試。支援下列 AWS SAM CLI 命令：

- `sam build`
- `sam local invoke`
- `sam local start-api`
- `sam local start-lambda`

Note

`Serverless.tf` 4.6.0 版和更新版本支援 AWS SAM CLI 整合。

若要開始搭配 `Serverless.tf` 模組使用 AWS SAM CLI，請更新至最新版本的 `Serverless.tf` 和 AWS SAM CLI。

從 `serverless.tf` 6.0.0 版開始，您必須將 `create_sam_metadata` 參數設定為 `true`。這會產生 `sam build` 命令所需的中繼資料資源 AWS SAM CLI。

若要進一步了解 `Serverless.tf`，請參閱 [terraform-aws-lambda-module](#)。

AWS SAM CLI 含 Terraform 參考

本節是搭配 使用 AWS Serverless Application Model 命令列界面 (AWS SAM CLI) Terraform 進行本機偵錯和測試的參考。

若要提供意見回饋並提交功能請求，請建立[GitHub問題](#)。

AWS SAM 支援的功能參考

如需支援搭配使用的功能參考 AWS SAM CLI文件，Terraform請參閱：

- [sam build](#)
- [sam local invoke](#)
- [sam local start-api](#)
- [sam local start-lambda](#)

Terraform 特定參考

您可以在這裡Terraform找到搭配使用 AWS SAM CLI 的特定參考文件：

- [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。

- 如需搭配使用 AWS Serverless Application Model 命令列界面 (AWS SAM CLI) 的簡介Terraform，請參閱 [什麼是 AWS SAM CLI 的支援Terraform？](#)。
- 若要搭配 AWS SAM CLI使用 Terraform，請參閱 [搭配使用 AWS SAM CLITerraform進行本機偵錯和測試](#)。

主題

- [引數](#)
- [範例](#)

引數

引數	說明
built_output_path	Event AWS Lambda Function 建置成品的路徑。

引數	說明
<code>docker_build_args</code>	Docker 建置引數 JSON 物件的解碼字串。此為選用引數。
<code>docker_context</code>	包含 Docker 映像建置內容的目錄路徑。
<code>docker_file</code>	Docker 檔案的路徑。此路徑與 <code>docker_context</code> 路徑相對。 此為選用引數。預設值為 <code>Dockerfile</code> 。
<code>docker_tag</code>	建立的 Docker 映像標籤的值。此值是選用的。
<code>depends_on</code>	Lambda 函數或 layer 的建置資源路徑。若要進一步了解，請參閱 登錄檔中的 <code>depends_on</code> 引數 。Terraform
<code>original_source_code</code>	定義 Lambda 函數的路徑。此值可以是字串、字串陣列或解碼的 JSON 物件做為字串。 <ul style="list-style-type: none"> 對於字串陣列，只使用第一個值，因為不支援多個程式碼路徑。 對於 JSON 物件，<code>source_code_property</code> 也必須定義。
<code>resource_name</code>	Lambda 函數名稱。
<code>resource_type</code>	Lambda 函數套件類型的格式。接受的值為： <ul style="list-style-type: none"> <code>IMAGE_LAMBDA_FUNCTION</code> <code>LAMBDA_LAYER</code> <code>ZIP_LAMBDA_FUNCTION</code>
<code>source_code_property</code>	JSON 物件中 Lambda 資源程式碼的路徑。當 <code>original_source_code</code> 是 JSON 物件時，定義此屬性。

範例

使用 ZIP 套件類型參考 Lambda 事件函數的 sam 中繼資料資源

```
# Lambda function resource
resource "aws_lambda_function" "tf_lambda_func" {
  filename = "${path.module}/python/hello-world.zip"
  handler = "index.lambda_handler"
```

```

runtime = "python3.8"
function_name = "function_example"
role = aws_iam_role.iam_for_lambda.arn
depends_on = [
    null_resource.build_lambda_function # function build logic
]
}

# sam metadata resource
resource "null_resource" "sam_metadata_function_example" {
  triggers = {
    resource_name = "aws_lambda_function.function_example"
    resource_type = "ZIP_LAMBDA_FUNCTION"
    original_source_code = "${path.module}/python"
    built_output_path = "${path.module}/building/function_example"
  }
  depends_on = [
    null_resource.build_lambda_function # function build logic
  ]
}

```

使用映像套件類型參考 Lambda 事件函數的 sam 中繼資料資源

```

resource "null_resource" "sam_metadata_function {
  triggers = {
    resource_name = "aws_lambda_function.image_function"
    resource_type = "IMAGE_LAMBDA_FUNCTION"
    docker_context = local.lambda_src_path
    docker_file = "Dockerfile"
    docker_build_args = jsonencode(var.build_args)
    docker_tag = "latest"
  }
}

```

參考 Lambda 層的 sam 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]
}
```

什麼是 AWS SAM CLI 的支援Terraform ？

使用 AWS Serverless Application Model 命令列界面 (AWS SAM CLI) 搭配您的Terraform專案或 Terraform Cloud執行本機偵錯和測試：

- AWS Lambda 事件函數和圖層。
- Amazon API Gateway APIs。

如需 的簡介Terraform，請參閱[什麼是 Terraform？](#) HashiCorp Terraform 網站。

若要提供意見回饋並提交功能請求，請建立[GitHub問題](#)。

Note

作為 AWS SAM CLI整合剖析步驟的一部分，AWS SAM CLI 會處理使用者命令產生專案檔案和資料。命令輸出應保持不變，但在某些環境中，環境或執行器可能會在輸出中插入其他日誌或資訊。

主題

- [什麼是 AWS SAM CLI？](#)
- [如何搭配 AWS SAM CLI使用 Terraform？](#)
- [後續步驟](#)

什麼是 AWS SAM CLI？

AWS SAM CLI 是命令列工具，您可以搭配 AWS SAM 範本和支援的第三方整合使用，例如 Terraform，以建置和執行無伺服器應用程式。如需 的簡介 AWS SAM CLI，請參閱 [什麼是 AWS SAM CLI？](#)。

AWS SAM CLI 支援 的下列命令Terraform：

- `sam local invoke` – 在本機啟動 AWS Lambda 函數資源的一次性調用。若要進一步了解此命令，請參閱 [使用 進行測試的簡介 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。這種類型的測試有助於 API Gateway 端點調用的 Lambda 函數。若要進一步了解此命令，請參閱 [使用進行測試的簡介 `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 SDKs 在本機叫用函數。若要進一步了解此命令，請參閱 [使用進行測試的簡介 `sam local start-lambda`](#)。

如何搭配 AWS SAM CLI使用 Terraform？

[核心Terraform工作流程](#)包含三個階段：寫入、規劃和套用。透過 AWS SAM CLI 的支援Terraform，您可以利用 AWS SAM CLIsam local一組命令，同時繼續使用Terraform工作流程來管理上的應用程式 AWS。一般而言，這表示下列事項：

- 寫入 – 使用 將基礎設施撰寫為程式碼Terraform。
- 測試和偵錯 – 使用 在 AWS SAM CLI本機測試和偵錯您的應用程式。
- 計劃 – 套用前預覽變更。
- 套用 – 佈建您的基礎設施。

如需搭配 AWS SAM CLI使用的範例Terraform，請參閱AWS「運算部落格」中的[一起改善：AWS SAM CLI 和 HashiCorp Terraform](#)。

後續步驟

若要完成所有先決條件並設定 Terraform，請參閱 [開始使用的 Terraform 支援 AWS SAM CLI](#)。

使用 發佈您的應用程式 AWS SAM CLI

若要讓您的 AWS SAM 應用程式可供其他人尋找和部署，您可以使用 AWS SAM CLI 將其發佈到 AWS Serverless Application Repository。若要使用 發佈應用程式 AWS SAM CLI，您必須使用 AWS SAM 範本加以定義。您也必須在本機或在 雲端中 AWS 進行測試。

請依照本主題中的指示建立新的應用程式、建立新的現有應用程式版本，或更新現有應用程式的中繼資料。（您執行的動作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 範本中繼資料區段屬性](#)。

先決條件

AWS Serverless Application Repository 使用 將應用程式發佈至 之前 AWS SAM CLI，您必須具有下列項目：

- 已安裝 AWS SAM CLI。如需詳細資訊，請參閱[安裝 AWS SAM CLI](#)。若要判斷是否已安裝 AWS SAM CLI，請執行下列命令：

```
sam --version
```

- 有效的 AWS SAM 範本。
- 範本 AWS SAM 參考的應用程式程式碼和相依性。
- 語意版本，只需要公開共用您的應用程式。此值可以簡單到 1.0。
- 指向應用程式原始碼的 URL。
- README.md 檔案。此檔案應描述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 帳戶中之前對其進行設定。
- LICENSE.txt 檔案，只需要公開共用您的應用程式。
- 如果您的應用程式包含任何巢狀應用程式，您必須已將其發佈至 AWS Serverless Application Repository。
- 有效的 Amazon Simple Storage Service (Amazon S3) 儲存貯體政策，可針對您在封裝應用程式時上傳至 Amazon S3 的成品授予服務讀取許可。若要設定此政策，請執行下列動作：
 1. 開啟位於 <https://console.aws.amazon.com/s3/> 的 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. 在編輯儲存貯體政策頁面上，將下列政策陳述式貼入政策編輯器。在政策陳述式中，請務必在 Resource 元素中使用儲存貯體名稱，並在 Condition 元素中使用您的帳戶 AWS ID。Condition 元素中的表達式可確保 AWS Serverless Application Repository 僅具有從指定 AWS 帳戶存取應用程式的許可。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](#)。

JSON

```
{
  "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，將 Metadata 區段新增至您的 AWS SAM 範本。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 範本中繼資料區段屬性](#)。

步驟 2：封裝應用程式

執行下列 AWS SAM CLI 命令，將應用程式的成品上傳到 Amazon S3，並輸出名為 `packaged.yaml` 的新範本檔案：

```
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，但具有關鍵差異：CodeUri、LicenseUrl 和 ReadmeUrl 屬性指向 Amazon S3 儲存貯體和包含個別成品的物件。

`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 應用程式的私有版本發佈至 AWS Serverless Application Repository，請執行下列 AWS SAM CLI 命令：

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

`sam publish` 命令的輸出包含您應用程式在上的連結 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：共用應用程式（選用）

根據預設，您的應用程式會設定為私有，因此其他 AWS 帳戶看不到。若要與他人共用您的應用程式，您必須將其設為公開，或授予特定 AWS 帳戶清單的許可。

如需使用 共用應用程式的相關資訊 AWS CLI，請參閱《AWS Serverless Application Repository 開發人員指南》中的以[AWS Serverless Application Repository 資源為基礎的政策範例](#)。如需使用 共用應用程式的資訊 AWS 管理主控台，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

將應用程式發佈至 之後 AWS Serverless Application Repository，您可能想要發佈新版本的應用程式。例如，您可能已變更 Lambda 事件函數程式碼，或將新元件新增至應用程式架構。

若要更新您先前發佈的應用程式，請使用先前詳述的相同程序再次發佈應用程式。在範本檔案的 AWS SAM Metadata 區段中，提供與最初發佈時相同的應用程式名稱，但包含新的 SemanticVersion 值。

例如，請考慮使用 的名稱 `SampleApp` 和 `SemanticVersion` 的 發佈的應用程式 `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 `SemanticVersion 1.0.1`（或以外的任何項目 `1.0.0`）。

其他主題

- [AWS SAM 範本中繼資料區段屬性](#)

AWS SAM 範本中繼資料區段屬性

`AWS::ServerlessRepo::Application` 是中繼資料金鑰，可用來指定要發佈至 的應用程式資訊 AWS Serverless Application Repository。

Note

`AWS::ServerlessRepo::Application` 中繼資料金鑰不支援 CloudFormation [內部函數](#)。

Properties

此資料表提供 AWS SAM 範本 Metadata 區段屬性的相關資訊。本節是 AWS Serverless Application Repository 使用 將應用程式發佈至 的必要項目 AWS SAM CLI。

屬性	Type	必要	描述
Name	String	TRUE	應用程式名稱。 最小長度 = 1。最大長度 = 140。 模式：" <code>[a-zA-Z0-9\\-]+</code> ";
Description	String	TRUE	應用程式的描述。 最小長度 = 1。最大長度 = 256。
Author	String	TRUE	發佈應用程式的作者名稱。 最小長度 = 1。最大長度 = 127。 模式：" <code>^[a-z0-9]([a-z0-9] -(?!-))*[a-z0-9]?\$</code> ";
SpdxLicenseId	String	FALSE	有效的授權識別符。若要檢視有效授權識別符的清單，請參閱 軟體套件資料交換 (SPDX) 網站上的 SPDX 授權清單 。
LicenseUrl	String	FALSE	本機授權檔案的參考，或授權檔案的 Amazon S3 連結，符合您應用程式的 <code>spdxLicenseId</code> 值。

屬性	Type	必要	描述
			<p>尚未使用 <code>sam package</code> 命令封裝的 AWS SAM 範本檔案可以參考此屬性的本機檔案。不過，若要使用 <code>sam publish</code> 命令發佈應用程式，此屬性必須是 Amazon S3 儲存貯體的參考。</p> <p>大小上限：5 MB。</p> <p>您必須為此屬性提供值，才能將您的應用程式公開。請注意，發佈應用程式後就無法更新此屬性。因此，若要將授權新增至應用程式，您必須先將其刪除，或使用不同的名稱發佈新的應用程式。</p>
ReadmeUrl	String	FALSE	<p>本機讀我檔案的參考或讀我檔案的 Amazon S3 連結，其中包含應用程式及其運作方式的更詳細說明。</p> <p>尚未使用 <code>sam package</code> 命令封裝的 AWS SAM 範本檔案可以參考此屬性的本機檔案。不過，若要使用 <code>sam publish</code> 命令發佈，此屬性必須是 Amazon S3 儲存貯體的參考。</p> <p>大小上限：5 MB。</p>
Labels	String	FALSE	<p>可改善搜尋結果中應用程式探索的標籤。</p> <p>最小長度 = 1。最大長度 = 127。最大標籤數量：10。</p> <p>模式：<code>"^[a-zA-Z0-9+\\-_:\\/@]+\$"</code>;</p>
HomePageUrl	String	FALSE	<p>包含應用程式詳細資訊的 URL，例如應用程式的 GitHub 儲存庫位置。</p>
SemanticVersion	String	FALSE	<p>應用程式的語義版本。如需語意版本控制規格，請參閱 語意版本控制 網站。</p> <p>您必須為此屬性提供值，才能將您的應用程式公開。</p>
SourceCodeUrl	String	FALSE	<p>應用程式原始程式碼的公有儲存庫連結。</p>

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。未針對指定使用案例列出的屬性會被忽略。

- 建立新的應用程式 – 如果 中沒有 AWS Serverless Application Repository 具有帳戶相符名稱的應用程式，則會建立新的應用程式。
 - 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 事件函數程式碼）
- 建立應用程式版本 – 如果 中 AWS Serverless Application Repository 已有符合帳戶名稱的應用程式，且 SemanticVersion 正在變更，則會建立應用程式版本。
 - 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 符合帳戶名稱的應用程式，且 SemanticVersion 未變更，則會更新應用程式。
- Description
- Author
- ReadmeUrl
- Labels
- HomePageUrl

範例

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
```

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。

- 文件最近更新時間：2026 年 10 月 9 日

變更	描述	日期
新增AWS::Serverless::WebFunction 資源	使用 AWS::Serverless::WebFunction 定義 Lambda Web 函數，透過受管端點提供 HTTP 請求。如需詳細資訊，請參閱 AWS::Serverless::WebFunction 。	2026 年 10 月 12 日
新增SelfManagedKafka事件來源的 OAuth 2.0和 IAM 身分驗證類型	SelfManagedKafka 事件來源的 SourceAccessConfigurations 屬性現在接受 OAUTHBEARER_AUTH、IAM_AUTH和 IAM_OAUTHBEARER_AUTH 身分驗證機制，以及 OAUTHBEARER_SCOPE、OAUTHBEARER_LOGICAL_CLUSTER、OAUTHBEARER_AUDIENCE 和 OAUTHBEARER_IDENTITY_POOL 中繼資料類型。如需詳細資訊，請參閱 SelfManagedKafka 。	2026 年 10 月 8 日
新增的 ManagedResourceTags 屬性 AWS::Serverless::CapacityProvider	將明確標籤套用至容量提供者代表您佈建的 Amazon Elastic Compute Cloud 執行個體和其他受管資源。如需詳細資訊，	2026 年 8 月 25 日

	請參閱 ManagedResourceTag S 。	
新增的 LoggingConfig 屬性 AWS::Serverless::CapacityProvider	設定容量提供者的系統記錄，以便擴展活動和操作事件在可設定的日誌層級傳送至 Amazon CloudWatch 日誌群組。如需詳細資訊，請參閱 AWS::Serverless::CapacityProvider 。	2026 年 8 月 24 日
新增命令的 AWS SAM CLI --output 選項	指定 --output json 以列印結構化、機器可讀取的輸出，您可以透過程式設計方式使用，例如在 CI/CD 管道和 IDE 延伸模組中。選項可用於 sam 建置 、 sam 部署 和 sam init 。	2026 年 8 月 19 日
新增函數程式碼和 layer 內容的 StorageMode 屬性	控制 Lambda 是否儲存部署套件的副本，或從 Amazon S3 儲存貯體參考它。如需了解詳細資訊，請參閱 FunctionCode 和 LayerContent 。	2026 年 8 月 4 日
將建置 Rust 函數的支援更新 AWS SAM CLI 為一般可用性	使用 建置 Rust 函數 Cargo Lambda 不再需要選擇加入 Beta 版功能。包含多個函數的專案指南現在使用 Cargo 工作區。若要進一步了解，請參閱 使用 建置 Rust Lambda 函數 Cargo Lambda 。	2026 年 8 月 3 日
新增 AWS SAM CLI 對 Java 上的 8、11 Java 和 Java 17 的支援 AL2023	java8.al2023 、 java11.al2023 和 java17.al2023 現可做為執行期和映像儲存庫使用。請參閱 映像儲存庫 和 sam init 。	2026 年 7 月 13 日

[新增 AWS::Serverless::MicrovmImage 和資源 AWS::Serverless::NetworkConnector 類型](#)

新的參考章節說明如何從存放在 Amazon S3 中的程式碼成品建立 Lambda MicroVM 映像，以及如何定義 Lambda 運算資源的網路連線。如需了解詳細資訊，請參閱 [AWS::Serverless::MicrovmImage](#) 和 [AWS::Serverless::NetworkConnector](#)。

2026 年 7 月 6 日

[新增 sam 部署和 sam 同步命令的 AWS SAM CLI --express 選項](#)

使用 CloudFormation 快速模式，以便在套用資源組態時完成堆疊操作，同時在背景繼續準備資源。若要進一步了解，請參閱 [sam 部署](#) 和 [sam 同步](#)。

2026 年 7 月 1 日

[新增 sam local start-api 和 sam local start-lambda 命令的 AWS SAM CLI --no-watch 選項](#)

當您使用時 `--warm-containers`，請針對監看會造成高 CPU 或 I/O 額外負荷的專案，停用檔案監看。若要進一步了解，請參閱 [sam local start-api](#) 和 [sam local start-lambda](#)。

2026 年 6 月 10 日

[新增 AWS SAM CLI](#)[對AWS::LanguageExtensions 轉換的支援](#)

現在 AWS SAM CLI 可使用 `Fn::FindInMap`、`DefaultValue`、`Fn::ForEach`、`Fn::ToJsonString`、`Fn::Length` 和 `Fn::Split`，因此 `sam build`、`sam package`、`sam validate`、`sam deploy` 和 `sam sync` 命令可使用使用 `AWS::LanguageExtensions` 轉換的範本。若要進一步了解，請參閱 [CloudFormation 語言延伸支援](#)。

2026 年 5 月 13 日

[新增 AWS SAM CLI 對 Ruby 4.0 的支援](#)

Ruby 4.0 現已提供做為執行時間和映像儲存庫。請參閱 [映像儲存庫](#) 和 [sam init](#)。

2026 年 4 月 27 日

[新增AWS::Serverless::WebSocketApi資源類型](#)

WebSocket API 使用 定義 AWS SAM，包括路由、授權、自訂網域和 Route 53 組態。如需詳細資訊，請參閱 [AWS::Serverless::WebSocketApi](#)。

2026 年 4 月 13 日

[已擴展本機命令的 AWS SAM CLI --invoke-image 選項](#)

為每個資源指定 `--invoke-image` 多次使用不同的容器映像，並覆寫使用的 `ImageUri` 函數 `PackageType: Image`。若要進一步了解，請參閱 [sam 本機調用](#)。

2026 年 4 月 10 日

新增的 SecurityPolicy 和 EndpointAccessMode 屬性 AWS::Serverless::Api	在自訂網域上設定 TLS 安全政策和端點存取模式。如需詳細資訊，請參閱 DomainConfiguration 。	2026 年 4 月 10 日
新增容器映像 DockerImage 組建的組建屬性	為 AWS SAM CLI 建置的 Docker 映像指定自訂名稱。若要進一步了解，請參閱 建置您的應用程式 。	2026 年 4 月 10 日
新增本機命令的 AWS SAM CLI --container-dns 選項	指定本機容器使用的 DNS 伺服器。選項適用於 sam 本機調用 、 sam 本機 start-api 和 sam 本機 start-lambda 。	2026 年 4 月 10 日
新增的 Amazon S3 檔案系統支援 AWS::Serverless::Function	除了 Amazon EFS 存取點之外，還會將 Amazon S3 Files 存取點連接至具有 <code>FileSystemConfigs</code> 的函數。如需詳細資訊，請參閱 AWS::Serverless::Function 。	2026 年 4 月 8 日
新增 AWS SAM CLI 對 Docker BuildKit 的支援	使用 <code>建置容器映像函數 BuildKit</code> 。若要進一步了解，請參閱 使用垃圾郵件建置 。	2026 年 3 月 10 日
新增 --env-vars 選項 .env 的檔案支援	除了 JSON 之外， <code>local</code> 命令 <code>--env-vars</code> 的選項現在還接受 <code>.env</code> 檔案。若要進一步了解，請參閱在 本機調用函數 。	2026 年 3 月 6 日
新增的 IpAddressType 屬性 AWS::Serverless::Api	將 API 端點設定為 <code>ipv4</code> 或 <code>dualstack</code> 。如需詳細資訊，請參閱 EndpointConfiguration 。	2026 年 2 月 20 日

新增 AWS SAM CLI對uv套件管理員的支援

使用 建置 Python函數uv。若要進一步了解，請參閱[使用 建置 Python函數uv](#)。

2026 年 2 月 19 日

新增事件的 ResponseTransferMode 屬性

將 ResponseTransferMode 設定為 `RESPONSE_STREAM`，透過 串流函數回應 `RESPONSE_STREAM`。如需詳細資訊，請參閱 [Api](#)。

2026 年 2 月 11 日

新增 Amazon MSK 事件的 LoggingConfig 和 MetricsConfig 屬性

設定 Amazon MSK 事件來源的事件來源映射記錄和增強指標。如需詳細資訊，請參閱 [MSK](#)。

2026 年 2 月 2 日

新增的 TenancyConfig 屬性 AWS::Serverless::Function

設定函數的租用戶隔離，並在 `sam local invoke` 和 `sam remote invoke` 上使用新 `--tenant-id` 選項傳遞租用戶。如需詳細資訊，請參閱 [AWS::Serverless::Function](#)。

2025 年 11 月 20 日

新增 Amazon SQS 事件的 ProvisionedPollerConfig 屬性

透過設定輪詢器的最小和最大數量，為 Amazon SQS 事件來源使用佈建模式。如需詳細資訊，請參閱 [SQS](#)。

2025 年 11 月 20 日

新增SelfManagedKafka事件的重試和批次屬性

自我管理的 Kinesis 事件來源現在支援 `BisectBatchOnFunctionError`、`MaximumRecordAgeInSeconds`、`FunctionResponseType` 和 `MaximumRetryAttempts`。如需詳細資訊，請參閱 [SelfManagedKafka](#)。

2025 年 11 月 20 日

[新增 AWS SAM CLI對 24 Node.js 的支援](#)

Node.js 24 現在可做為執行期和映像儲存庫使用。請參閱[映像儲存庫](#)和 [sam init](#)。

2025 年 11 月 19 日

[新增 Amazon MSK 事件的重試和批次屬性](#)

Amazon MSK 事件來源現在支援 `BisectBatchOnFunctionError`、`MaximumRecordAgeInSeconds`、`FunctionResponseType` 和 `MaximumRetryAttempts`。如需詳細資訊，請參閱 [MSK](#)。

2025 年 11 月 19 日

[新增AWS::Serverless::CapacityProvider資源類型](#)

在您擁有的 Amazon Elastic Compute Cloud 容量上執行函數，並使用 `將容量提供者連接至函數CapacityProviderConfig`。如需詳細資訊，請參閱 [AWS::Serverless::CapacityProvider](#)。

2025 年 11 月 17 日

[新增 AWS SAM CLI對 Java 25 和 3.14 Python 的支援](#)

Java 25 和 Python 3.14 現可做為映像儲存庫和 `sam init` 基礎映像使用。請參閱[映像儲存庫](#)和 [sam init](#)。

2025 年 11 月 10 日

[新增 Amazon MSK 事件的 Enabled 屬性](#)

暫停和繼續 Amazon MSK 事件來源映射。如需詳細資訊，請參閱 [MSK](#)。

2025 年 10 月 16 日

[新增了保留Python套件結構的建置屬性](#)

指定 `ParentPackageMode` 和 `ParentPackages` 以下，`BuildProperties` 以便您建置的 Lambda 函數可以使用絕對匯入。若要進一步了解，請參閱[建置您的應用程式](#)。

2025 年 10 月 6 日

[新增對耐用函數的 AWS SAM 支援](#)

使用 `DurableConfig` 屬性設定耐用的函數，並使用 `sam local callback` 命令在本機測試。若要進一步了解，請參閱 [DurableConfig](#) 和 [測試耐用函數](#)。

2025 年 9 月 25 日

[新增 AWS SAM CLI 對的支援 Finch](#)

使用 Finch 而非 Docker 搭配 `--use-container` 和建置，以在本機測試您的應用程式。若要進一步了解，請參閱 [安裝 Finch](#)。

2025 年 8 月 19 日

[新增 Amazon MSK 事件的 BatchSize 屬性](#)

設定 Lambda 在單一批次中從 Amazon MSK 事件來源擷取的記錄數目上限。如需詳細資訊，請參閱 [MSK](#)。

2025 年 8 月 5 日

[新增將 Lambda 主控台函數轉換為的指引 AWS SAM](#)

將您在 Lambda 主控台中建立的函數移至 AWS SAM 範本，以便您可以將它做為基礎設施做為程式碼進行管理。若要進一步了解，請參閱 [將 Lambda 函數 AWS SAM 轉換為應用程式](#)。

2025 年 8 月 5 日

[新增 Kinesis 事件的 SchemaRegistryConfig 屬性](#)

使用結構描述登錄檔搭配 Amazon MSK 或自我管理的 Kinesis 事件來源。如需了解詳細資訊，請參閱 [MSK](#) 和 [SelfManagedKafka](#)。

2025 年 6 月 19 日

[新增 AWS SAM CLI 對 .NET 9 的支援](#)

.NET 9 現可做為映像儲存庫使用。請參閱 [映像儲存庫](#)。

2025 年 6 月 4 日

新增對私有 APIs 上自訂網域名稱的 AWS SAM 支援	使用新的 Policy 和 DomainAccessAssociation 屬性，將自訂網域連接至私有 REST API。如需了解詳細資訊，請參閱 DomainConfiguration 和 DomainAccessAssociation 。	2025 年 4 月 4 日
新增 AWS SAM CLI 對 Ruby 3.4 的支援	Ruby 3.4 現已提供做為執行期和映像儲存庫。請參閱 映像儲存庫 和 sam init 。	2025 年 3 月 24 日
新增 sam local invoke 命令的 AWS SAM CLI --runtime 選項	覆寫範本中定義的執行時間，以針對不同的執行時間測試函數。若要進一步了解，請參閱 sam 本機調用 。	2025 年 3 月 11 日
新增 命令的 AWS SAM CLI --mount-symlinks 和 --no-memory-limit 選項	在 sam build 和 sam local 調用 --mount-symlinks 上使用 將頂層符號連結掛載到建置容器中，並在 sam local 命令 --no-memory-limit 上使用 移除容器記憶體限制。	2025 年 2 月 26 日
新增 sam build 和 sam sync 命令的 AWS SAM CLI --no-use-container 選項	在本機機器而非容器內建置您的函數。若要進一步了解，請參閱 sam build 和 sam sync 。	2024 年 12 月 9 日
新增 AWS SAM CLI 對 Node.js 22 和 3.13 Python 的支援	Node.js 22 和 Python 3.13 現已提供執行時間和映像儲存庫。請參閱 映像儲存庫 和 sam init 。	2024 年 12 月 5 日
新增的 PublishLambdaVersion 屬性 AWS::Serverless::LayerVersion	控制 AWS SAM 是否發佈新的 layer 版本。如需詳細資訊，請參閱 AWS::Serverless::LayerVersion 。	2024 年 12 月 5 日

[新增事件來源映射的
ProvisionedPollerConfig 和 MetricsConfig
屬性](#)

搭配使用 Amazon MSK 和自我管理 Kinesis 事件來源的佈建模式 ProvisionedPollerConfig，並搭配加入 DynamoDB、Kinesis 和 Amazon SQS 事件來源的增強型事件來源映射指標 MetricsConfig。如需了解詳細資訊，請參閱 [MSK](#) 和 [SQS](#)。

2024 年 11 月 25 日

[新增的 SourceKMSKeyArn
屬性 AWS::Serverless::Function](#)

指定 AWS SAM 用來加密 .zip 部署套件的 AWS KMS 金鑰。如需詳細資訊，請參閱 [AWS::Serverless::Function](#)。

2024 年 11 月 14 日

[新增的 RecursiveLoop 屬性
AWS::Serverless::Function](#)

設定 Lambda 是否終止或允許遞迴函數叫用迴圈。如需詳細資訊，請參閱 [AWS::Serverless::Function](#)。

2024 年 8 月 27 日

[新增事件來源映射的
KmsKeyArn 屬性](#)

指定加密靜態事件來源映射篩選條件的 AWS KMS 金鑰。屬性可在 Amazon DocumentDB、DynamoDB、Kinesis、Amazon MQ、Amazon MSKSelfManagedKafka、和 Amazon SQS 事件來源上使用。如需詳細資訊，請參閱 [MSK](#)。

2024 年 8 月 21 日

[新增的 UseAliasAsEventTarget 屬性
AWS::Serverless::StateMachine](#)

使用建立的別名 AutoPublishAlias 做為事件來源目標。如需詳細資訊，請參閱 [AWS::Serverless::StateMachine](#)。

2024 年 8 月 12 日

[開發人員指南中的重組和更新內容](#)

重組和重組指南，以改善可探索性和可用性。更新並改善標題。介紹主題和概念時提供其他詳細資訊。

2024 年 6 月 20 日

[新增 AWS SAM CLI對 Ruby 3.3 的支援](#)

Ruby 3.3 現可做為執行期和映像儲存庫使用。如需詳細資訊，請參閱[映像儲存庫](#)和 [sam init](#)。

2024 年 4 月 4 日

[新增 AWS SAM CLI的命令選項](#)

新選項可用於命令 [sam local start-api](#)：--ssl-cert-file PATH、--ssl-key-file PATH。此外，新的命令列選項--add-host LIST 也可用於 [sam local invoke](#)、[sam local start-api](#) 和 [sam local start-lambda](#)

2024 年 3 月 20 日

[新增 AWS SAM CLI對 .NET 8 的支援](#)

.NET 8 現可做為執行期和映像儲存庫使用。不再支援 .NET Core 3.1、Node.js 14、Node.js 12、Python 3.7、Ruby 2.7 的執行時間和映像儲存庫。請參閱[映像儲存庫](#)和 [sam init](#)。

2024 年 2 月 22 日

[新增 AWS SAM CLI適用於的 arm64 套件安裝程式 Linux](#)

如需說明，請參閱[安裝 AWS SAM CLI](#)。

2023 年 12 月 6 日

[新增 sam sync 命令的 AWS SAM CLI --watch-exclude 選項](#)

排除檔案和資料夾啟動同步。若要進一步了解，請參閱[指定不會啟動同步的檔案和資料夾](#)。

2023 年 12 月 6 日

[新增 sam sync 命令的 AWS SAM CLI --build-in-source 選項](#)

在來源資料夾中建置您的專案，以加速建置程序。若要進一步了解，請參閱[在來源資料夾中建置您的專案，以加快建置時間](#)。

2023 年 12 月 6 日

[新增 sam build 命令的 AWS SAM CLI --build-in-source 選項](#)

在來源資料夾中建置您的專案，以加速建置程序。若要進一步了解，請參閱[在來源資料夾中建置您的專案，以加快建置時間](#)。

2023 年 12 月 6 日

[新增對 AWS SAM CLI遠端調用命令的新資源支援](#)

sam remote invoke 搭配 Kinesis Data Streams 應用程式、Amazon SQS 佇列和 Step Functions 狀態機器使用。若要進一步了解，請參閱[使用 sam 遠端叫用](#)。

2023 年 11 月 15 日

[新增 AWS SAM CLI可共用測試事件的新遠端 test-event 命令](#)

使用從 AWS SAM CLI EventBridge 結構描述登錄檔存取和管理可共用的測試事件，以在本地測試 Lambda 函數 AWS 雲端。若要進一步了解，請參閱[使用 sam 遠端測試事件](#)。

2023 年 10 月 3 日

[AWS SAM CLI 的支援Terraform現已正式推出](#)

若要進一步了解 AWS SAM CLI 的支援Terraform，請參閱[AWS SAM CLITerraform 支援](#)。

2023 年 9 月 5 日

[新增 AWS SAM CLI對 Terraform Cloud 的支援](#)

現在 AWS SAM CLI支援的本機測試Terraform Cloud。若要進一步了解，請參閱[設定 Terraform Cloud](#)。

2023 年 9 月 5 日

[新增組態YAML檔案的 AWS SAM CLI檔案格式支援](#)

現在 AWS SAM CLI支援【.yaml|.yml】檔案格式。[設定 AWS SAM CLI和AWS SAM CLI組態檔案](#)頁面已更新。

2023 年 7 月 18 日

[新增 AWS SAM CLI 的 sam local start-api命令支援 Terraform](#)

[什麼是 AWS SAM CLI 的支援 Terraform ?](#) 章節已更新為包含 AWS SAM CLI 的 sam local start-api 命令支援Terraform。

2023 年 7 月 6 日

[新增新的 AWS SAM CLI遠端叫用命令](#)

若要開始使用 sam remote invoke，請參閱[使用 sam 遠端叫用](#)。

2023 年 6 月 22 日

[新增 AWS AppSyncGraphQL API無伺服器資源類型](#)

建立新的[AWS::Serverless::GraphQLApi](#) 章節，說明如何使用 定義GraphQL API資源 AWS SAM。

2023 年 6 月 22 日

[新增 AWS SAM CLI對 3.2 Ruby 的支援](#)

更新 [sam init](#) 頁面以包含新的基礎映像和執行時間值。使用 Ruby 3.2 Amazon ECR URI 更新[映像儲存庫](#)頁面。

2023 年 6 月 6 日

[新增套件安裝程式完整性驗證的 AWS SAM CLI選用步驟](#)

更新 [安裝頁面 AWS SAM CLI](#)以反映選用步驟。建立 [驗證安裝程式頁面的 AWS SAM CLI完整性](#)以記錄步驟。

2023 年 5 月 31 日

[新增 sam 同步選項以略過基礎設施同步](#)

自訂每次sam sync執行時是否需要 CloudFormation 部署。若要進一步了解，請參閱[略過初始 CloudFormation 部署](#)。

2023 年 3 月 23 日

[新增對 DocumentDB 事件來源類型的支援](#)

AWS SAM 範本規格現在支援 `AWS::Serverless::Function` 資源 `DocumentDB` 的事件來源類型。若要進一步了解，請參閱 [DocumentDB](#)。

2023 年 3 月 10 日

[使用 建置 Rust Lambda 函數 Cargo Lambda](#)

使用 AWS SAM CLI 以使用 建置 Rust Lambda 函數 `Cargo Lambda`。若要進一步了解，請參閱 [使用 建置 Rust Lambda 函數 Cargo Lambda](#)。

2023 年 2 月 23 日

[在 外部建置函數資源 AWS SAM](#)

新增使用 `sam build` 命令時略過函數的指引。若要進一步了解，請參閱在 [外部建置 函數 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。若要進一步了解，請參閱 [使用 AWS SAM 連接器管理資源許可](#)。

2023 年 2 月 8 日

[新增的 sam list 命令 AWS SAM CLI](#)

使用 `sam list` 檢視有關無伺服器應用程式中資源的重要資訊。若要進一步了解，請參閱 [sam 清單](#)。

2023 年 2 月 2 日

[新增 esbuild 的格式和 OutExtension 建置屬性](#)

使用 `esbuild` 建置 `Node.js` Lambda 函數現在支援 `Format` 和 `OutExtension` 建置屬性。若要進一步了解，請參閱 [使用 esbuild 建置 Node.js Lambda 函數](#)。

2023 年 2 月 2 日

[將執行時間管理選項新增至 AWS SAM 範本規格](#)

為您的 Lambda 函數設定執行時間管理選項。如需詳細資訊，請參閱 [RuntimeManagementConfig](#)。

2023 年 1 月 24 日

[目標屬性已新增至 EventSource for AWS::Serverless::StateMachine 資源。](#)

AWS::Serverless::StateMachine 資源類型支援 [EventBridgeRule](#) 和 [Schedule](#) 事件來源的 Target 屬性。

2023 年 1 月 13 日

[設定 Lambda 函數 SQS 輪詢器的擴展](#)

使用的 [ScalingConfig](#) 屬性設定 SQS 輪詢器的擴展 `AWS::Serverless::Function`。如需詳細資訊，請參閱 [ScalingConfig](#)。

2023 年 1 月 12 日

[使用 cfn-lint 驗證 AWS SAM 應用程式](#)

您可以使用 cfn-lint 透過驗證您的 AWS SAM 範本 `AWS SAM CLI`。若要進一步了解，請參閱 [使用 cfn-lint 驗證](#)。

2023 年 1 月 11 日

[使用 CloudWatch Application Insights 監控無伺服器應用程式](#)

設定 Amazon CloudWatch Application Insights 以監控您的 AWS SAM 應用程式。若要進一步了解，請參閱 [使用 CloudWatch Application Insights 監控無伺服器應用程式](#)。

2022 年 12 月 19 日

[新增 AWS SAM CLI macOS 的套件安裝程式](#)

使用新的 AWS SAM CLI macOS 套件安裝程式安裝。若要進一步了解，請參閱 [安裝 AWS SAM CLI](#)。

2022 年 12 月 6 日

[新增對 Lambda SnapStart 的支援](#)

為您的 Lambda 函數設定 SnapStart 以建立快照，快照是初始化函數的快取狀態。如需詳細資訊，請參閱 [AWS::Serverless::Function](#)。

2022 年 11 月 28 日

[新增 AWS SAM CLI對 nodejs18.x 的支援](#)

AWS SAM CLI 現在支援 nodejs18.x 執行時間。若要進一步了解，請參閱 [sam init](#)。

2022 年 11 月 17 日

[新增有關設定存取和許可的指引](#)

AWS SAM 提供兩個選項，可簡化無伺服器 applications.To 的存取和許可管理，如需詳細資訊，請參閱[管理資源存取和許可](#)。

2022 年 11 月 17 日

[新增使用原生 AOT 編譯建置 .NET 7 Lambda 函數的支援](#)

使用 建置和封裝您的 .NET 7 Lambda 函數 AWS SAM，利用原生Ahead-of-Time(AOT)編譯來改善 Lambda 冷啟動時間。若要進一步了解，請參閱[使用原生 AOT 編譯建置 .NET 7 Lambda 函數](#)。

2022 年 11 月 15 日

[新增 AWS SAM CLI對本機偵錯和測試的Terraform支援](#)

使用Terraform專案中的 AWS SAM CLI，對 Lambda 函數和層執行本機偵錯和測試。若要進一步了解，請參閱 [AWS SAM CLI Terraform支援](#)。

2022 年 11 月 14 日

[新增對 EventBridge 排程器的 AWS SAM 支援](#)

AWS Serverless Application Model (AWS SAM) 範本規格提供簡單的速記語法，可讓您使用 AWS Lambda 和的 EventBridge 排程器來排程事件 AWS Step Functions。如需詳細資訊，請參閱[使用 EventBridge 排程器排程事件](#)。

2022 年 11 月 10 日

[簡化 AWS SAM CLI 安裝指示](#)

AWS SAM CLI 先決條件和選用步驟已移至不同的頁面。如需支援作業系統的安裝步驟，請參閱[安裝 AWS SAM CLI](#)。

2022 年 11 月 4 日

[新增修正以允許 Windows 10 使用者的長路徑](#)

應用程式 AWS SAM CLI 範本儲存庫包含一些較長的檔案路徑，這些路徑可能會在執行時 `sam init` 因為 Windows 10 `MAX_PATH` 限制而導致錯誤。如需詳細資訊，請參閱[安裝 AWS SAM CLI](#)

2022 年 11 月 4 日

[更新了第一次部署的逐步部署程序](#)

使用 逐漸部署 Lambda 函數 AWS CodeDeploy 需要兩個步驟。若要進一步了解，請參閱[第一次逐漸部署 Lambda 函數](#)。

2022 年 10 月 13 日

[支援更多類型事件的其他 Lambda 事件篩選](#)

`FilterCriteria` 屬性已新增至 [MSK](#)、[MQ](#) 和 [SelfManagedKafka](#) 事件來源類型。

2022 年 10 月 13 日

[新增 AWS SAM 管道的 OpenID Connect \(OIDC\) 支援](#)

AWS SAM 支援 Bitbucket、GitHub Actions 和 GitLab 持續整合和持續交付 (CI/CD) 平台的 OpenID Connect (OIDC) 使用者身分驗證。若要進一步了解，請參閱[搭配 AWS SAM 管道使用 OIDC 使用者帳戶](#)。

2022 年 10 月 13 日

[JwtConfiguration 屬性的注意事項](#)

新增在下為定義 issuer 和 audience 屬性 JwtConfiguration 的注意事項 [OAuth2Authorizer](#)。

2022 年 10 月 7 日

[Function 和 StateMachine EventSource 的新屬性](#)

Enabled 和 State 屬性已新增至 CloudWatchEvent 的事件來源 [AWS::Serverless::Function](#)。State 屬性已新增至 [AWS::Serverless::Function](#) 和 Schedule 的事件來源 [AWS::Serverless::StateMachine](#)。

2022 年 10 月 6 日

[AWS SAM 連接器現在已全面推出](#)

連接器是一種 AWS SAM 抽象資源類型，識別為 `AWS::Serverless::Connector`，提供在無伺服器應用程式資源之間佈建許可的簡單且安全方法。若要進一步了解，請參閱[使用 AWS Serverless Application Model 連接器管理資源許可](#)。

2022 年 10 月 6 日

[已將新的 sam 同步選項新增至 AWS SAM CLI](#)

`--dependency-layer` 和 `--use-container` 選項已新增至 [sam sync](#)。

2022 年 9 月 20 日

已將新的 sam 部署選項新增至 AWS SAM CLI	--on-failure 選項已新增至 sam deploy 。	2022 年 9 月 9 日
esbuild 支援現已全面推出	若要建置和封裝 Node.js Lambda 函數，您可以使用 AWS SAM CLI 搭配 esbuild JavaScript bundler 。	2022 年 9 月 1 日
更新 AWS SAM CLI 遙測	已更新所收集的 系統和環境資訊 的描述，以包含用量屬性的雜湊值。	2022 年 9 月 1 日
已將本機環境變數支援新增至 AWS SAM CLI	在 本機叫用 Lambda 函數 時，以及在 本機執行 API Gateway 時，使用 AWS SAM CLI 環境變數搭配。	2022 年 9 月 1 日
支援 Lambda 指令集架構	使用 AWS SAM CLI 為 x86_64 或 arm64 指令集架構建置 Lambda 函數和 Lambda 層。如需詳細資訊，請參閱 AWS::Serverless::Function 資源類型的 Architectures 屬性，以及 AWS::Serverless::LayerVersion 資源類型的 CompatibleArchitectures 屬性。	2021 年 10 月 1 日
產生範例管道組態	使用 AWS SAM CLI 來產生多個 CI/CD 系統的範例管道，並使用新的 sam pipeline bootstrap 和 sam pipeline init 命令。如需詳細資訊，請參閱 產生範例 CI/CD 管道 。	2021 年 7 月 21 日

[AWS SAM CLI 與 AWS CDK 整合 \(預覽, 階段 2\)](#)

透過公有預覽版本的第 2 階段, 您現在可以使用 AWS SAM CLI 來封裝和部署 AWS CDK 應用程式。您也可以直接使用下載範例 AWS CDK 應用程式 AWS SAM CLI。如需詳細資訊, 請參閱 [AWS Cloud Development Kit \(AWS CDK\) \(預覽\)](#)。

2021 年 7 月 13 日

[支援 RabbitMQ 做為函數的事件來源](#)

新增對 RabbitMQ 的支援, 做為無伺服器函數的事件來源。如需詳細資訊, 請參閱 [AWS::Serverless::Function](#) 資源類型 MQ 事件來源的 [SourceAccessConfigurations](#) 屬性。

2021 年 7 月 7 日

[使用 Amazon ECR 建置容器映像部署無伺服器應用程式](#)

使用 Amazon ECR 建置容器映像來部署具有常見 CI/CD 系統的無伺服器應用程式 AWS CodePipeline, 例如 Jenkins、GitLab CI/CD 和 GitHub Actions。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 Toolkits 偵錯 AWS SAM 應用程式](#)

AWS 工具組現在支援具有更多整合開發環境 (IDEs) 和執行時間組合的逐步偵錯。如需詳細資訊, 請參閱 [使用 AWS 工具組](#)。

2021 年 5 月 20 日

[AWS SAM CLI 與 AWS CDK 整合 \(預覽\)](#)

您現在可以使用在 AWS SAM CLI 本機測試和建置 AWS CDK 應用程式。這是公有預覽版本。如需詳細資訊，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 Public](#)

預設容器映像儲存庫從 DockerHub 變更為 [Amazon ECR Public](#)。如需詳細資訊，請參閱 [映像儲存庫](#)。

2021 年 4 月 6 日

[每 AWS SAM CLI 夜組建](#)

您現在可以安裝的發行前版本 AWS SAM CLI，該版本是每夜建置的。如需詳細資訊，請參閱 [安裝 AWS SAM CLI](#) 下您選擇的作業系統子主題的每晚建置區段。

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 `--container-env-var` 和 `--container-env-var-file` 選項 [sam build](#)。

2021 年 3 月 4 日

[新的 Linux 安裝程序](#)

您現在可以使用原生 Linux AWS SAM CLI 安裝程式安裝。如需詳細資訊，請參閱在 [Linux AWS SAM CLI 上安裝](#)。

2021 年 2 月 10 日

支援 EventBridge 的無效字母佇列

新增對 EventBridge 的無效字母佇列和無伺服器函數和狀態機器Schedule的事件來源的支援。如需詳細資訊，請參閱 [EventBridgeRule](#) 和 [Schedule](#)事件來源的 [DeadLetterConfig](#) 屬性，以了解 [AWS::Serverless::Function](#) 和 [AWS::Serverless::StateMachine](#) 資源類型。

2021 年 1 月 29 日

支援自訂檢查點

新增對 DynamoDB 的自訂檢查點和無伺服器函數的 Kinesis 事件來源的支援。如需詳細資訊，請參閱 [AWS::Serverless::Function](#) 資源類型的 [Kinesis](#)和 [DynamoDB](#)資料類型的 [FunctionResponseType](#) 屬性。

2021 年 1 月 29 日

支援輪轉時段

新增支援 DynamoDB 的輪轉時段和無伺服器函數的 Kinesis 事件來源。如需詳細資訊，請參閱 [AWS::Serverless::Function](#) 資源類型的 [Kinesis](#)和 [DynamoDB](#)資料類型的 [TumblingWindowInSeconds](#) 屬性。

2020 年 12 月 17 日

支援暖容器

新增使用命令[sam local start-api](#)和在本機測試時對暖容器的 AWS SAM CLI 支援[sam local start-lambda](#)。如需詳細資訊，請參閱這些命令 `--warm-containers` 的選項。

2020 年 12 月 16 日

支援 Lambda 容器映像

新增對 Lambda 容器映像的支援。如需詳細資訊，請參閱[建置應用程式](#)。

2020 年 12 月 1 日

支援程式碼簽署

新增對無伺服器應用程式程式碼的程式碼簽署和信任部署的支援。如需詳細資訊，請參閱[設定 AWS SAM 應用程式的程式碼簽署](#)。

2020 年 11 月 23 日

支援平行和快取建置

透過將兩個選項新增至 [sam build](#) 命令來改善無伺服器應用程式建置的效能：`--parallel` 會平行建置函數和層，而不是循序建置，而 `--cached` 會在未進行任何需要重建的變更時，使用先前建置的建置成品。

2020 年 11 月 10 日

[支援 Amazon MQ 和交互 TLS 身分驗證](#)

新增對 Amazon MQ 的支援，做為無伺服器函數的事件來源。如需詳細資訊，請參閱 [AWS::Serverless::Function](#) 資源類型的 [EventSource](#) 和 [MQ](#) 資料類型。也新增對 API Gateway APIs 和 HTTP APIs 的交互 Transport Layer Security (TLS) 身分驗證的支援。如需詳細資訊，請參閱 [AWS::Serverless::Api](#) 資源類型的 [DomainConfiguration](#) 資料類型，或 [AWS::Serverless::HttpApi](#) 資源類型的 [HttpApiDomainConfiguration](#) 資料類型。

2020 年 11 月 5 日

[支援適用於 HTTP APIs Lambda 授權方](#)

新增對 [AWS::Serverless::HttpApi](#) 資源類型的 Lambda 授權方的支援。如需詳細資訊，請參閱 [Lambda 授權方範例 \(AWS::Serverless::HttpApi\)](#)。

2020 年 10 月 27 日

[支援多個組態檔案和環境](#)

新增對多個組態檔案和環境的 AWS SAM CLI 支援，以存放命令的預設參數值。如需詳細資訊，請參閱 [AWS SAM CLI 組態檔案](#)。

2020 年 9 月 24 日

[支援 X-Ray 搭配 Step Functions，以及控制 APIs 存取時的參考](#)

新增對 X-Ray 做為無伺服器狀態機器事件來源的支援。如需詳細資訊，請參閱 [AWS::Serverless::StateMachine](#) 資源類型的 [Tracing](#) 屬性。也新增了控制 APIs 存取時的參考支援。如需詳細資訊，請參閱 [ResourcePolicyStatement](#) 資料類型。

2020 年 9 月 17 日

[支援 Amazon MSK](#)

新增對 Amazon MSK 做為無伺服器函數事件來源的支援。這可讓 Amazon MSK 主題中的記錄觸發 Lambda 函數。如需詳細資訊，請參閱 [AWS::Serverless::Function](#) 資源類型的 [EventSource](#) 和 [MSK](#) 資料類型。

2020 年 8 月 13 日

[支援 Amazon EFS](#)

新增將 Amazon EFS 檔案系統掛載至本機目錄的支援。這可讓 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](#) 屬性。

2020 年 6 月 16 日

[協調無伺服器應用程式](#)

透過使用 建立 Step Functions 狀態機器，新增協調應用程式的支援 AWS SAM。如需詳細資訊，請參閱 [使用 和 AWS 資源類型協調 AWS Step Functions](#) [AWS::Serverless::StateMachine](#) 資源。

2020 年 5 月 27 日

建置自訂執行時間	新增了建置自訂執行時間的功能。如需詳細資訊，請參閱 建置自訂執行時間 。	2020 年 5 月 21 日
建置層	新增建置個別LayerVersion 資源的能力。如需詳細資訊，請參閱 建置層 。	2020 年 5 月 19 日
產生的 CloudFormation 資源	提供 AWS SAM 產生之 CloudFormation 資源的詳細資訊，以及如何參考這些資源。如需詳細資訊，請參閱 產生的 CloudFormation 資源 。	2020 年 4 月 8 日
設定 AWS 登入資料	新增設定 AWS 登入資料的指示，以防您尚未將登入資料設定為與其他 AWS 工具搭配使用，例如其中一個 AWS SDKs 或 AWS CLI。如需詳細資訊，請參閱 設定 AWS 登入資料 。	2020 年 1 月 17 日
AWS SAM 規格和 AWS SAM CLI更新	從 GitHub 遷移 AWS SAM 規格。如需詳細資訊，請參閱 AWS SAM 規格 。也會使用 sam deploy 命令的變更來更新部署工作流程。	2019 年 11 月 25 日
控制 API Gateway APIs存取和政策範本更新的新選項	新增控制 API Gateway APIs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 APIs存取 。也更新了兩個政策範本：RekognitionFacesPolicy 和 ElasticsearchHttpPostPolicy。如需詳細資訊，請參閱 AWS SAM 政策範本 。	2019 年 8 月 29 日

[入門更新](#)

已更新入門章節，提供 AWS SAM CLI 和 Hello World 教學課程的改善安裝指示。如需詳細資訊，請參閱 [入門 AWS SAM](#)。

2019 年 7 月 25 日

[控制對 API Gateway APIs 存取](#)

新增控制 API Gateway APIs 存取的支援。如需詳細資訊，請參閱 [控制 API Gateway APIs 存取](#)。

2019 年 3 月 21 日

[已sam publish新增至 AWS SAM CLI](#)

中的 AWS SAM CLI 新 [sam publish](#) 命令可簡化在 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 SAM CLI](#)。

2018 年 12 月 21 日

[巢狀應用程式和層支援](#)

新增對巢狀應用程式和層的支援。如需詳細資訊，請參閱 [使用巢狀應用程式](#) 和 [使用層](#)。

2018 年 11 月 29 日

[已sam build新增至 AWS SAM CLI](#)

中的 AWS SAM CLI 新 [sam build](#) 命令簡化了使用相依性編譯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

[新增 的新安裝選項 AWS SAM CLI](#)

新增的 Linuxbrew (Linux)、MSI (Windows) 和 Homebrew (macOS) 安裝選項 AWS SAM CLI。如需詳細資訊，請參閱 [安裝 AWS SAM CLI](#)。

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。